Share Your Data Carefree: An Efficient, Scalable and Privacy-Preserving Data Sharing Service in Cloud Computing

Jianfei Sun, Guowen Xu, Tianwei Zhang, Hu Xiong, *Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

Abstract—Benefiting from the powerful computing and storage capabilities of cloud services, data sharing in the cloud has been permeated across various applications including social networks, e-health and crowdsourcing transportation system. Intuitively, outsourcing data to untrusted cloud commonly raises concerns about data privacy breaches. To combat this, one approach is exploiting Broadcast Based Searchable Encryption (BBSE) for secure data sharing. Nevertheless, the latest proposed BBSE is still defective in either security or efficiency. In this article, we propose ESPD, an Efficient, Scalable and Privacy-preserving Data sharing framework over encrypted cloud dataset. Different from previous works, ESPD supports sharing target data to multiple users with distinct secret keys, and keeps a constant ciphertext length with the changes of the amount of system users. This feature significantly improves search efficiency and makes ESPD scalable in real-world scenarios. We show a formal analysis to prove the security of ESPD in terms of file privacy, keyword privacy and trapdoor privacy. Also, extensive experiments on real-world dataset are conducted to indicate the desirable performance of ESPD compared to other similar schemes.

Index Terms—Searchable encryption, broadcast, privacy-preserving, scalable

1 Introduction

Cloud-assisted data sharing services, as a common feature exist in a variety of applications ranging from e-health to social networking [1], [2], [3]. For example, with social software such as Twitter, Facebook and WeChat as carriers [5], [9], one can share their own messages, pictures and voices to multiple friends, where the powerful cloud can provide customers with high quality service experience, including seamless image transmission, voice delivery and concurrent data processing [6], [7]. Meanwhile, with the popularity of cloud computing in the medical field, institutions such as hospitals, insurance companies, and pharmaceutical alliances have preferred to outsource medical data to cloud providers. Customers in this way can access authorized data anytime, anywhere without geographical restrictions. Moreover, by shifting these services to the cloud, data owners can be freed from burdensome local data storage and management burdens. Clearly, this "one-to-many" (i.e., one data owner-to-multiple users) outsourced data sharing services have become an integral part of life, which is spread across many areas such as social, medical, financial, etc.

While outsourced data sharing services possess appealing advantages, data owners may raise concerns about privacy breaches once uploading their data to the untrusted cloud [8], [10], [11]. Data owners are afraid of losing the ability to control the utilization of the data stored on the server as they can only access the data in a black-box way. In other words, the server may derive private and sensitive information, more seriously, abuse the outsourced data thereby seeking certain inappropriate benefits. On the other hand, users who retrieve data from the cloud without defense also present a risk of privacy leakages. Intuitively, a curious server is fully capable of collecting user history of queries over a period of time (such as prescriptions for cancer treatment, address, and even insurance records). This information is also sensitive and should not be exposed to any unauthorized entity (including the cloud server).

To address these privacy concerns, a straightforward way is to conduct encryption operations prior to outsourcing raw data to the cloud. However, traditional encryption methods (such as AES) significantly impair the data usability while providing a strong level of security [4], [12]. As a consequence, data sharing over the encrypted domain will be extremely difficult since users are hard to recognize the actual meaning of ciphertexts. To combat this, searchable

- *Jianfei Sun, Guowen Xu, and Tianwei Zhang are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore. E-mail: sjf215.uestc@gmail.com, {guowen.xu, tianwei.zhang}@ntu.edu.sg.*
- *Hu Xiong is with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China. E-mail: xionghu.uestc@gmail.com.*
- *Hongwei Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China. E-mail: hongweili@uestc.edu.cn.*
- *Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 178902, Singapore. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 2 Nov. 2020; revised 21 Sept. 2021; accepted 28 Sept. 2021. Date of publication 6 Oct. 2021; date of current version 8 Mar. 2023.
This work was supported by NTU-Desay Research Program 2018-0980. (Corresponding author: Guowen Xu.)
Recommended for acceptance by P. Trunfio.
Digital Object Identifier no. 10.1109/TCC.2021.3117998

encryption, as a promising technology, has been applied to secure data sharing with satisfactory efficiency. The fly in the ointment is that existing Symmetric-based Searchable Encryption (SSE) are usually suitable for one-to-one (one data owner-to-one user) data sharing scenarios. This is mainly due to the inherent limitation of symmetric encryption (i.e., require encryption and decryption operations with the same secret key), which makes ciphertext sharing with one-to-many impractical once multiple users collude with each other. Other variants of SSE require a data owner to encrypt each data with a distinct key generated for every authorized user, and then each user independently performs ciphertext searching with its unique key. Obviously, this inevitably leads to unacceptable storage redundancy, and also makes data owners fall into the trap of large-scale key management. Hence, it is crucial to design an efficient and privacy-preserving framework that is compatible with one-to-many data sharing in the ciphertext domain.

To address the above challenges, various ways have been widely investigated and applied to distinct practical scenarios. In general, existing solutions are mainly originated from two underlying technologies: *Attribute based Searchable Encryption*(ABSE) and *Broadcast based Searchable Encryption* (BBSE), both of which are constructed based on the Public Key Searchable Encryption (PKSE) primitive. ABSE is a well-known encryption algorithm that can provide data confidentiality as well as fine-grained data access control [28], [33]. Specifically, the data owner encrypts messages with a set of attribute values (i.e., access policies) so that the ciphertext could be only accessed by authorized entities who own the corresponding set of attribute values. For example, *Zheng et al.* [28] put forward a fine-grained keyword-based searchable encryption scheme. Through an access policy over encrypted data, the authors stated that the scheme can achieve secure data sharing with multiple data receivers. Recently, *Miao et al.* [36] also proposed PP-ABDS, a privacy-preserving attribute-based data sharing scheme for secure cloud storage. Based on the bilinear pairing and Decisional Diffie-Hellman (DDH) assumption, PP-ABDS is proved to be secure against selectively chosen keyword attack (SCKA). While ABSE schemes can provide expressive access policies over the encrypted data to be shared and retrieved, many ABSE have been proven to be inefficient for sharing and searching [32], [34], [35]. Concretely, in ABSE, the size of each ciphertext (such as keywords and search token) is linearly incremental to the amount of attributes in the access policy, which means that data owner needs to scale up each data to a large ciphertext once the quantity of attributes involved in the system reaches a certain large magnitude. Moreover, most ABSE-based solutions require the assistance of a trusted entity to produce the search token, which also leads to poor scalability in most real-world data sharing scenarios [29], [33]. To break the above restrictions, *Kiayias et al.* [13] proposed *Broadcast based Searchable Encryption* (BBSE), the first privacy-preserving data sharing scheme with the property of constant-size ciphertext, which exploits an aggregation approach called broadcast encryption technology to realize that the computation cost of the entire search process is independent of the number of users in the system. Besides, compared with ABSE, it also enhances the level of privacy protection, especially concerning raw data privacy and keyword privacy. However, as we will show in this paper, BBSE schemes [13] fail to achieve its claimed security, and even more, it could neither perfectly support the function of secure multi-user data sharing nor realize the privacy-preserving of the retrieved contents. Therefore, as far as our knowledge goes, there is no prior work that can realize lightweight and privacy-preserving data sharing for the one-to-many scenarios.

In this paper, we raise ESPD, an Efficient, Scalable and Privacy-preserving Data sharing framework over encrypted cloud dataset. Specifically, to preserve the keyword privacy of index and trapdoor, the linear splitting technique is utilized to split the bilinear group element into two pieces, which prevents malicious entities from deriving keyword privacy from ciphertext and trapdoor. To achieve efficiency as well as scalability, the aggregation-based broadcast technique is used to realize the constant-size ciphertext irrespective of the number of users.

The main contributions are as follows:

- We first point out the security vulnerabilities of the data sharing schemes in BBSE [13] by demonstrating our attack and stating the reason why they fail to achieve the claimed privacy, i.e., keyword privacy and trapdoor privacy.
- We design an efficient and privacy-preserving data sharing framework, which supports sharing target data with multiple-users while allowing authorized users to search a keyword in a subset of files.
- We avail of an aggregation method to realize the constant-size ciphertext and a linear splitting technique to solve the keyword privacy leakage issues of index and trapdoor. With these solutions, our ESPD realizes a relatively better performance in network bandwidth usage and cloud storage. Besides, our ESPD is desirable in file privacy, keyword privacy and trapdoor privacy.
- File privacy, keyword privacy and trapdoor privacy in our ESPD are formally proven. Further, we show theoretical and experimental evaluations to demonstrate the efficiency and practicality of our ESPD.

Organization. The rest of current manuscript is illustrated below. In Sections 2 and 3, the preliminaries and problem definitions considered in this manuscript are introduced. Then, Section 4 describes the security definitions. In Sections 5 and 6, our ESPD as well as the security analysis is stated. Section 7 elaborately shows the implementation and permanence evaluation. Finally, Section 8 introduces the related works of privacy-preserving data sharing, and Section 9 summarizes this manuscript.

2 System Model and Threat Model

This section briefly describes the general scenario considered in our ESPD , which consists of system & threat model and privacy requirements.

2.1 System Model

As presented in Fig. 1, three generic entities in our model are taken into consideration, i.e., *data owner*, *cloud server* and

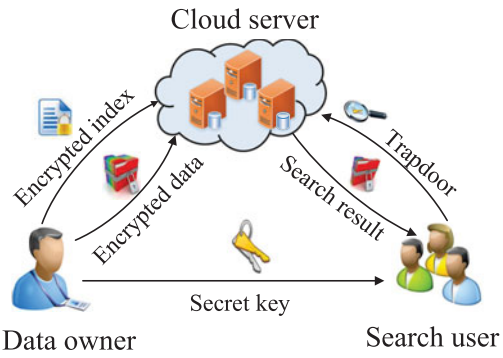


Fig. 1. System model.

search users, and the three of them play the following roles in ESPD, respectively.

- *Data owner*: The data owner is required to implement encryption on raw data prior to uploading them to the cloud. To facilitate data sharing under the ciphertext, each data is linked to an encrypted index constructed from a single keyword and a subset of users, which signifies that only the users to be authorized (belongs to the subset above) holding the target keyword are allowed to access this data. Then, the encrypted data along with encrypted indexes are subsequently transmitted to the cloud by data owner.
- *Cloud server*: The primary responsibility of the cloud server is to maintain the integrity of the ciphertext data stored on it, thereby providing a private data sharing service for multiple users. To be specific, upon capturing a search query from a legitimate user, the server is asked to retrieve the target data that the user is authorized to, and returns the ciphertext results that is consistent with the currently queried keyword.
- *Search users*: In ESPD, each legitimate user will be assigned a distinct secret key used to generate encrypted search tokens (i.e., trapdoors). Then, given a keyword, the legitimate user is free to generate the trapdoor to be uploaded to the server, and obtain authorized ciphertexts with the assistance of the cloud server.

We can observe that the above system model is universal and has implemented in various real-world applications. For example, in social networks, the data owner can share various multimedia messages including photos and videos to a number of friends using well-known cloud services (such as iCloud and Amazon Web Services). In the transportation field, pedestrians and drivers can also share their traffic conditions with cloud assistance, thus ensuring the performance of vehicle networking in real-time navigation and road condition perception. Clearly, this "one-to-many" outsourced data sharing services have become an integral part of life, which is spread across many areas such as social, traffic and medical.

2.2 Threat Model and Privacy Requirements

In our ESPD, the adversaries mainly originate from the untrusted server and some malicious users. Specifically, we assume that the cloud server is *honest-but-curious* [2], which implies the pre-agreed protocols are honestly conducted by the cloud to complete its mission. However, users' data privacy may be also attempted to be snooped by exploiting mastered prior knowledge. Every search user is considered malicious, and we allow them to collude with each other with the most offensive ability to try to gain the privacy of other honest users. In addition, the data owner is considered to be trustworthy since it is the requester of the cloud service.

Under the aforementioned threat model, the following privacy requirements are put forward.

- *Confidentiality of data owner's raw data*: As described before, the data owner's raw data may be inherently sensitive, or involve private information such as medical records, addresses and IDs. Hence, these raw data should be outsourced in the form of ciphertext to the cloud server, and can only be accessed by legitimate authorized users.
- *Privacy protection of indexes and trapdoors*: In our ESPD, indexes and trapdoors are constructed from keywords and access policies to facilitate querying under the encrypted domain. Obviously, the leakage of the indexes or trapdoors inevitably gives the adversary more clues to derive the privacy of the original data. Therefore, the contents of them should be hindered from being disclosed to other parties (such as the untrusted server and illegal users).

3 Preliminaries

Some fundamental cryptographic primitives are briefly reviewed for ease of helping readers understand the technical particulars of the proposed schemes.

3.1 Bilinear Map

Definition 1 (Bilinear map). *Assume* $\{\mathbb{G}_i\}_{i\in\{0,1\}}$ *are multiplicative groups with prime order* p*, where* $\mathbb{G}_0$ *owns a generator* g*. Let* $e:\mathbb{G}_0\times\mathbb{G}_0\rightarrow\mathbb{G}_1$ *be a computable bilinear map [15] with the properties below: (1) Bilinearity: for all* $x,y\in\mathbb{G}_0$ *and* $r,t\in\mathbb{Z}_p$*,* $e(x^r,y^t)=e(x,y)^{rt}$*. (2) Non-Degeneracy:* $e(g,g)\neq 1$*.*

3.2 Complexity Assumptions

Definition 2 (BDHE problem). *On input an instance* $(g,h,g_1,\ldots,g_n,g_{n+2},\ldots,g_{2n},\mathcal{Z})$*, where* g *is a generator of* $\mathbb{G}_0$ *and* $g_i=g^{\tau^i}$ *for some unknown* $\tau\in\mathbb{Z}_p$*, it is intractable to distinguish* $\mathcal{Z}=e(g_{n+1},h)$ *or* $\mathcal{Z}=\mathcal{U}$ *for the bilinear Diffie-Hellman exponentiation assumption (BDHE) [17], where* $\mathcal{U}$ *is a random element.*

Definition 3 (DLIN problem). *Given a tuple* $(g,g^{x_1},g^{x_2},g^{x_1x_3},g^{x_2x_4},\mathcal{Z})$*, where* g *is a generator of* $\mathbb{G}_0$*, it is intractable to discern* $\mathcal{Z}=g^{x_3+x_4}$ *from* $\mathcal{Z}=\mathcal{U}$ *for the decision linear (DLIN) problem [17], where* $\mathcal{U}$ *is chosen at random.*

Definition 4 (DDHI assumption). *Given a tuple* $(g,h_1,h_2,g_1,\ldots,g_n,g_{n+1},\ldots,g_{2n},g^{\varphi},\mathcal{Z})$*, where* $\{g_i=g^{\tau^i}\}_{i=1,\ldots n,n+2,\ldots 2n}$*, it is intractable to differentiate* $\mathcal{Z}=h_1^{\frac{\varphi}{\tau^{n+1}}}$ *from* $\mathcal{Z}=\mathcal{U}$ *for the decision Diffie-Hellman inverse (DDHI) assumption [13], where* $\mathcal{U}$ *is picked at random.*

3.3 Formal Definition

Definition 5 (ESPD system). *Our ESPD consists of the following algorithms:*

- Setup (n, λ)*: Accepts the security parameter* λ *and the number of users* n*, returns the system public key* pk *and the master secret key* msk*.*
- KeyGen (i, pk, msk)*: Takes the system public key* pk *and the master secret key* msk*, returns a secret key* sk_i *for user* i*.*
- Enc (pk, w, S_k, M_k)*: Accepts the system public key* pk*, the keyword* w*, file* M_k *and an user set* S_k*, returns a keyword ciphertext* C_k *and file ciphertext* C'_k*.*
- Trap (sk_i, w)*: Takes the secret key* sk_i *of user* i*, generates a search token called trapdoor* $t_{i,w}$*.*
- Test $(pk, S_k, t_{i,w}, C_k)$*: Takes the public key* pk*, the keyword ciphertext* C_k*, the user set* S_k*, the trapdoor* $t_{i,w}$ *of user* i*, checks whether the target keyword* w *is found in* C_k*. If it holds, it returns 1. Otherwise, it aborts and returns 0.*
- Dec (pk, sk_i, C'_k, S_k)*: Takes the the public key* pk*, the user set* S_k*, the secret key* sk_i *of user* i *and the file ciphertext* C'_k*, recovers the encrypted file* M_k*.*

4 Security Definition

In this section, we define the security games for file privacy, keyword privacy and trapdoor privacy of our ESPD.

Definition 6 (File privacy). *File privacy means that any malicious entities including cloud servers cannot derive plaintext privacy from data ciphertexts except that they are authorized. The semantic security for file privacy is defined via the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

- *Init*: $\mathcal{A}$ gives $\mathcal{C}$ a set $\mathcal{S}_k$ that he wants to challenge on.
- *Setup*: $\mathcal{C}$ performs *Setup* algorithm to obtain the system public key pk and sends it to $\mathcal{A}$.
- *Phases 1 & 2*: $\mathcal{A}$ adaptively sends secret key queries for $j \notin S_k$ to $\mathcal{C}$, $\mathcal{C}$ then responds $\mathcal{A}$ with the secret key sk_j generated by performing *KeyGen* algorithm.
- *Challenge*: $\mathcal{A}$ sends two equal length plaintexts M_0 and M_1 to $\mathcal{C}$, $\mathcal{C}$ then selects a random number $r \in \{0, 1\}$, runs *Enc*(pk, M_r, S_k) to produce the ciphertext C', which is subsequently sent to $\mathcal{A}$ for guessing.
- *Guess*: $\mathcal{A}$ outputs a guess r' for r and wins the game if $r = r'$.

We say that the file privacy game is secure if the probability $|Pr[r = r'] - 1/2| \leq \epsilon$, where ϵ is a negligible probability.

Definition 7 (Keyword privacy). *Keyword privacy indicates that any malicious users or cloud servers cannot snoop the encrypted keyword information from the keyword ciphertexts. The semantic security for keyword privacy is defined via the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

- *Init*: $\mathcal{A}$ gives $\mathcal{C}$ a set $\mathcal{S}_0$ and the keyword w^* that he wants to challenge on.
- *Setup*: $\mathcal{C}$ performs *Setup* algorithm to obtain the system public key pk and the master secret key msk, then sends the pk to $\mathcal{A}$.
- *Phases 1 & 2*: $\mathcal{A}$ adaptively sends trapdoor queries for (i, w) to $\mathcal{C}$, $\mathcal{C}$ then responds $\mathcal{A}$ with the trapdoor $t_{i,w}$ generated by performing *Trap* algorithm.
- *Challenge*: $\mathcal{C}$ selects a random number $r \in \{0, 1\}$, runs $\mathbf{Enc}(pk, w_r, S_0)$ to produce the ciphertext C, where $w_0 = w^*$ and w_1 is a random keyword, and then returns the produced challenge ciphertext C to $\mathcal{A}$.
- *Guess*: $\mathcal{A}$ outputs a guess r' for r and wins the game if $r = r'$.

Restriction. The trapdoor queries can be queried by the adversary $\mathcal{A}$ only in case that $i \notin S_0$ and $w \neq w^*$ hold. The scheme is keyword private if $Pr[win_{\mathcal{A}}(r = r')] < 1/2 + \epsilon$, where $win_{\mathcal{A}}(r = r')$ is a random variable showing whether $\mathcal{A}$ wins the game and ϵ is a negligible probability.

Definition 8 (Trapdoor privacy). *Trapdoor privacy refers to that neither malicious users nor cloud server can learn the keyword information from trapdoors of other participants or delegated users. In the security game of the security game, the goal of the adversary is to obtain the trapdoor privacy of other participants. The semantic security for trapdoor privacy is defined via the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

- *Init*: $\mathcal{A}$ gives $\mathcal{C}$ a user and keyword tuple (i^*, w^*) that he wants to challenge on.
- *Setup*: $\mathcal{C}$ performs *Setup* algorithm to obtain the system public key pk and the master secret key msk, then sends the pk to $\mathcal{A}$.
- *Phases 1 & 2*: $\mathcal{A}$ adaptively sends trapdoor queries for (i, w) to $\mathcal{C}$, $\mathcal{C}$ then responds $\mathcal{A}$ with the trapdoor $t_{i,w}$ generated by performing *Trap* algorithm.
- *Challenge*: $\mathcal{C}$ selects a random number $r \in \{0, 1\}$, runs *Enc*(pk, w_r, S_0) to produce the ciphertext C and C', where $w_r = w^*$ if $r = 0$ and w_1 is a random keyword, and then returns the produced challenge ciphertexts C and C' to $\mathcal{A}$.
- *Guess*: $\mathcal{A}$ outputs a guess r' for r and wins the game if $r = r'$.

We say that the scheme is trapdoor private if $\Pr[win_{\mathcal{A}}(r = r')] < 1/2 + \epsilon$, where $win_{\mathcal{A}}(r = r')$ is a random variable indicating whether $\mathcal{A}$ wins the game and ϵ is a negligible probability.

Remark 1. In our manuscript, the goal of our paper is to do our best to achieve a Secure Identity-based Broadcast Searchable Encryption for data sharing, which can be viewed as the modified and enhanced version of Efficient Encrypted Keyword Search for Multi-user Data Sharing scheme [13]. In [13], although some security defeats leading to keyword privacy leakage have existed, the corrected security definitions for file privacy, keyword privacy and trapdoor privacy are correctly defined. The superficial reason resulting in security vulnerabilities in [13] originates from that the access control list S_k does not make sense at all, such that the adversary can bypass the access structure and directly access the data. The basic reason is that the adversary can self-produce the authorized trapdoor and then pick partial ciphertexts to be challenged on to snoop the keyword privacy. In the definition of our security games, we follow the original security definitions as those defined in [13]. Specifically, in the keyword privacy game, the adversary is static that he/she

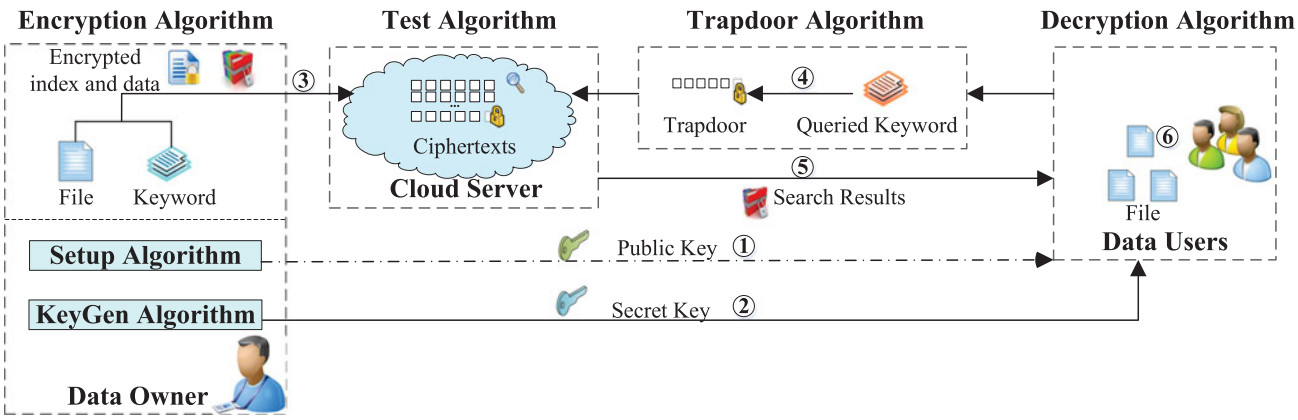


Fig. 2. Workflow of our ESPD.

outputs a keyword and a set pair that he/she wants to be challenged on. Then he observes encryption of keywords and trapdoors. However, he/she is not able to distinguish whether the challenge ciphertext is encoded by the challenge keyword or a random keyword. In the trapdoor privacy game, the adversary outputs challenge user index and keyword pair at the beginning of the game to be challenged on. Then, the adversary observes encryptions of keywords and the challenge trapdoor, but he/she is not able to distinguish the challenge keyword from a random keyword.

5 Our Constructions

5.1 High Level of Our Construction

In this section, we propose ESPD, which consists of two privacy-preserving data sharing frameworks, named ESPD-I and ESPD-II (shown in Figs. 3 and 4). At a high level view, the workflow of ESPD shown in Fig. 2 is briefly described as follows: First, algorithm *Setup* is exploited to initialize public parameters used in the system (Step ①). Based on this, each system user is granted a secret key with *KeyGen* algorithm (Step ②). Then, to protect data privacy, the data owner resorts to *Enc* algorithm to encrypt data to be outsourced, and generated corresponding encrypted indexes for subsequent queries (Step ③). When a user desires to access the data of his interest, he first requires to locate the target data. In this way, he produces the search token called trapdoor with *Trapdoor* algorithm and then delegates it to the cloud server for retrieving the interest's data (Step ④). After receiving the submitted trapdoor, the cloud server performs the *Test* algorithm to search the target data and returns it to the delegated user if the target data is retrieved (Step ⑤). Finally, the data user conducts *Dec* algorithm to recover the raw data (Step ⑥).

Compared to *Kiayias et al.*'s data sharing system [13] (see *Supplemental Material A*, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCC.2021.3117998 for detailed security analysis), our ESPD-I addresses the issues of non-authorization access and keyword leakage. To realize one-to-many authorized data access and privacy-preserving of keyword information, we resort to the primitives of broadcast encryption [16] and expressive searchable encryption [33]. Specifically, the "aggregation" technique [16] provides an effective solution to realize one-to-many authorized data sharing with constant-size ciphertexts and secret key. The "linear splitting" technique [33] is used to split ciphertexts corresponding to each keyword into two randomized components. Even if keyword information is still contained in ciphertexts, it is infeasible to be computationally derived from the public parameters and the ciphertexts. Our constructed ESPD-I enables a single server to conduct authorized keyword retrieval over the target data ciphertext and reaches the goals of particular privacy requirements, i.e., privacy preserving of index and trapdoor. For technical details, please refer to Fig. 3.

It should be noted that there is still an unresolved privacy issue in ESPD-I, i.e., the linkability among trapdoors, which means that whether any two delegated trapdoors contain the same encrypted keyword can be easily discerned by the server. This may incur some leakage of trapdoor information. In real applications, any user may submit his trapdoor to a cloud server for retrieving the data of his interests, and even the same user may send the search queries several times for retrieval. For the cloud server, it can distinguish whether different users search the same interested data by checking the trapdoors that encrypt the same information. To combat that, we propose an enhanced model called ESPD-II. In ESPD-II, the re-randomness method is utilized to re-randomize trapdoor components to match with related components in the ciphertext, such that the server cannot distinguish whether the delegated trapdoors encrypt the same keyword. With our ESPD-II, keyword privacy of both ciphertext and trapdoor can be protected from being snooped. Further, the server cannot identity whether the received trapdoors hide the same keyword information. For technical details, please refer to Fig. 4.

Remark 2. Compared with the works [13], our ESPD (ESPD-I and ESPD-II) elegantly solves the security vulnerabilities of the privacy leakage of keyword ciphertext and trapdoor appeared in [13] without sacrificing the efficiency. With our ESPD to construct data sharing service, authorized users can securely and efficiently retrieve a keyword, while keyword privacy of ciphertext and trapdoor cannot be derived by the server.

5.2 Detailed Descriptions of Our Construction

In our construction, the trapdoor consists of six different trapdoors, both the secret key and the ciphertext have

- **Setup** (n,λ): It initially chooses $\tau \in \mathbb{Z}_p$ and computes $g_i = g^{\tau^i}$, where $i \in [1,2n]\backslash\{n+1\}$. Next, $\alpha, \beta, \theta, \gamma, \varphi, \varphi' \in \mathbb{Z}_p$ are randomly picked, and then compute $T_1 = g^{\alpha}$, $T_2 = g^{\beta}$, $T_3 = g^{\theta}$, $T_4 = g^{\gamma}$, $\Phi = g^{\varphi}$ and $\Phi' = g^{\varphi'}$. After that, it selects $u, v, h \in \mathbb{G}_0$ at random. Finally it sets $pk = (g, \{g_i\}_{i\in[1,2n]\backslash\{n+1\}}, T_1, T_2, T_3, T_4, \Phi, \Phi', u, v, h)$ and $msk = (\alpha, \beta, \theta, \gamma, \varphi, \varphi')$.
- **KeyGen** (i, pk, msk): For user i, it picks two random values $z_i, z_i' \in \mathbb{Z}_p$ and computes secret key $sk_i = (sk_{i,1}, \ldots, sk_{i,11})$ as follows:

$$sk_{i,1} = g_i^{\varphi} \cdot v^{-(\alpha\beta z_i + \theta\gamma z_i')}, sk_{i,2} = g^{\alpha\beta z_i + \theta\gamma z_i'}, sk_{i,3} = u^{-\beta z_i},$$
$$sk_{i,4} = h^{-\beta z_i}, sk_{i,5} = u^{-\alpha z_i}, sk_{i,6} = h^{-\alpha z_i}, sk_{i,7} = u^{-\gamma z_i'},$$
$$sk_{i,8} = h^{-\gamma z_i'}, sk_{i,9} = u^{-\theta z_i'}, sk_{i,10} = h^{-\theta z_i'}, sk_{i,11} = g_i^{\varphi'}.$$

- **Enc** (pk, w, S_k, M_k): Data owner randomly picks $r, s, s', t, t' \in \mathbb{Z}_p$ for keyword w and plaintext message M_k as follows:

$$K = e(g_n, g_1)^s, K' = e(g_n, g_1)^{s'}, C_{k,1} = v^{-s}(u^w h)^r, C_{k,2} = T_1^{r-t},$$
$$C_{k,3} = T_2^t, C_{k,4} = T_3^{r-t'}, C_{k,5} = T_4^{t'}, C_{k,6} = g^s, C_{k,7} = (\Phi \prod_{j\in S_k} g_{n+1-j})^s,$$
$$C_{k,8} = K, C_{k,9} = g^{s'}, C_{k,10} = (\Phi' \prod_{j\in S_k} g_{n+1-j})^{s'}, C_{i,11} = K' \cdot M_k.$$

  Denote $C_k = (C_{k,1}, \ldots, C_{k,8})$ and $C_k' = (C_{k,9}, C_{k,10}, C_{k,11})$ as the first part of the keyword ciphertext and the second part of file ciphertext, respectively.
- **Trap** $(sk_{i,1}||\ldots||sk_{i,6}, w)$: User i computes the trapdoor $t_{i,w} = (tr_1, \ldots, tr_6)$ as follows:

$$tr_1 = sk_1 = g_i^{\varphi} \cdot v^{-(\alpha\beta z_i + \theta\gamma z_i')}, tr_2 = sk_2 = g^{\alpha\beta z_i + \theta\gamma z_i'},$$
$$tr_3 = sk_3^w \cdot sk_4 = ((u^w h)^{-\beta})^{z_i}, tr_4 = sk_5^w \cdot sk_6 = ((u^w h)^{-\alpha})^{z_i},$$
$$tr_5 = sk_7^w \cdot sk_8 = ((u^w h)^{-\gamma})^{z_i'}, tr_6 = sk_9^w \cdot sk_{10} = ((u^w h)^{-\theta})^{z_i'}.$$

- **Test** $(pk, S_k, t_{i,w}, C_k)$: The server checks if

$$C_{k,8} \stackrel{?}{=} \frac{e(g_i, C_{k,7}) \cdot T}{e(tr_1 \prod_{j\in S_k, i\neq j} g_{n+1-j+i}, C_{k,6})},$$

  where $T = e(tr_2, C_{k,1}) \cdot e(tr_3, C_{k,2}) \cdot e(tr_4, C_{k,3}) \cdot e(tr_5, C_{k,4}) \cdot e(tr_6, C_{k,5})$. If the above equation holds, it returns the test result 1. Otherwise, it returns 0.
- **Dec** $(pk, sk_{i,11}, C_k', S_k)$: Once the result 1 is returned from the **Test** algorithm, then the cloud server sends S_k, C_k' to user i. Then, user i does decryption to recover K' by computing $K' = \frac{e(g_i, C_{k,10})}{e(sk_{i,11} \cdot \prod_{j\in S_k, i\neq j} g_{n+1-j+i}, C_{k,9})}$, and then extracts the plaintext message M_k by calculating $M_k = C_{i,11}/K'$.

Fig. 3. Implementation of ESPD-I.

eleven parts. The reason leading to these settings originates from the need for security proofs by embedding hard problems into those redundant secret keys and ciphertexts and the trapdoor is generated with partial secret keys. For ease of better illustration and understanding, the attribute-based searchable encryption schemes (ABSE) [28], [29], [30], [31] are taken as examples. In ABSE [28], [29], [30], [31], the trusted authority generates secret keys associated with a set of attributes for each system user. A data owner selects an access policy to encrypt his/her files to produce the corresponding ciphertexts, which are generally stored on the cloud server for data sharing and searching. To locate the target ciphertext of his/her interests, a user uses his partial secret keys to create a search token (also called trapdoor), which is commonly delegated to a cloud server for performing a search query. After receiving the target ciphertexts from the cloud server, the user uses the rest of the other partial secret keys to recover the encrypted files. Similar to [28], [29], [30], [31], in our designed scheme, a users secret key associated with his/her own identity i is produced by the trusted authority. When a user desires to search his interests data, he/she just uses his/her partial secret keys $(sk_{i,1}, \ldots, sk_{i,10})$ to produces the trapdoor $(tr_{i,1}, \ldots, tr_{i,6})$ for finding the corresponding ciphertext. After locating the target ciphertext, the user uses the rest of the other secret key ski,10 to recover the encrypted files. From the above descriptions, its easy to learn that this trapdoor indeed consists of six different trapdoors, but the secret key to decrypt files actually consists of a single key.

In the first encryption of our designed scheme, the ciphertext setting could have been set as $(C_1, C_2, C_3, C_6, \ldots, C_{11})$, but it is eventually set as $(C_1, \ldots, C_{11})$. Correspondingly, the secret key could be set as $(sk_{i,1}, \ldots, sk_{i,7}, sk_{i,11})$, but it actually is set as $(sk_{i,1}, \ldots, sk_{i,11})$. The reason leading to these settings originates from the need for security proofs by embedding hard problems into those redundant ciphertexts. As we all know, in most of the public key cryptosystems, the designed schemes are closely related to the hard problems for the

- Here we only present the **Enc**, **Trap** and **Test** algorithms, and omit the **Setup**, **KeyGen** and **Decrypt** algorithms due to the same as those of our first scheme. Readers can refer to Fig. 3 for related algorithms.
- **Enc** (pk, w, S_k, M_k): Data owner randomly picks $r, s, s', t, t' \in \mathbb{Z}_p$ for keyword w and plaintext message M_k as follows:

$$K = e(g_n, g_1)^s, K' = e(g_n, g_1)^{s'}, C_{k,1} = v^{-s}(u^w h)^r, C_{k,2} = T_1^{r-t},$$
$$C_{k,3} = T_2^t, C_{k,4} = T_3^{r-t'}, C_{k,5} = T_4^{t'}, C_{k,6} = g^s, C_{k,7} = (\Phi \prod_{j \in S_k} g_{n+1-j})^s,$$
$$C_{k,8} = K, C_{k,9} = g^{s'}, C_{k,10} = (\Phi' \prod_{j \in S_k} g_{n+1-j})^{s'}, C_{i,11} = K' \cdot M_k.$$

  Denote $C_k = (C_{k,1}, \ldots, C_{k,11})$, $C'_k = (C_{k,6}, C_{k,7}, C_{k,8})$ and $C''_k = (C_{k,9}, C_{k,10}, C_{k,11})$ as the first part of the ciphertext, the second part of ciphertext and the third part of the ciphertext, respectively. In this phase, the main server $\mathcal{S}_m$ stores C_k, C''_k, S_k , and the aided one $\mathcal{S}_a$ retains C'_k, S_k.
- **Trap** $(sk_{i,1}||\ldots||sk_{i,6}, w)$: User i picks σ, σ_1, σ_2 such that $\sigma = \sigma_1 + \sigma_2$ and computes the trapdoor $t_{i,w} = (tr_1, \ldots, tr_6)$ as follows:

$$tr_1 = sk_1^\sigma = g_i^{\varphi\sigma} \cdot v^{-(\alpha\beta z_i + \theta\gamma z'_i)\sigma}, tr_2 = sk_2^\sigma = g^{(\alpha\beta z_i + \theta\gamma z'_i)\sigma},$$
$$tr_3 = (sk_3^w \cdot sk_4)^\sigma = ((u^w h)^{-\beta})^{z_i\sigma}, tr_4 = (sk_5^w \cdot sk_6)^\sigma = ((u^w h)^{-\alpha})^{z_i\sigma},$$
$$tr_5 = (sk_7^w \cdot sk_8)^\sigma = ((u^w h)^{-\gamma})^{z'_i\sigma}, tr_6 = (sk_9^w \cdot sk_{10})^\sigma = ((u^w h)^{-\theta})^{z'_i\sigma}.$$

  Finally, $t_{i,w}$ and σ_2 are sent to the main server $\mathcal{S}_m$ and σ_1 is sent to the aided server $\mathcal{S}_a$.
- **Test** $(pk, S_k, f(C_k, \sigma_2), f(C'_k, \sigma_1), t_{i,w}, C_k)$: Both the two servers compute $A_k = \frac{e(g_i, C_{k,7})}{C_{k,8} \cdot e(\prod_{j \in S_k, i \neq j} g_{n+1-j+i}, C_{k,6})}$ for file k. The aided server $\mathcal{S}_a$ sends $f(C_k, \sigma_2) = A_k^{\sigma_2}$ to the main server $\mathcal{S}_m$, where f denotes a function that can easily perform exponentiation computation. The main server $\mathcal{S}_m$ computes $f(C_k, \sigma_2) f(C'_k, \sigma_1) = A_k^{\sigma_2} A_k^{\sigma_1} = A_k^\sigma$ and determines whether $A_k^\sigma \stackrel{?}{=} \frac{e(tr_1, C_{k,6})}{T}$, where $T = e(tr_2, C_{k,1}) \cdot e(tr_3, C_{k,2}) \cdot e(tr_4, C_{k,3}) \cdot e(tr_5, C_{k,4}) \cdot e(tr_6, C_{k,5})$. If the above equation passes, $\mathcal{S}_m$ outputs 1. Otherwise, 0 is returned. Fig. 5 shows the whole search process for a user i.

Fig. 4. Implementation of ESPD-II.

security proofs. In other words, if a proposed scheme is proven to be secure, then the scheme must be bound to one or more hard problems. Similar to the schemes [33], [42], [43] for the need of security proofs, our scheme also needs to set more redundant parts of ciphertexts and secret keys. Namely, our scheme requires to set both the whole ciphertext and secret key as eleven different parts for proving the security.

From the concrete construction of our efficient privacy-preserving data sharing scheme, it is not hard to observe that our scheme is an identity-based broadcast searchable encryption (BBSE) scheme, which essentially belongs to one of the identity-based encryption crypto-primitives. In the existing public-key encryption schemes, such as identity-based encryption schemes and attribute-based encryption schemes, etc., a third-party authority is considered as a trusted entity to produce secret keys and public keys for all system users. Specifically, the trusted authority takes charge of producing and distributing the secret keys for a user based on his/her identity or attribute set. A data owner encrypts his/her files by specifying an access control or an access list to decide who can access them. The user can be eligible to decrypt the encrypted data in the case that his/her identity or attribute set satisfies the access policy hidden in the ciphertext. For our BBSE scheme for data sharing, indeed, as you understand, whenever a data owner encrypts a file, the ciphertext is associated with a set of users that can decrypt it. It is deserved to notice that the user list S (access control) embedded in ciphertext determines that only the authorized users can decrypt the encrypted files. This implies that the condition for decrypting files is that the user identity hidden in the secret key must satisfy the access control embedded in the ciphertext. In our BBSE-based data sharing scheme, even if massive of files are encrypted, as long as the access controls of these 1000 files all specify user i to be able to access, then the user only needs to store one his/her own private key to decrypt all these files.

In the encryption of our second scheme, whenever a data owner encrypts a file and a keyword, the ciphertext is associated with a set of users that can search and then decrypt it. It is deserved to notice that the user list S_k (access control) embedded in ciphertext determines that only the authorized users can retrieve and then decrypt the encrypted files. This implies that the condition for retrieving and decrypting files is that the user identity hidden in the secret key must satisfy the access control embedded in the ciphertext. In the encryption of our first scheme, $C_k = (C_{k,1}, \ldots, C_{k,8})$ and $C'_k = (C_{k,9}, \ldots, C_{k,11})$ are keyword ciphertext and file ciphertext, respectively. When a user identity i satisfies the user list S_k, it means she/he has been authorized to retrieve the corresponding keyword and then access the encrypted file. In the encryption of our second scheme, $C_k = (C_{k,1}, \ldots, C_{k,11})$, $C'_k = (C_{k,6}, \ldots, C_{k,8})$ and $C''_k = (C_{k,6}, \ldots, C_{k,8})$ denote the whole keyword and file ciphertext, partial keyword ciphertext, and file ciphertext, respectively. When a user desires to retrieve his/her target keyword ciphertext, he/she delegates his/her trapdoor to the main cloud server and sends a quest of

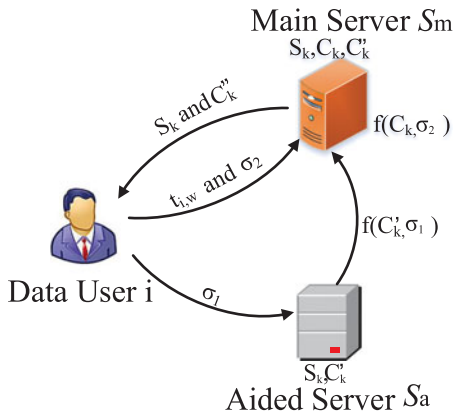


Fig. 5. Search process in ESPD-II.

operation on $((C_{k,6}, \ldots, C_{k,8})$ to the aided cloud server. Then, the cloud servers perform operations on $(C_{k,6}, \ldots, C_{k,8})$. After that, the main server implements search operations to find target keyword ciphertext and return corresponding file ciphertext if the delegated trapdoor is legitimate and authorized. Finally, the file ciphertext can be recovered with an authorized secret key.

6 Security Analysis

This part analyzes the security of our ESPD in detail. Due to the space limitations, here we only show the security proof of ESPD-II, and put the proof of ESPD-I in the *Supplemental Material B*, available in the online supplemental material. Interested Readers can refer to the supplemental material for its detailed proofs.

6.1 Proof of File Privacy

Lemma 1 (File privacy). *Provided that n-BDHE assumption holds, then our ESPD-II has file privacy.*

Proof 1. Assume that there is an adversary $\mathcal{A}$ that can breach the security game, then another algorithm $\mathcal{B}$ that must be existed with certain advantage ϵ solves the n-BDHE problem. On input $(g, h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, \mathcal{R})$, where $g_i = g^{\tau^i}$ and for the unknown $\tau \in \mathbb{Z}_p$, the $\mathcal{B}$'s goal is to distinguish $\mathcal{R} = e(g_{n+1}, h)$ or $\mathcal{R}$ is a random.

- *Init*: A challenge set S_k is picked and then transmitted to $\mathcal{B}$.
- *Setup*: $\mathcal{B}$ picks $d \in \mathbb{Z}_p$ randomly, generates $\Phi' = g^d \prod_{j \in S_k} g_{n+1-j}^{-1} = g^{d - \sum_{j \in S_0} \tau^{n+1-j}} = g^{\varphi'}$ and sets $g_i = g^{\tau^i}$. After that, $\mathcal{B}$ publishes the public parameter $pk = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, \Phi')$.
- *Phases 1 & 2*: $\mathcal{B}$ produces the secret key for user $i \notin S_k$ as $sk_{i,11} = g_i^d \prod_{j \in S_k} g_{n+1-j+i}^{-1} = g^{\tau^i (d - \sum_{j \in S_k} \tau^{n+1-j})} = g_i^{\varphi'}$ and then sends it to $\mathcal{A}$.
- *Challenge*: $\mathcal{A}$ gives two messages M_0 and M_1 to $\mathcal{B}$. Then $\mathcal{B}$ flips a coin and produces the challenge ciphertext by setting $C_{k,9} = g^{s'} = h$ for unknown s' and $C_{k,10} = h^d = (g^d)^{s'} = (g^d \prod_{j \in S_k} g_{n+1-j}^{-1} \prod_{j \in S_k} g_{n+1-j})^{s'} = (\Phi' \prod_{j \in S_k} g_{n+1-j})^{s'}$. Then $\mathcal{B}$ sends the challenge ciphertext $CT_k^* = (M_r \cdot \mathcal{R}, C_{k,9}, C_{k,10})$ for file k to $\mathcal{A}$.
- *Guess*: A guess $r' \in \{0, 1\}$ is given by $\mathcal{A}$, and if $r = r'$, then $\mathcal{B}$ returns 1; otherwise outputs 0.
- *Analysis*: If $\mathcal{R} = e(g_{n+1}, h)$, the real game is then simulated by this game. So, r is guessed correctly by $\mathcal{A}$ with the probability $1/2 + \epsilon$. If $\mathcal{R}$ is a random, r is guessed correctly with the probability $1/2$. □

6.2 Proof of Keyword Privacy

Let $[C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6}, C_{i,7}, C_{i,8}]$ be the challenge ciphertext that is delivered to $\mathcal{A}$. In addition, R, R' are random elements. The following sequence of hybrid games are the definitions in producing challenge ciphertext for $\mathcal{A}$:

- Game $\mathcal{G}_0$: The challenge ciphertext is expressed as $[C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6}, C_{i,7}, C_{i,8}]$.
- Game $\mathcal{G}_1$: The challenge ciphertext is marked as $[C_{i,1}, R, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6}, C_{i,7}, C_{i,8}]$.
- Game $\mathcal{G}_2$: The challenge ciphertext is represented as $[C_{i,1}, R, C_{i,3}, R', C_{i,5}, C_{i,6}, C_{i,7}, C_{i,8}]$.

The following lemmas are indicated to be all computationally indistinguishable for the transitions from $\mathcal{G}_0$ to $\mathcal{G}_1$ and $\mathcal{G}_1$ to $\mathcal{G}_2$.

Lemma 2 (Keyword privacy-I). *Provided that DLIN assumption holds, then our ESPD-II construction has keyword privacy.*

Proof 2. If there is an adversary $\mathcal{A}$ who can distinguish between $\mathcal{G}_0$ and $\mathcal{G}_1$, then another algorithm $\mathcal{B}$ that can be constructed with a non-negligible advantage ϵ solves the DLIN issue as follows. On input $(g, g^{x_1}, g^{x_2}, g^{x_1 x_3}, g^{x_2 x_4}, \mathcal{R})$, the $\mathcal{B}$'s aiming is to discern $\mathcal{R} = g^{x_3 + x_4}$ or $\mathcal{A}$ is a random element.

- *Init*: $\mathcal{A}$ gives $\mathcal{B}$ the challenge set S_k and the challenge keyword w^*.
- *Setup*: In order to produce the public parameter, $\mathcal{B}$ sets $\alpha = x_2, \beta = x_1$ implicitly. Then, $\mathcal{B}$ picks $t_3, t_4, \tau, \varphi, \widetilde{v}, y, d \in \mathbb{Z}_p$ randomly and computes the public parameter $pk = (g, g_1, \ldots, g_n, g_{n+1}, \ldots, g_{2n}, T_1, T_2, T_3, T_4, \Phi, u, v, h)$.

$$g_i = g^{\tau^i}, T_1 = g^{x_2}, T_2 = g^{x_1}, T_3 = g^{t_3}, T_4 = g^{t_4},$$
$$v = g^{\widetilde{v}}, u = g^{x_2 d}, h = g^{-x_2 d w^*} g^y, \Phi = g^{\varphi}.$$

- *Phases 1 & 2*: To reply the trapdoor queries for (i, w), where $w \neq w^*$ and $i \notin S_k$, $\mathcal{B}$ picks $\widetilde{z}_i, \widetilde{z}'_i, \sigma, \sigma_1, \sigma_2 \in \mathbb{Z}_p$ at random, such that $\sigma = \sigma_1 + \sigma_2$, then sets $z_i = \frac{z_i d(w - w^*)}{d(w - w^*)x_2 + y}$ and $z'_i = \widetilde{z}'_i + \frac{\widetilde{z}_i y x_1}{t_3 t_4 (d(w - w^*)x_2 + y)}$. Next, $\mathcal{B}$ computes the trapdoor as follows.

$$tr_1 = g_i^{\varphi\sigma} \cdot (g^{x_1 \widetilde{z}_i} g^{t_3 t_4 \widetilde{z}'_i})^{\widetilde{v}\sigma} = g_i^{\varphi\sigma} \cdot v^{(x_1 x_2 z_i + t_3 t_4 z'_i)\sigma},$$
$$tr_2 = g^{x_1 \widetilde{z}_i \sigma} g^{t_3 t_4 \widetilde{z}'_i \sigma} = g^{(x_1 x_2 z_i + t_3 t_4 z'_i)\sigma},$$
$$tr_3 = (g^{x_1})^{-d(w - w^*)\widetilde{z}_i \sigma} = ((u^w h)^{z_i}))^{-x_1 \sigma},$$
$$tr_4 = (g^{x_2})^{-d(w - w^*)\widetilde{z}_i \sigma} = ((u^w h)^{z_i}))^{-x_2 \sigma},$$
$$tr_5 = (g^{x_1})^{\frac{-y\sigma\widetilde{z}_i}{t_3}} (u^w h)^{-\widetilde{z}'_i t_4 \sigma}$$
$$= (g^{-t_4 \sigma})^{d(w - w^*)x_2 + y} = ((u^w h)^{z'_i}))^{-t_4 \sigma},$$
$$tr_6 = (g^{x_1})^{\frac{-y\sigma\widetilde{z}_i}{t_4}} (u^w h)^{-\widetilde{z}'_i t_3 \sigma}$$
$$= (g^{-t_3 \sigma})^{d(w - w^*)x_2 + y} = ((u^w h)^{z'_i}))^{-t_3 \sigma}.$$

After that, $\mathcal{B}$ gives $(tr_1, \ldots, tr_6)$, σ_2 to adversary $\mathcal{A}$ and σ_1 to the aided server $\mathcal{S}_a$.

- *Challenge*: To reply encryption query for (S_k, w^*), $\mathcal{B}$ picks $s, t' \in \mathbb{Z}_p$ and produces the challenge ciphertext as follows.

$$C_{k,1} = v^{-s}(u^{w^*}h)^r = v^{-s}\mathcal{R}^y, C_{k,2} = T_1^{r-t} = Y,$$
$$C_{k,3} = T_2^t = g^{x_1x_3}, C_{k,4} = (g^r)^{t_3}g^{-t't_3} = \mathcal{R}^{t_3}g^{-t't_3},$$
$$C_{k,5} = g^{t't_4} = T_4^{t'}, C_{k,6} = g^s,$$
$$C_{k,7} = \left(\Phi \prod_{j\in S_k} g_{n+1-j}\right)^s, C_{k,8} = e(g_1, g_n)^s.$$

If $Y = g^{x_2(r-x_3)}$, $\mathcal{R} = g^{x_3+x_4}$, then $C_{k,2} = T_1^{r-t}$ and $C_{k,3} = T_2^t$.

- *Guess*: A guess $r' \in \{0, 1\}$ is outputted by $\mathcal{A}$ to distinguish which hybrid game the challenger $\mathcal{B}$ has been playing. To summarize, $\mathcal{B}$ replies r' as his/her answer in DLIN game. If the instances of DLIN are well-formed, $r' = 0$ is outputted to indicate that $\mathcal{R}$ is the random value of $\mathcal{G}_1$; otherwise, outputs $r' = 1$ to show that $\mathcal{R} = g^{x_3+x_4}$.
- *Restriction*: Due to that the set S_0 does not contain the index i, the function f should be independent of g_i. From the above proof process, we can see that in our proof the aided server $\mathcal{S}_a$ doesn't avail of g_i that is queried in the phase of trapdoor generation to compute $f(r_1, C_k)$.
- *Analysis*: From the above simulation, it is easy to see that the produced challenge ciphertext is independent of w^*, so the best success probability of the adversary $\mathcal{A}$ is $1/2$ to get $\mathcal{G}_1$ as the challenge ciphertext. In other words, the best success probability of the adversary $\mathcal{A}$ to get $\mathcal{G}_0$ as the challenge ciphertext is $1/2 + \epsilon$. So, the DLIN assumption is breached with the non-negligible probability $|Pro[\mathcal{A}(\mathcal{G}_0) = 1] - Pro[\mathcal{A}(\mathcal{G}_1) = 1]| = 1/2 + \epsilon - 1/2 = \epsilon$. □

Lemma 3 (Keyword privacy-II). *Under the decision linear (DLIN) assumption, no adversary $\mathcal{A}$ can distinguish the games $\mathcal{G}_1$ and $\mathcal{G}_2$ with advantage greater than ϵ.*

Proof 3. If there is an $\mathcal{A}$ that can easily discern between the games $\mathcal{G}_1$ and $\mathcal{G}_2$, then another algorithm $\mathcal{B}$ can be easily constructed with a non-negligible advantage ϵ to win the DLIN game below. On input $(g, g^{x_1}, g^{x_2}, g^{x_1x_3}, g^{x_2x_4}, \mathcal{R})$, the $\mathcal{B}$'s motivation is to ascertain $\mathcal{R} = g^{x_3+x_4}$ or $\mathcal{R}$ is a random.

- *Init*: A challenge set S_k and a challenge keyword w^* are picked and subsequently transmitted to $\mathcal{B}$.
- *Setup*: In order to produce the public parameter, $\mathcal{B}$ sets $\theta = x_2, \gamma = x_1$ implicitly. Then, $\mathcal{B}$ picks $t_3, t_4, \tau, \varphi, \widetilde{v}, y, d \in \mathbb{Z}_p$ randomly and computes the public parameter $pk = (g, g_1, \ldots, g_n, g_{n+1}, \ldots, g_{2n}, T_1, T_2, T_3, T_4, \Phi, u, v, h)$.

$$g_i = g^{\tau^i}, T_1 = g^{t_3}, T_2 = g^{t_4}, T_3 = g^{x_2}, T_4 = g^{x_1},$$
$$v = g^{\widetilde{v}}, u = g^{x_1 d}, h = g^{-x_1 dw^*}g^y, \Phi = g^{\varphi}.$$

- *Phases 1 & 2*: To reply the trapdoor queries for (i, w), where $w \neq w^*$ and $i \notin S_k$, $\mathcal{B}$ picks $\widetilde{z}_i, \widetilde{z}'_i, \sigma, \sigma_1, \sigma_2$ randomly, such that $\sigma = \sigma_1 + \sigma_2$, then sets $z_i = \widetilde{z}_i + \frac{\widetilde{z}'_i yx_2}{t_3t_4(d(w-w^*)x_1+y)}$ and $z'_i = \frac{\widetilde{z}'_i d(w-w^*)}{d(w-w^*)x_1+y}$. Next, $\mathcal{B}$ computes the trapdoor as follows.

$$tr_1 = g_i^{\varphi\sigma} \cdot (g^{t_3t_4\widetilde{z}_i}g^{x_2\widetilde{z}'_i})^{\widetilde{v}\sigma} = g_i^{\varphi\sigma} \cdot v^{(t_3t_4z_i+x_1x_2z'_i)\sigma},$$
$$tr_2 = g^{t_3t_4\sigma\widetilde{z}_i}g^{x_2\widetilde{z}'_i\sigma} = g^{(t_3t_4z_i+x_1x_2z'_i)\sigma},$$
$$tr_3 = (g^{x_2})^{\frac{-y\sigma\widetilde{z}'_i}{t_3}}(u^wh)^{-\widetilde{z}_it_4\sigma}$$
$$= (g^{-t_4\sigma})^{d(w-w^*)x_1+y} = ((u^wh)^{z_i}))^{-t_4\sigma},$$
$$tr_4 = (g^{x_2})^{\frac{-y\sigma\widetilde{z}'_i}{t_4}}(u^wh)^{-\widetilde{z}_it_3}$$
$$= (g^{-t_3\sigma})^{d(w-w^*)x_1+y} = ((u^wh)^{z_i}))^{-t_3\sigma},$$
$$tr_5 = (g^{x_1})^{-d(w-w^*)\widetilde{z}'_i\sigma} = ((u^wh)^{z'_i}))^{-x_1\sigma},$$
$$tr_6 = (g^{x_2})^{-d(w-w^*)\widetilde{z}'_i\sigma} = ((u^wh)^{z'_i}))^{-x_2\sigma}.$$

After that, $\mathcal{B}$ gives $(tr_1, \ldots, tr_6)$, σ_2 to adversary $\mathcal{A}$ and σ_1 to the aided server $\mathcal{S}_a$.

- *Challenge*: To reply encryption query for (S_k, w^*), $\mathcal{B}$ picks $s, t' \in \mathbb{Z}_p$ and produces the challenge ciphertext as follows.

$$C_{k,1} = v^{-s}(u^{w^*}h)^r = v^{-s}\mathcal{R}^y, C_{k,2} = T_1^{r-t} = \mathcal{R}^{t_3}g^{-t_3t},$$
$$C_{k,3} = T_2^t = g^{t_4t}, C_{k,4} = T_3^{r-t'} = (g^{x_2})^{r-t'} = Y,$$
$$C_{k,5} = T_4^{t'} = (g^{x_1})^{t'} = g^{x_1x_3}, C_{k,6} = g^s,$$
$$C_{k,7} = \left(\Phi \prod_{j\in S_k} g_{n+1-j}\right)^s, C_{k,8} = e(g_1, g_n)^s.$$

If $Y = g^{x_2(r-x_3)}$, $\mathcal{R} = g^{x_3+x_4}$, then $C_{k,4} = T_3^{r-t'}$ and $C_{k,5} = T_4^{t'}$.

- *Guess*: A guess $r' \in \{0, 1\}$ is given to distinguish which hybrid game the challenger $\mathcal{B}$ has been playing. To summarize, $\mathcal{B}$ replies r' as his/her answer in DLIN game. If the instance of DLIN is well-formed, $\mathcal{A}$ tells $r' = 0$ to indicate that $\mathcal{R}$ is the random value of $\mathcal{G}_1$; otherwise, outputs $r' = 1$ to show that $\mathcal{R} = g^{x_3+x_4}$.
- *Restriction*: The restriction in this proof is the same as that in the *proof* 2.
- *Analysis*: From the above simulation, it is easy to observe that the produced challenge ciphertext is also independent of w^*, so the best $\mathcal{A}$'s success probability is $1/2$ to get $\mathcal{G}_2$ as the challenge ciphertext. In other words, the probability of $\mathcal{A}$ to get $\mathcal{G}_1$ is $1/2 + \epsilon$. So, the DLIN assumption is breached with the non-negligible probability $|Pro[\mathcal{A}(\mathcal{G}_1) = 1] - Pro[\mathcal{A}(\mathcal{G}_2) = 1]| = 1/2 + \epsilon - 1/2 = \epsilon$. □

6.3 Proof of Trapdoor Privacy

Lemma 4 (Trapdoor privacy). *Under the n-decision Diffie Hellman Inverse assumption, then our ESPD-II construction has trapdoor privacy.*

This lemma can be proved by that the challenge keyword is indistinguishable from the same length random keyword. Two games are presented as follows: $\mathcal{G}_0$ and $\mathcal{G}_1$. In detail, in $\mathcal{G}_0$ challenger $\mathcal{B}$ picks uniformly σ_1, σ_2 while challenger $\mathcal{B}$ follows the protocol $\sigma = \sigma_1 + \sigma_2$ in $\mathcal{G}_1$.

Proof 4. The game $\mathcal{G}_0$ is shown below. Suppose that there is an $\mathcal{A}$ that can distinguish between the challenge keyword w^* from the random keyword with a non-negligible advantage ϵ, then another $\mathcal{B}$ can be simulated to address

in the n-DDHI problem. On input $(g, h_1, h_2, g_1, \ldots, g_n, g_{n+1}, \ldots, g_{2n}, g^{\varphi}, \mathcal{R})$, where $g_i = g^{\tau^i}$ for $i \in [1, 2n] \backslash \{n+1\}$, $\mathcal{B}$'s goal is to distinguish $\mathcal{R} = h_1^{\frac{\varphi}{\tau^{n+1}}}$ or $\mathcal{R}$ is a random.

- *Init*: $\mathcal{A}$ gives $\mathcal{B}$ the challenge keyword w^* and user index i^*.
- *Setup*: $\mathcal{B}$ lets $(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ be as the instance and selects $\alpha, \beta, \theta, \gamma, y, d, \widetilde{v} \in \mathbb{Z}_p$. After that, $\mathcal{B}$ computes the public parameter $pk = (g, g_1, \ldots, g_n, g_{n+1}, \ldots, g_{2n}, T_1, T_2, T_3, T_4, \Phi, u, v, h)$ below.

$$g = g, T_1 = g^{\alpha}, T_2 = g^{\beta}, T_3 = g^{\theta}, T_4 = g^{\gamma},$$
$$v = g^{\widetilde{v}}, u = g^{d}, h = g^{-dw^*} g^{y}, \Phi = g^{\widetilde{\varphi}}, g_i = g^{\tau^i}.$$

- *Query*: $\mathcal{B}$ assigns h_1 and h_2 that are in the form of $h_1 = g_{n+1+i}^{\sigma}$ and $h_2 = g^{\sigma}$, respectively. $\mathcal{B}$ picks $z_i, z_i' \in \mathbb{Z}_p$ at random and computes the trapdoor as follows.

$$tr_1 = \mathcal{R} \cdot h_2^{-(\alpha\beta z_i + \theta\gamma z_i')\widetilde{v}} = \mathcal{R} \cdot v^{-(\alpha\beta z_i + \theta\gamma z_i')\sigma},$$
$$tr_2 = h_2^{-(\alpha\beta z_i + \theta\gamma z_i')} = g^{-(\alpha\beta z_i + \theta\gamma z_i')\sigma},$$
$$tr_3 = h_2^{-y\beta z_i}, tr_4 = h_2^{-y\alpha z_i},$$
$$tr_5 = h_2^{-y\gamma z_i'}, tr_6 = h_2^{-y\theta z_i'}.$$

  After that, $\mathcal{B}$ gives $(tr_1, \ldots, tr_6)$, σ_2 to adversary $\mathcal{A}$ and σ_1 to the aided server $\mathcal{S}_a$.
- *Challenge*: To reply encryption query for (S_k, w), $\mathcal{B}$ proceeds the following steps as follows. If $w^* \neq w$ and $i \notin S_l$, $\mathcal{B}$ picks $r, s, t, t' \in \mathbb{Z}_p$ and produces the challenge ciphertext as follows.

$$C_{k,1} = v^{-s} g^{(w-w^*)dr} g^{yr} = v^{-s}(u^w h)^r, C_{k,2} = T_1^{r-t},$$
$$C_{k,3} = T_2^t, C_{k,4} = T_3^{r-t'}, C_{k,5} = T_4^{t'}, C_{k,6} = g^s,$$
$$C_{k,7} = \left(\Phi \prod_{j \in S_k} g_{n+1-j}\right)^s, C_{k,8} = e(g_1, g_n)^s.$$

  Subsequently, $\mathcal{B}$ gives $C_{k,1}, \ldots, C_{k,8}$ and S_k to adversary $\mathcal{A}$. Besides, $\mathcal{B}$ gives $(C_{k,6}, C_{k,7}, C_{k,8})$ and S_k to the aided server $\mathcal{S}_a$. If $w^* = w$ and $i \in S_l$, $\mathcal{B}$ aborts and outputs $\perp$.
- *Guess*: $r' \in \{0, 1\}$ as a guess is given to distinguish which hybrid game the challenger $\mathcal{B}$ has been playing. To summarize, $\mathcal{B}$ replies r' as his/her response in n-DDHI game. If the well-formed n-DDHI instance is produced, $\mathcal{A}$ outputs $r' = 0$ to indicate that $\mathcal{R}$ is the random keyword; otherwise, outputs $r' = 1$ to show that $\mathcal{R}$ is the challenge keyword.
- *Analysis*: From the above simulation, it is not hard to view that the keyword ciphertext formed by the challenge index is not given to the aided server $\mathcal{S}_a$. Hence, the challenge trapdoor is not compatible with any keyword ciphertext when $\mathcal{S}_a$ performs the function f of σ_1. This means that the challenge trapdoor is irrelevant to w^*, thus the best $\mathcal{A}$'s success probability is $1/2$ when $\mathcal{A}$ outputs $r' = 0$. In other words, the best $\mathcal{A}$'s success probability to return $r = 1$ is $1/2 + \epsilon$. So, the n-DDHI assumption is breached with the non-negligible probability $|Pro[\mathcal{A}(r' = 0) = 1] - Pro[\mathcal{A}(r = 1) = 1]| = 1/2 + \epsilon - 1/2 = \epsilon$. Therefore, $|Pro[\mathcal{G}_0^{\mathcal{A}}]| \leq \epsilon$.

  In game $\mathcal{G}_1$, $\mathcal{B}$ picks σ_1, σ_2, such that $\sigma = \sigma_1 + \sigma_2$ and gives σ_1 to $\mathcal{S}_a$ and σ_2 to adversary $\mathcal{A}$. In this game, $\mathcal{A}$ is not allowed to get the challenge ciphertext for $i^* \in S_k$ and $w = w^*$. We argue that due to that $\mathcal{A}$ conducts the function f of σ_1 and $C_k' = (C_{k,6}, C_{k,7}, C_{k,8})$ and is incapable of capturing any important information about σ_1. The function f of σ_1 and C_k' is completely random to $\mathcal{A}$ since $\mathbb{B}$ uniformly picks r, s, t, t' from $\mathbb{Z}_p$. On the whole, the randomized ciphertext sent to $\mathcal{A}$ is semantically secure. Let ω be the $\mathcal{A}$'s advantage that wins the semantic security encryption, then we say that the n-DDHI assumption can be broken by $\mathcal{B}$ with the probability $|Pro[\mathcal{A}(r' = 0) = 1] - Pro[\mathcal{A}(r = 1) = 1]| = 1/2 + \epsilon - (1/2 + \omega) = \epsilon - \omega$. Then, $|Pro[\mathcal{G}_1^{\mathcal{A}}]| = \epsilon - \omega$. Consequently, $|Pro[\mathcal{G}_0^{\mathcal{A}}]| - |Pro[\mathcal{G}_1^{\mathcal{A}}]| \leq \epsilon - (\epsilon - \omega) = \omega$.

In fact, since two servers are utilized in our ESPD-II construction, then the security definition of trapdoor privacy is slightly modified compared to the security definition of keyword privacy. In detail, the adversary can make encryption queries, whereas he/she cannot upload any ciphertext that he/she desires since the aided server is honest and controlled by $\mathcal{B}$. □

Remark 3. For our security proofs, the security proof of our file privacy is almost the same as that in [16]. The security proofs of keyword privacy are obtained by applying Boyens anonymous identity-based system methodology [44] and Cuis linear splitting technique [33] via a hybrid argument over a sequence of games. The security proof of our trapdoor privacy mainly originates from [33], [44]. In our security proof of keyword privacy, although the adversary can simply create a trapdoor for any keyword w, he/she can still learn nothing from keyword ciphertext unless he/she can produce the correct trapdoor as the challenger produces for him/her. This is because the server that receives a self-produced trapdoor from the adversary cannot tell which ciphertext encrypts which keyword without the trapdoors for the access control list satisfied by the keyword associated with the ciphertexts. As well, in our security proof of trapdoor privacy, the adversary cannot discern that a picked keyword or a random keyword is encrypted in a trapdoor. This is because the adversary cannot drive the legitimate secret keys of other users to produce the trapdoors. To prove the security of keyword privacy and trapdoor privacy, the corresponding hard problems are embedded into the keyword ciphertext part and trapdoor part. If the adversary can always win the game, it implies that the decisional hard problem can be always solved. In our security proofs, its easy to learn that the adversary does not always win the game under he/she always holds legitimate trapdoors.

7 Performance Evaluation

This part shows the theoretical and experimental analysis to indicate our efficiency in terms of related "one-to-many"

TABLE 1
Functionality Comparisons in One-to-Many Data Sharing Schemes

Schemes	$\mathbf{P}_0$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_5$	$\mathbf{P}_6$	$\mathbf{P}_7$	$\mathbf{P}_8$	$\mathbf{P}_9$
ZXA [28]	✓	×*	×*	×	×	×*	×	×	×	×*
LW [29]	✓	✓	×	×	×	✓	✓	✓	×	×
CWR+ [33]	✓	×*	×*	×	×	×*	✓	✓	×	×
MML+ [30]	✓	✓	×	×	×	✓	×	×	✓	✓
HGW+ [32]	✓	×*	×	×	×	×	×	×	✓	×*
HYL [34]	✓	×*	×	×	✓	×*	✓	✓	✓	×*
MLL+ [31]	✓	✓	×	×	×	✓	×	×	✓	✓
SYL+ [35]	✓	×*	×	×	×	×*	✓	✓	✓	×*
MLC+ [36]	✓	×*	×	×	×	×*	×	×	✓	×*
AOR+ [13]-I	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
AOR+ [13]-II	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
Ours-I	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
Ours-II	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note: *$\mathbf{P}_0$: Keyword search; $\mathbf{P}_1$: Encrypted data sharing; $\mathbf{P}_2$: Constant-size secret key; $\mathbf{P}_3$: Constant-size search token (trapdoor); $\mathbf{P}_4$: Constant-size ciphertext; $\mathbf{P}_5$: File privacy; $\mathbf{P}_6$: Keyword privacy; $\mathbf{P}_7$: Trapdoor privacy; $\mathbf{P}_8$: No trusted third party for generating search token; $\mathbf{P}_9$: Lightweight decryption.*

data sharing works. The experiments is conducted to evaluate the performance of ESPD-I and ESPD-II. The configuration is as follows: All the experiments are compiled in the JAVA language. The "Cloud" is simulated with one Lenovo server which has 512SSD, 1TB mechanical hard disk and runs on the Windows 10 operating system with Intel(R) 8 Core(TM) i7-7820HK CPU @2.9 GHz and 16GB RAM. Each user is replaced by a Huawei nova3 android phone equipped with 6GB RAM, four-core 2.36GHz Cortex A73 processor and four-core Cortex A53 1.8GHz processor. All the raw data are selected from Enron Email Dataset[1], in which half a million records from 15 users are and has been used for the performance evaluation of data sharing systems. All of experimental simulations are depended on the average time of 100 times.

7.1 Functionality

In Table 1, we compare that whether the following functionalities can be achieved in one-to-many keyword search schemes, such as keyword search, encrypted data sharing, constant-size secret key, constant-size search token, constant-size ciphertext, file privacy, keyword privacy, trapdoor privacy, no trusted third party for generating search token and lightweight decryption. Here, "✓" means the specified function can be supported. "×*" means the scheme has not provided this functionality, and "×" indicates the scheme has this function but does not achieve the specified functionality.

As shown in Table 1, we can see that Zheng *et al.*'s [28] scheme can provide keyword search function instead of supporting encrypted data sharing. The sizes of a secret key, search token scale linearly with the amount of attributes and the ciphertext size is incremental with the quantity of access policy's attributes. Neither Keyword privacy nor trapdoor privacy can be protected due to the keyword guessing attacks on keyword ciphertext and trapdoor. The scheme in [29] supports both keyword search and encrypted data sharing, and can provide file privacy, keyword privacy and trapdoor privacy. However, the storage costs of the secret key, search token and ciphertext are linearly incremental with the amount of ascribed attributes. The relationship between decryption computation cost and the amount of user attributes is linear. Cui *et al.*'s scheme [33] supports keyword search with keyword privacy and trapdoor privacy, nevertheless, it does not provide encrypted data sharing function. Although Miao *et al.*'s scheme [30] provides keyword search and encrypted data sharing with lightweight decryption, which however cannot protect keyword privacy as well as trapdoor privacy. Besides, the sizes of the secret key, trapdoor and ciphertext are not constant. The scheme in [32] also realizes keyword search but it fails to guarantee keyword privacy and trapdoor privacy. Besides, the encrypted data sharing function is not rendered. In Han *et al.*'s work [34], it cannot achieve constant-size ciphertext, keyword privacy and trapdoor privacy but also does not realize the encrypted data sharing. While Miao *et al.*'s scheme [31] provides keyword search and encrypted data sharing with lightweight decryption, the privacy protection of keyword and trapdoor cannot be considered. Additionally, the storage costs of secret key, trapdoor and ciphertext also scale linearly with the number of attributes. In Sun *et al.*'s scheme [35], the keyword privacy and trapdoor privacy are ensured but the encrypted data sharing is not provided. Miao *et al.* also proposed an ABSE scheme [36], which also fails to realize the privacy-preserving of both keyword and trapdoor. Also, encrypted data sharing is not considered. In [13], Kiayias *et al.* raised two keyword search schemes, which achieve neither keyword privacy nor trapdoor privacy they claimed. In our first scheme (ESPD-I), most functionalities are realized except for trapdoor privacy, and our enhanced scheme achieves all the desirable features. As described above, we can easily conclude that only the schemes [29], [30], [36] support both keyword search and encrypted data sharing while other schemes [13], [28], [31], [32], [33], [34], [35] only realize keyword search. The decryption computation cost is lightweight in the schemes [13], [30], [36] and ours. The sizes of the secret key, trapdoor and ciphertext are constant in [13] and ours. We can also observe that only our first scheme can achieve all table-listed properties except trapdoor privacy and only our second scheme owns all these desirable features. These nice features make our schemes feasible and practical for data sharing services.

Remark 4. In our experimental part, we mainly focus on the comparisons among related "one-to-many" data sharing and retrieving schemes, i.e., broadcast-based SE (BBSE) and attribute-based SE (ABSE), and our constructed ESPD schemes. For ABSE works, due to the fact that the computation and storage cost of ABSE is increased with the increment of the quantity of attributes, which would be frequently beyond the capabilities of users with limited resources, hence it is essential to construct an efficient one-to many data sharing and retrieving scheme. As an alternative solution, BBSE scheme can realize constant and stable computation and storage cost, regardless of the amount of system attributes. In our ESPD schemes,

1. http://www.cs.cmu.edu/ enron/

TABLE 2
Storage and Communication Cost Comparisons in One-to-Many Data Sharing Schemes

Schemes	Public key size	Secret key size	Ciphertext size	Trapdoor size
ZXA [28]	$4\|\mathbb{G}_0\|$	$2S\|\mathbb{G}_0\|$	$(2m+3)\|\mathbb{G}_0\|$	$(2S+3)\|\mathbb{G}_0\|$
LW [29]	$(U+1)\|\mathbb{G}_0\|+(U+1)\|\mathbb{G}_1\|+\|\mathbb{G}_T\|$	$3S\|\mathbb{G}_0\|$	$(m+5)\|\mathbb{G}_0\|$	$3S\|\mathbb{G}_0\|$
CWR+ [33]	$8\|\mathbb{G}_0\|+\|\mathbb{G}_T\|$	$\perp$	$(5m+1)\|\mathbb{G}_0\|$	$6S\|\mathbb{G}_0\|$
MML+ [30]	$5\|\mathbb{G}_0\|+\|\mathbb{G}_T\|$	$(2S+3)\|\mathbb{G}_0\|$	$(2m+2)\|\mathbb{G}_0\|$	$(2S+3)\|\mathbb{G}_0\|$
HGW+ [32]	$5\|\mathbb{G}_0\|$	$(2S+1)\|\mathbb{G}_0\|$	$(2m+3)\|\mathbb{G}_0\|$	$(2S+3)\|\mathbb{G}_0\|$
HYL [34]	$(U+1)\|\mathbb{G}_0\|+U\|\mathbb{G}_1\|$	$(S+1)\|\mathbb{G}_0\|$	$4\mathbb{G}_0\|$	$2\|\mathbb{G}_0\|$
MLL+ [31]	$4\|\mathbb{G}_0\|$	$(2S+4)\|\mathbb{G}_0\|+2\|\mathbb{Z}_p\|$	$(2m+2)\|\mathbb{G}_0\|+(2m+1)\|\mathbb{Z}_p\|$	$(2S+3)\|\mathbb{G}_0\|+\|\mathbb{Z}_p\|$
SYL+ [35]	$(3U+1)\|\mathbb{G}_0\|+\|\mathbb{G}_T\|$	$(2S+1)\|\mathbb{G}_0\|+\|\mathbb{Z}_p\|$	$(2m+1)\|\mathbb{G}_0\|+\|\mathbb{G}_T\|$	$(2S+1)\|\mathbb{G}_0\|+\|\mathbb{Z}_p\|$
MLC+ [36]	$(U+1)\|\mathbb{G}_0\|+\|\mathbb{G}_T\|$	$(2S+5)\|\mathbb{G}_0\|+(S+3)\|\mathbb{Z}_p\|$	$(m+6)\|\mathbb{G}_0\|+(2m+1)\|\mathbb{Z}_p\|$	$(2S+1)\|\mathbb{G}_0\|+\|\mathbb{Z}_p\|$
AOR+ [13]-I	$(2n+10)\|\mathbb{G}_0\|$	$14\|\mathbb{G}_0\|$	$8\|\mathbb{G}_0\|+2\|\mathbb{G}_T\|$	$5\|\mathbb{G}_0\|$
AOR+ [13]-II	$(2n+10)\|\mathbb{G}_0\|$	$14\|\mathbb{G}_0\|$	$8\|\mathbb{G}_0\|+2\|\mathbb{G}_T\|$	$6\|\mathbb{G}_0\|$
Ours-I and Ours-II	$(2n+9)\|\mathbb{G}_0\|$	$10\|\mathbb{G}_0\|$	$9\|\mathbb{G}_0\|+2\|\mathbb{G}_T\|$	$6\|\mathbb{G}_0\|$

we realize two "one-to-many" data sharing and retrieving schemes with lower computation and computation cost by using BBSE technology.

7.2 Storage and Communication Cost

Now we discuss the storage and communication cost of our proposed ESPD schemes. Here, the storage and communication cost refers to the space for storing the output result of each cryptographic algorithm, such as the storage and communication cost of Setup algorithm the space to store the public key produced by Setup algorithm. Besides, the state-of-the-art works [13], [28], [29], [30], [31], [32], [33], [34], [35], [36] are also analyzed in this part. These works are very similar to ours since they are one-to-many keyword search schemes. In our following experiment part, only the works [13], [29], [30], [36] and ours are simulated, as they are committed to designing efficient data sharing service with retrieval function in the ciphertext environment. In Table 2, we compare the storage and communication cost in terms of the sizes of the public key, secret key, ciphertext and trapdoor. U, S, m and n denote the amount of system attributes, user attributes, attributes of access policy and users. Besides, an element length in $\mathbb{G}_0$, an element length in $\mathbb{G}_1$ and $\mathbb{G}_T$ are also correspondingly represented as $|\mathbb{G}_0|$, $|\mathbb{G}_1|$ and $|\mathbb{G}_T|$.

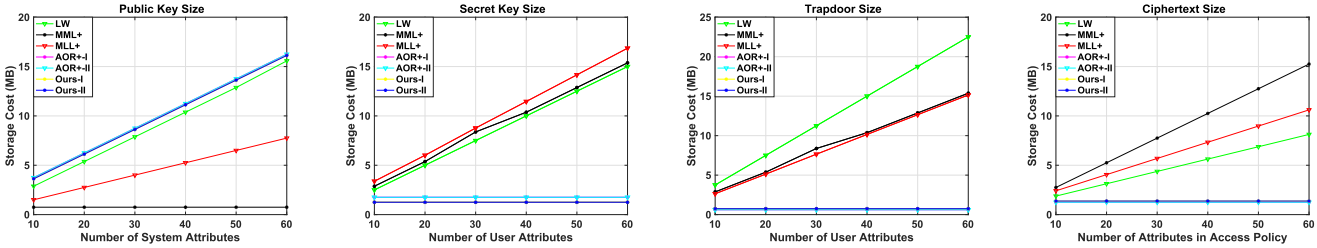
7.2.1 Theoretical Analysis

As showed in Table 2, the storage and communication costs of the public key, secret key, ciphertext and trapdoor are presented. Specifically, in the phase of setup, only the public key size in works [28], [30], [32], [33], [36] are constant while that of the others [13], [29], [31], [34], [35] and ours is growing linearly with the number of either system attributes or system users. In the key generation phase, only our works and Kiayias *et al.*'s works [13] have constant secret key size irrespective of the number of users or attributes while the storage cost of the secret key in other works is increasing with the quantity of user attributes. Formally speaking, constant-size secret key results in smaller decryption computation cost, which is desirable for resource-limited devices. Besides, as the work [33] has no encrypted data sharing function, it does not require to distribute a secret key to users. In the phase of ciphertext generation, only the ciphertext size in [13], [34] and ours are constant regardless of the number of attributes or identities in access control. Conversely, the ciphertext size in the rest of other works is incremental linearly with the number of attributes or identities in access control. As we all know, smaller ciphertext size means smaller storage costs for users when they download them to decrypt, which is much appropriate for resource-constrained users. In the trapdoor generation phase, except for the works [13], [34], only our works can achieve constant-size trapdoor while the trapdoor size in others is also incremental linearly the amount of user attributes. From Table 2, we can easily observe that our secret key size is the smallest in these one-to-many data sharing schemes, and our trapdoor size is slightly larger only than that of the ESPD-I scheme in [13].

7.2.2 Experimental Results

The experiment is performed to further demonstrate the communication overhead of our ESPD constructions and other schemes with encrypted data sharing service. According to above analysis, we can learn that the communication complexity of our ESPD is mainly related to the number of users while the others [13], [29], [30], [36] are associated with the number of attributes.

As shown in Fig. 6, we give the detailed storage and communication cost comparisons of the public key, secret key, trapdoor and ciphertext among distinct one-to-many data sharing schemes [13], [29], [30], [36] and ours. Specifically, Fig. 6a shows the storage and communication cost comparisons of public key. In this simulation, we always assume the number of users is two times the number of system attributes. From Fig. 6a, we can see that only the work [30] owns a constant-size public key while the others' public key sizes [13], [29], [36] are growing with the number of system attributes. Fig. 6b depicts the storage and communication cost comparisons of the secret key. From this figure, we can straightforwardly observe that the storage and communication cost of a secret key in works [29], [36], [36] follows the linear relationship with the number of user attributes while our works and Kiayias's works have constant secret key sizes. Further, our works show relatively desirable performance in the storage cost of a secret key compared to the works in [13]. Fig. 6c indicates the storage and communication cost comparisons of



(a) Storage cost of public key (b) Storage cost of secret key (c) Storage cost of trapdoor (d) Storage cost of ciphertext

Fig. 6. Storage costs of public key, secret key, trapdoor and ciphertext in distinct one-to-many data sharing schemes.

ciphertext. As shown in Fig. 6c, we can easily conclude that the storage and communication cost of ciphertext in works [29], [36], [36] is increasing linearly with the number of attributes in access policy while our works and Kiayias's works always have constant ciphertext sizes without requiring to consider the amount of attributes in policy. Also, we can find that our works have a slightly higher storage cost than the works [13]. Fig. 6d describes the storage and communication cost comparisons of trapdoor. As illustrated in Fig. 6d, we can easily obtain that the works [13] and our works are constant for the storage overhead of the trapdoor although the storage overhead of the trapdoor in other three works is growing linearly with the number of attributes.

In summary, from Fig. 6, our works almost have desirable performance in secret key, trapdoor and ciphertext sizes, which indicates their feasibility and practicability for real-world scenarios.

7.3 Computation Overhead

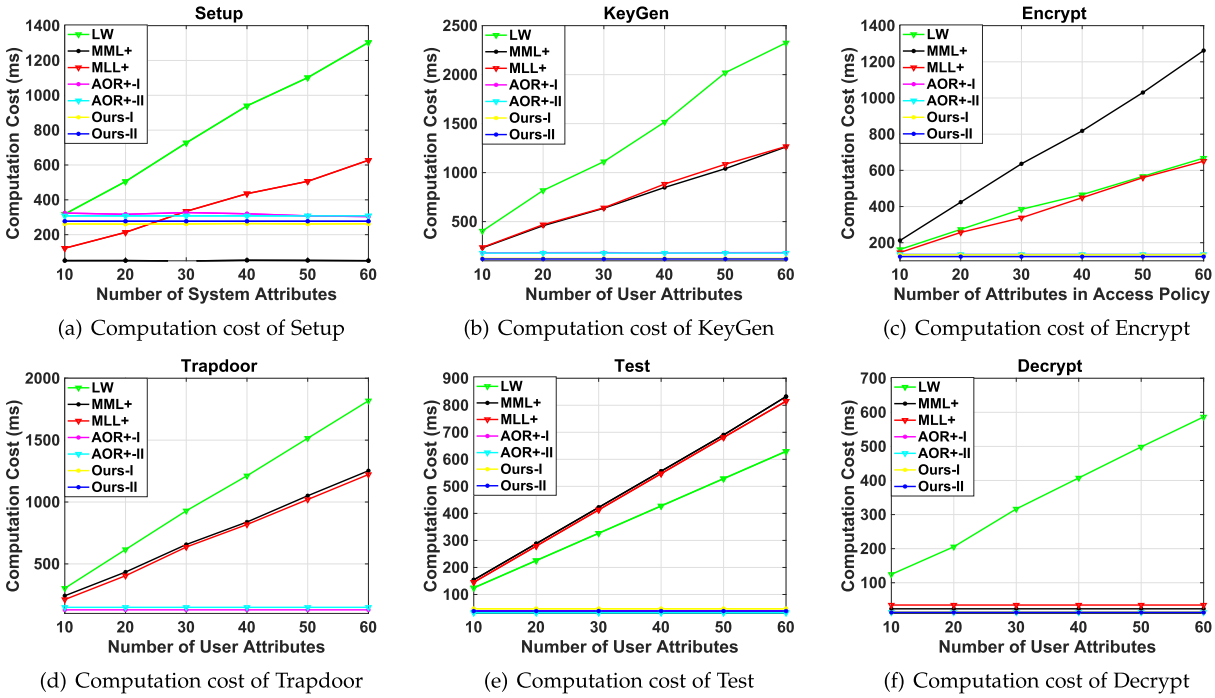
The computational overhead of our ESPD schemes is discussed in this part. Here, computation cost means the execution time of the cryptographic algorithms, such as the computation cost of Setup algorithm refers to the time required to execute this algorithm. Similarly, the state-of-the-art works [13], [28], [29], [30], [31], [32], [33], [34], [35], [36] are also analyzed to show more comparability in the part of theoretical analysis. Besides, we also conduct the experimental simulations of the works [13], [29], [30], [36] and our ESPD, as these works completely have the same functions and are utilized for one-to-many data sharing services. In Table 3, we make comparisons of the computation cost for different algorithms of similar schemes in terms of Setup, KeyGen, Encrypt, Trapdoor, Test and Decryption algorithms. An exponentiation computation in $\mathbb{G}_0$ and $\mathbb{G}_T$ as well as a pairing computation are correspondingly expressed as e_0, e_1 and p.

7.3.1 Theoretical Analysis

Table 3 depicts that the computation overheads of Setup, KeyGen, Encrypt, Trapdoor, Test and Decrypt phases are presented. In detail, in the phase of Setup, only the works [28], [31], [33] have constant computation overheads while the setup computation costs in other works [13], [29], [30], [32], [34], [35], [36] are incremental linearly with the quantity of attributes or users. In the KeyGen phase, only the works [13] and our ESPD have constant computation costs with no need to consider the number of attributes or users, while the relationship between key generation of the remained works and the number of user attributes follows a linear growth. In the phase of Encrypt, only the works [13], [34] and ours support the constant computation overhead while the computation cost of the others has a linear growth relationship with the amount of attributes. In the phase of Trapdoor, the computation costs in these works [29], [30], [32], [34], [35], [36] increase as the the number of user attributes scales, while ours and Kiayias *et al.*'s works own the constant computation costs. In the Test phase, only the works [13], [34] and ours realize the constant calculation overhead, while the computation cost in others is also growing linearly with the number of attributes in access control. In the Decrypt phase, the calculation cost of Liang *et al.*'s work [29] has a linear relationship with the number of user

TABLE 3
Computation Cost Comparisons for Various Algorithms in One-to-Many Data Sharing Schemes

Schemes	Setup	KeyGen	Encrypt	Trapdoor	Test	Decrypt
ZXA [28]	$3e_0$	$(2S+1)e_0$	$(2m+4)e_0$	$(2S+3)e_0$	$(2m+3)p$	$\perp$
LW [29]	$(2U+10)e_0+3p$	$4Se_0$	$(m+5)e_0+e_1+p$	$3Se_0$	$(m+1)e_0+2p$	$(m+1)e_0+2p$
CWR+ [33]	$4e_0$	$\perp$	$(6m+2)e_0+e_1$	$14Se_0+e_1+p$	$(6m+1)e_1+(6m+1)p$	$\perp$
MML+ [30]	$4e_0+e_1+p$	$(2S+3)e_0$	$(2m+2)e_0$	$(2S+4)e_0$	$(2m+3)p$	$2p+e_1$
HGW+ [32]	$4e_0$	$(2S+2)e_0$	$(5m+3)e_0$	$(2S+3)e_0$	$(2m+1)p$	$\perp$
HYL [34]	$U\cdot(e_0+e_1+p)$	$(2S+1)e_0$	$3e_0+p$	$(2S+1)e_0$	$3p$	$\perp$
MLL+ [31]	$3e_0$	$(2S+2)e_0$	$(2m+4)e_0+e_1$	$(2S+3)e_0$	$e_1+(2m+4)p$	$\perp$
SYL+ [35]	$3Ue_0+e_1$	$(2S+1)e_0+e_1$	$(m+1)e_0+e_1$	$(2S+1)e_0$	$e_1+(m+1)p$	$\perp$
MLC+ [36]	$(U+1)e_0+e_1+p$	$(2S+5)e_0+e_1$	$(m+4)e_0+e_1$	$(2S+1)e_0$	$e_1+(2m+1)p$	e_0+e_1+3p
AOR+ [13]-I	$(2n+10)e_0$	$18e_0$	$12e_0+2e_1+2p$	$13e_0$	$6p$	$2p$
AOR+ [13]-II	$(2n+10)e_0$	$18e_0$	$12e_0+2e_1+2p$	$15e_0$	$7p+e_1$	$2p$
Ours-I	$(2n+9)e_0$	$12e_0$	$12e_0+2e_1+2p$	$4e_0$	$7p$	$2p$
Ours-II	$(2n+9)e_0$	$12e_0$	$12e_0+2e_1+2p$	$10e_0$	$8p+e_1$	$2p$



(a) Computation cost of Setup (b) Computation cost of KeyGen (c) Computation cost of Encrypt

(d) Computation cost of Trapdoor (e) Computation cost of Test (f) Computation cost of Decrypt

Fig. 7. Computation costs of Setup, KeyGen, Encrypt, Trapdoor, Test and Decrypt algorithms in distinct one-to-many data sharing schemes.

attributes and that of the works [13], [30], [36] and ours has constant decryption computation cost. From Table 3, it is not hard to get that only our works and the works [13] have constant computation cost in the phases of KeyGen, Encrypt, Trapdoor, Test and Decrypt. As we all know, smaller computation cost makes resource-constrained users also capable of obtaining related services. Compared to [13], the computation cost of the KeyGen phase in ours is relatively lower. The computation of Encrypt and Decrypt phases in ours are the same as that of [13].

7.3.2 Experimental Analysis

We utilize the version of Intellij IDEA-2018.2.5, Java 8 and install the latest JPBC library [18] for underlying cryptographic operations. All the experiments are simulated over a supersingular elliptic curve with the bilinear map pairing on it. This curve is denoted as $E(F_q) : y^2 = x^3 + x$. Then, we set $|\mathbb{Z}_p| = 160$ bits and $|\mathbb{G}_0| = |\mathbb{G}_T| = 1024$ bits.

As illustrated in Fig. 6, we give the computation cost comparisons of Setup, KeyGen, Encrypt, Trapdoor, Test and Decrypt among the works [13], [29], [30], [36] and ours. Specifically, Fig. 7a presents the computation cost comparisons in the Setup phase of different works. From Fig. 7a, we can see that only the work [30] owns a constant computation cost in Setup while that of other works [13], [29], [36] is linearly incremental with the amount of attributes. Fig. 7b depicts the computation cost comparisons in the KeyGen phase of various works. From this figure, we can observe that the relationship between the computation cost for secret key generation in works [29], [36] and the number of user attributes is incrementally linear while our works and Kiayias's works have constant computation cost for secret key generation. Further, our works show relatively better performance in the computation cost of KeyGen compared to the works in [13]. Fig. 7c indicates the computation cost comparisons in the Encrypt phase of distinct works. As shown in Fig. 7c, it is easy to conclude that the computation cost of ciphertext generation in works [29], [36], [36] is linearly proportional to the number of attributes in an access policy while our works and Kiayias's works have constant computation cost regardless of the number of attributes in access policy. Also, we can find that the computation costs in these works than our works is almost the same as the works [13] in the computation cost of Encrypt. Fig. 7d presents the computation cost comparisons of trapdoor generation. As illustrated in Fig. 7d, we are apt to get that the computation cost of the works [13] and our works are constant in trapdoor generation while that in other three works is growing linearly with the number of attributes. Fig. 7e presents the computation cost comparisons of Trapdoor. As illustrated in Fig. 7d, it's really easy to summarize that the computational cost of the works [13] and our works are constant in performing keyword search while that in the remained works is increasing linearly with the number of attributes. As illustrated in Fig. 7f, we can also learn that the decryption computation cost of the works [13], [30], [36] and our works are constant while the decryption computation cost of Liang *et al.*'s work is incrementally linear with the number of attributes. To summarize, from Fig. 6, we can conclude that our works have lower computation costs in Setup, KeyGen, Encrypt, Trapdoor, Test and Decrypt phases.

According to the above theoretical and experimental analysis, we can conclude that our works are almost outperformed in communication and computation cost than the other works, which make our ESPD schemes practical and appropriate in real-world applications.

8 Related Works

This part illustrates the related works of privacy-preserving data sharing over the outsourced data. In summary, existing

works can be evolved from three underlying technologies, i.e., symmetric searchable encryption (SSE), attribute-based searchable encryption (ABSE), broadcast-based searchable encryption (BBSE) and their hybrid approaches. We review the related works of them respectively.

8.1 SSE-Based Data Sharing

Song *et al.* [14] first put forward the primitive of symmetric searchable encryption (SSE), which allows a cloud server to retrieve directly over the encrypted data. After that, various SSE works have been proposed with varying degrees of tradeoffs between security [19], [20], [21], efficiency [22], [23], [24] and functionality [25], [26], [27]. For instance, Fisch *et al.* [20] proposed a scalable SSE scheme, which solves the semi-honest security issue. Bost *et al.* [19] gave the concept of backward security for dynamic SSE and raised two backward-secure schemes. However, the cost scales with the number of entries in the database. Cash *et al.* [22] designed a dynamic SSE scheme, which supports a user in efficiently and privately searching server-held encrypted databases. To improve locality efficiency and handle a dynamic message, Miers *et al.* [24] proposed a scaling dynamic SSE scheme to millions of indexes by improving locality. Cash *et al.* [25] introduced a highly-scaling SSE scheme with support for boolean queries. Chase *et al.* [26] proposed a structured and efficient SSE scheme, which considers the issue of encrypting structured data (e.g., a web graph or a social network). Kamara *et al.* [27] also formulated a boolean SSE with worst-case sub-linear complexity. Although SSE schemes can provide fast keyword search with various functionalities, the fly in the ointment is that existing SSE schemes are usually suitable for one-to-one (i.e., one data owner-to-one user) data sharing scenarios. This is mainly due to the inherent limitation of symmetric encryption (i.e., requires encryption and decryption operations to share the same secret key), which makes one-to-many plaintext sharing impractical once multiple users collude with each other.

8.2 ABSE-Based Data Sharing

To address the one-to-many data sharing problem while preserving data utility, one of major approaches is exploiting Attribute Based Searchable Encryption (ABSE). ABSE schemes enable a data owner to share the data with a specified group of data receivers in a fine-grained manner while users can retrieve and decrypt the target data in the case that the attributes of a data receiver satisfy the access policy. For example, Zheng *et al.* [28] first proposed the notion of ABSE and designed a ciphertext-policy ABSE scheme (CP-ABSE), however, the communication and computation costs are linear increasingly with the complexity of access tree. Besides, the CP-ABSE scheme [28] is vulnerable to keyword guessing attacks that result in keyword privacy leakage. Liang *et al.* [29] put forward a searchable attribute-based mechanism with efficient data sharing. However, it cannot enable a secret key holder to generate trapdoor (search token) individually without the support of the trusted key generation center. Further, the sizes of keyword ciphertext, trapdoor and secret key depend on the number of attributes involved in the specified control policy. To realize more flexible authorization, Miao *et al.* [30] put forward a novel ABSE scheme, which supports the shared records that have hierarchical structures instead of considering keyword privacy and trapdoor privacy. Miao *et al.* [31] also raised a practical attribute-based multi-keyword search scheme, however, which suffers from the same keyword and trapdoor privacy problem as that in [30]. He *et al.* [32] raised an attribute-based hybrid boolean keyword search over outsourced encrypted data, which supports more expressive search, such as any required boolean keyword expression search. However, it can only support keyword retrieval but does not support the encryption of plaintext information. Besides, it also suffers from high computation and communication efficiency. Cui *et al.* [33] proposed an efficient and expressive keyword retrieval scheme, which can resist the keyword guessing attacks. However, it also suffers from the same efficiency defect as that in [32]. Han *et al.* [34] presented an expressive ABSE scheme with constant-size ciphertext. Whereas, it is incapable of encrypted data sharing. Sun *et al.* [35] raised a verifiable ABSE scheme, which has the same issue as that in [34]. Recently, a novel privacy-preserving ABSE scheme [36] is proposed. However, it is incapable of realizing the claimed keyword privacy and trapdoor privacy. Although the above ABSE schemes can achieve versatile keyword search and have been applied in various applications, neither lower communication and computation cost nor the claimed keyword privacy in these ABSE schemes is achieved.

8.3 BBSE-Based Data Sharing

To achieve one-to-many keyword search with constant communication and computation cost, Kiayias *et al.* [13] first invented a broadcast-based searchable encryption (BBSE) scheme for multi-user data sharing, which allows a user with constant-size search token to perform keyword search in a subset of files that he is granted to access. Such promising and appealing properties make BBSE primitive applicable in various applications, such as task recommendation services [37] and fog-assisted Internet of Things [38]. However, it only supports file privacy and keyword privacy instead of trapdoor privacy due to that the keyword guessing attacks on the delegated trapdoor launched by the cloud server cannot be blocked. To further realize trapdoor privacy, an enhanced BBSE scheme was also proposed in [13]. Regrettably, after careful observations, it is easy to find that the first scheme in [13] fails to reach the goal of the claimed keyword privacy while the second one can achieve neither the stated keyword privacy nor the trapdoor privacy. The basic reason leading to the leakages of both keyword privacy and trapdoor privacy originates from that non-authorization users can bypass the access control, thus illegally retrieving the data of his interests.

8.4 Other Hybrid Approaches for Data Sharing

The study [39] introduced a scheme based on SSE and ABE for data sharing. In which, a data owner encrypts their files using SSE, but the resulted indexes are encrypted with ABE. In this way, users can locally generate search tokens based on their attributes, that are then sent to the cloud for retrieving. Although this hybrid encryption scheme realizes data sharing and retrieving, the ciphertext size is incremental to

the number of attributes. Bakas *et al*. [41] proposed a hybrid encryption scheme that combines both SSE and ABE in a way that the main advantages of each encryption technique are used. Specifically, the proposed scheme enables clients to search over encrypted data by using an SSE scheme, while the symmetric key required for the decryption is protected via an ABE scheme. However, in [41], SSE and ABE are used as black-boxes to realize data retrieval and sharing, which is differ significantly from our study that presents the concrete implementation details for searching and sharing. Recently, Michalas [40] also invented a hybrid encryption scheme based on both SSE and ABE schemes, which allows users to directly search over encrypted data by using an SSE scheme while the access to the decryption key is protected by utilizing an ABE scheme. However, this scheme has the same issue as that in [41]. Besides, since the scheme is the integration of SSE and ABE schemes, it inevitably inherits the disadvantages of most ABE schemes that the size of the produced ciphertexts and the time required to decrypt grows with the complexity of the access.

As a consequence, to the best of our knowledge, there is no such a lightweight and privacy-preserving data sharing service for multi-user settings, which can realize constant communication and computation cost, keyword privacy and trapdoor privacy simultaneously.

9 Conclusion

This paper investigated two efficient, scalable and privacy-preserving data sharing services referred to be as ESPD-I and ESPD-II, which empower a data owner to store a set of encrypted files in a not fully trusted server, and a user is permitted to securely and efficiently retrieve keyword in a subset of files that he/she is granted to access. Besides, this paper gave strict security proofs to indicate the file privacy, keyword privacy and trapdoor privacy of the suggested constructions, which demonstrated that our solution could lower the leakage risks of both keyword privacy and trapdoor privacy in the cloud computing. We present the detailed theoretical and experimental analysis to reveal that the proposed ESPD schemes are efficient and feasible for real-world applications.

Acknowledgments

We would like to thank the anonymous reviewers for their invaluable comments and suggestions.

References

[1] Z. Li and Y. Yang, "RRect: A novel server-centric data center network with high power efficiency and availability," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 914–927, Third Quarter 2020.

[2] H. Wang, J. Ning, X. Huang, G. Wei, G. Poh, and X. Liu, "Secure fine-grained encrypted keyword search for E-Healthcare cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1307–1319, May/Jun. 2021.

[3] S. Qiu, B. Wang, M. Li, J. Liu, and Y. Shi, "Toward practical privacy-preserving frequent itemset mining on encrypted cloud data," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 312–323, First Quarter 2020.

[4] P. Chaudhari and M. L. Das, "Privacy preserving searchable encryption with fine-grained access control," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 753–762, Second Quarter 2021.

[5] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Apr. 2019.

[6] Y. Huang, Y. Yang, X. Song, F. Ye, and X. Li, "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 852–864, Apr. 2020.

[7] P. Li, S. Guo, S. Yu, and W. Zhuang, "Cross-cloud MapReduce for big data," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 375–386, Second Quarter 2020.

[8] C. Wang, Y. Yang, and P. Zhou, "Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 394–410, Feb. 2021.

[9] B. Kalsnes and A. O. Larsson, "Understanding news sharing across social media: Detailing distribution on Facebook and Twitter," *Journalism Stud.*, 2018, vol. 19, no. 11, pp. 1669–1688, 2018.

[10] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.

[11] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2019.2910113.

[12] G. Xu, H. Li, H. Ren, X. Lin, and X. S. Shen, "DNA similarity search with access control over encrypted cloud data," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.2968893.

[13] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2016, pp. 173–195.

[14] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in*Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.

[15] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 506–522.

[16] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Proc. Annu. Int. Cryptol. Conf.*, 2005, pp. 258–275.

[17] D. Boneh, X. Boyen, and E. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2005, pp. 440–456.

[18] B. Lynn, "The stanford pairing based crypto library," Accessed: Sept. 2019. [Online]. Available: http://crypto.stanford.edu/pbc/

[19] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1465–1482.

[20] B. Fisch *et al.*, "Malicious-client security in blind seer: A scalable private DBMS," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 395–410.

[21] F. Sun *et al.*, "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 763–780.

[22] D. Cash *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 23–26.

[23] D. Cash and S. Tessaro, "The locality of searchable symmetric encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2014, pp. 351–368.

[24] I. Miers and P. Mohassel, "IO-DSSE: Scaling dynamic searchable encryption to millions of indexes by improving locality," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–23.

[25] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc. Annu. Cryptol. Conf.*, 2013, pp. 353–373.

[26] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2010, pp. 577–594.

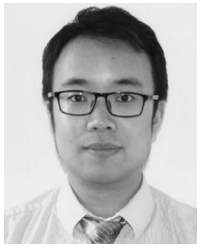
[27] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2017, pp. 94–124.

[28] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 522–530.

[29] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1981–1992, Sep. 2015.
[30] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 985–998, Nov./Dec. 2020.
[31] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things J.*, vol. 4, no. 5, pp. 3008–3018, Aug. 2018.
[32] K. He, J. Guo, J. Weng, J. Weng, J. Liu, and X. Yi, "Attribute-based hybrid Boolean keyword search over outsourced encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1207–1217, Nov./Dec. 2020.
[33] H. Cui, Z. Wan, R. H. Deng, G. Wang, and Y. Li, "Efficient and expressive keyword search over encrypted data in cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 409–422, May/Jun. 2018.
[34] J. Han, Y. Yang, and J. Liu, "Expressive attribute-based keyword search with constant-size ciphertext," *Soft Comput.*, vol. 22, no. 15, pp. 5163–5177, 2018.
[35] W. Sun, S. Yu, W. Lou, Y. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 226–234.
[36] Y. Miao *et al.*, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1080–1094, May/Jun. 2021.
[37] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 235–247, Jan./Feb. 2021.
[38] R. Zhou, X. Zhang, X. Wang, G. Yang, and H. Wang, "Privacy-preserving data search with fine-grained dynamic search right management in fog-assisted Internet of Things," *Inf. Sci.*, vol. 491, pp. 251–264, 2019.
[39] W. Guo *et al.*, "Efficient attribute-based searchable encryption on cloud storage," *J. Phys.*, vol. 1087, no. 5, 2018, Art. no. 052001.
[40] A. Michalas, "The lord of the shares: Combining attribute-based encryption and searchable encryption for flexible data sharing," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 146–155.
[41] A. Bakas and A. Michalas, "Modern family: A revocable hybrid encryption scheme based on attribute-based encryption, symmetric searchable encryption and SGX," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2019, pp. 472–486.
[42] A. Ge and P. Wei, "Identity-based broadcast encryption with efficient revocation," in *Proc. IACR Int. Workshop Public Key Cryptog.*, 2019, pp. 405–435.
[43] T. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions, *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 35–45, Jan. 2016.
[44] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Proc. Annu. Int. Cryptol. Conf.*, 2006, pp. 290–307.

Jianfei Sun received the PhD degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China. Currently, he is a postdoctoral with the School of Computer Science and Engineering, Nanyang Technological University. His research interests include public key cryptography and network security.

Guowen Xu is currently a postdoctoral with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include searchable encryption and privacy-preserving issues in deep learning.



Tianwei Zhang received the bachelor's degree from Peking University, Beijing, China, in 2011, and the PhD degree from Princeton University, Princeton, New Jersey, in 2017. He is currently an assistant professor with the School of Computer Science and Engineering, Nanyang Technological University. His research interest include computer system security, security threats and defenses in machine learning systems, autonomous systems, computer architecture, and distributed systems.

Hu Xiong (Member, IEEE) received the PhD degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2009. He is currently a full professor with the School of Information and Software Engineering, UESTC. His research interests include applied cryptography and cyberspace security.

Hongwei Li (Senior Member, IEEE) received the PhD degree from the University of Electronic Science and Technology of China, Chengdu, China, in June 2008. He is currently the head and a professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He worked as a postdoctoral fellow with the University of Waterloo from October 2011 to October 2012. His research interests include network security and applied cryptography. He is the distinguished lecturer of IEEE Vehicular Technology Society.

Robert H. Deng (Fellow, IEEE) is currently a AXA chair professor of cybersecurity and professor of information systems with the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017) and Best Paper Award (ESORICS 2020). He served/is serving on the editorial boards of many international journals in security, including the *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Security and Privacy Magazine* and *ACM Transactions on Security and Privacy*.