This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

Privacy-Preserving Task Recommendation Services for Crowdsourcing

Jiangang Shu, *Student Member, IEEE,* Xiaohua Jia, *Fellow, IEEE,* Kan Yang and Hua Wang

Abstract—Crowdsourcing is a distributed computing paradigm that utilizes human intelligence or resources from a crowd of workers. Existing solutions of task recommendation in crowdsourcing may leak private and sensitive information about both tasks and workers. To protect privacy, information about tasks and workers should be encrypted before being outsourced to the crowdsourcing platform, which makes the task recommendation a challenging problem. In this paper, we propose a privacy-preserving task recommendation scheme (PPTR) for crowdsourcing, which achieves the task-worker matching while preserving both task privacy and worker privacy. In PPTR, we first exploit the polynomial function to express multiple keywords of task requirements and worker interests. Then, we design a key derivation method based on matrix decomposition, to realize the multi-keyword matching between multiple requesters and multiple workers. Through PPTR, user accountability and user revocation are achieved effectively and efficiently. Extensive privacy analysis and performance evaluation show that PPTR is secure and efficient.

Index Terms—Crowdsourcing, task recommendation, multi-keyword, privacy-preserving, proxy re-encryption.

1 Introduction

Crowdsourcing [1] is a distributed technique to outsource complex tasks to a massive group of unspecified workers. It is usually used to execute the tasks that require human intelligence and cannot be completed by computers or individuals alone, such as large scale image categorization, text translation, data collection, etc. Nowadays, there are many platforms that enable customers to publish tasks for crowdsourcing and workers to find tasks they would like to take, such as Amazon Mechanical MTurk[1], CrowdFlower[2] and Witmart[3]. However, it is quite burdensome for the workers to manually search for the suitable tasks due to the large number of tasks available in the crowdsourcing platforms. Hence, automatic task recommendation services are required in the crowdsourcing platforms.

In current solutions of task recommendation, *task requesters* and *task workers* have to specify their *task requirements* (a collection of needs that should be met by the assigned workers) and *worker interests* (indicating the tasks that the workers are interest in) to the crowdsourcing platform (*broker*) to facilitate accurate task recommendation services. These information may reveal the professions or job nature of requesters, and the geographic locations or profiles of workers. The private and sensitive information may be used to identify an individual, infer his/her health status, or track his/her daily routines. And the broker is usually not fully trusted in the sense that it may disclose users' information to other parities for profit making, or it may be compromised by malicious adversaries. Hence, revealing sensitive information to the broker for task recommendation may put the privacy of requesters and workers at risk.

A common way to protect the privacy against the broker is to encrypt task requirements and worker interests first and then enable the broker to do the task-worker matching over the encrypted version. Searchable encryption (SE) [20] is a potential technique to solve the problem of matching over encrypted data. However, most of works [20]- [26] in SE only allow a single user who holds the secret key to search. Applying the single-user SE technique to the task recommendation for crowdsourcing by sharing the same secret key among all users will incur two significant shortcomings. Firstly, user accountability cannot be achieved in a provable manner, because all the encrypted task requirements and worker interests are generated under the same secret key. Secondly, user revocation would lead to a high communication and computation cost, because this key-sharing method would require the key redistribution to all non-revoked users and the update of encrypted data after every user revocation. This method is not scalable for the multi-user crowdsourcing situation where the users join and leave frequently. Therefore, the single-user SE cannot be directly applied to the dynamic crowdsourcing environment with multiple requesters and multiple workers.

Proxy re-encryption based schemes [38], [39] are other alternatives to address the above challenge. However, they only provide solutions for the single-keyword matching. Since both task requirements and worker interests are generally specified as a set of keywords, it is not efficient to use a single-keyword matching method to achieve the multi-keyword matching. Moreover, trapdoor indeterminacy cannot be preserved in [38], [39]. Therefore, it is important to design an efficient and secure solution to support the multi-keyword matching for task recommendation services

- *Jiangang Shu and Xiaohua Jia are with the Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong SAR, China.*
  E-mail: jgshu2-c@my.cityu.edu.hk, csjia@cityu.edu.hk.
- *Kan Yang is with the Department of Computer Science, University of Memphis, Memphis, TN, USA.*
  E-mail: kan.yang@memphis.edu.
- *Hua Wang is with the Centre for Applied Informatics, Victoria University, Australia.*
  E-mail: Hua.Wang@vu.edu.au.

1. https://www.mturk.com/mturk/welcome
2. http://crowdflower.com
3. http://www.witmart.com/cn/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

in crowdsourcing.

In this paper, we formulate the problem of Privacy-Preserving Task Recommendation services (PPTR) for crowdsourcing and propose a PPTR scheme, which focuses on two issues: 1) rich expression of task requirements and worker interests, where task requirements and worker interests are expressed as multiple keywords; 2) privacy-preserving multi-keyword matching, where each requester and worker has its own distinct secret key. Considering the number of keywords in the task requirements and worker interests could be very large, it is infeasible to utilize the vector space model based on a predefined keyword dictionary for expression. In PPTR, with exploiting the polynomial function, we represent task requirements and worker interests as vectors of a small size. To achieve the multi-keyword matching, we first design a key derivation method to distribute distinct secret keys to requesters and workers. Then, we utilize asymmetric scalar-product-preserving encryption (ASPE) [7] together with proxy re-encryption, to securely compute the inner products between task requirements and worker interests, encrypted with different secret keys by requesters and workers. Through the above designs, user accountability and user revocation can be achieved effectively and efficiently. Detailed privacy analysis shows that both task privacy and worker privacy are preserved simultaneously. We also conduct an extensive performance evaluation and comparison to validate the efficiency of PPTR. More specifically, our contributions are summarized as follows:

- We formulate the problem of privacy-preserving task recommendation services for crowdsourcing, and also define a set of utility and security goals.
- We exploit the polynomial function to represent task requirements and worker interests as the vectors of the small size, eliminating the predefined keyword dictionary and reducing the computation cost.
- We propose a key derivation method based on matrix decomposition, which can achieve the multi-keyword matching between multiple requesters and multiple workers with utilizing the proxy re-encryption and ASPE.

The rest of the paper is organized as follows. In the next section, we discuss the related work and highlight the difference between our scheme and other multi-user searchable encryption schemes. Then, we provide some necessary preliminaries in Section 3, and formulate our problem including system model, threat model, and utility and security goals in Section 4. In Section 5, we propose a framework for PPTR and also give its security definitions, followed by the detailed construction in Section 6. Security analysis is described in Section 7 and performance evaluation is given in Section 8. In Section 9, we also give other pracical applications where our proposed method can be applied. Finally, we conclude the paper in Section 10.

2 Related Work

Different from access control works [4]- [6], our research is related with two topics, task recommendation and privacy in crowdsourcing, and searchable encryption. They are discussed separately below.

2.1 Task Recommendation and Privacy in Crowdsourcing

With the increasing popularity of crowdsourcing [1]- [3], many studies have been conducted on task recommendation for crowdsourcing. The mode to get crowdsourcing tasks can be mainly classified into two categories: pull mode and push mode. The pull mode is the way where workers actively search over tasks with input keywords. The push mode is the way where the system automatically pushes tasks to the right workers according to different worker models, such as skill and interest [8], worker performance and search history [9], social worker profiles [10]. Meanwhile, many security and privacy issues have also been investigated by [11]- [17], including user privacy, trustworthiness and worker-private quality control. But privacy disclosure in task recommendation has not attracted too much attention. Gong *et al.* [18] proposed a privacy-preserving task recommendation framework for mobile crowdsourcing, which only considers the protection of worker information but ignores the preservation of task information. Such a way may leak worker privacy eventually even when worker information is well protected, because the broker may infer additional worker information by linking the task-worker matching fact with the task information in plaintext. To the best of our knowledge, our prior work [19] is the first one to consider both task privacy and worker privacy for task recommendation. But it only supports the single-keyword matching, which is not efficient enough to deal with the problem of multi-keyword matching in this paper.

2.2 Searchable Encryption

There have been many constructive studies on searchable encryption, including searchable symmetric encryption (SSE) [20]- [25] and public-key encryption with keyword search (PEKS) [26]. Song *et al.* [20] and Boneh *et al.* [26] proposed the first searchable encryption scheme in the symmetric and public setting, respectively. After that, a lot of schemes have been put forward, including fuzzy keyword search [21], single-keyword ranked search [22] and multi-keyword ranked search [23], preferred keyword search [24] and semanic search [25]. However, all of these schemes only apply to the single-user scenario where only the secret key holder is allowed to query. Multi-user SE [27]- [28], [31]- [40] has also been studied, which allows each authorized user to query. Due to a lack of utility features summarized in TABLE 1, they cannot be directly applied into the multi-keyword task recommendation for crowdsourcing, either.

Broadcast encryption is first introduced to realize the multi-user query by Curtmola *et al.* [27], who proposed a general construction to combine broadcast encryption with a single-user SSE. In their setting, the data owner encrypts its data for a fixed group of users, but only authorized users within this group can obtain their own secret keys from the data owner and thereby generate correct trapdoors to query. Moreover, it only allows a single owner to publish the encrypted data. In addition, user enrollment and revocation largely affects the system performance including re-generating the user secret keys or re-encrypting the data. The similar limitations hold for [28] which combines

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

TABLE 1
Utility Comparison with Existing Multi-user Searchable Encryption

Schemes	Multi-keyword	Multi-owner	No middleware	Non-interaction	User scalability
Curtmola *et al.* [27]	×	×	√	√	×
Kiayias *et al.* [28]	×	√	√	√	×
MKSE [31], [32], [33]	×	√	√	√	×
KASE [34]	×	×	√	√	×
Multi-Client SSE [35], [37]	×	×	√	×	×
MSDE [38]	×	√	√	√	√
MuED [39]	×	√	√	×	√
PRMSM [40]	√	√	×	√	√
Our PPTR	√	√	√	√	√

broadcast encryption [29] with anonymous identity-based encryption [30].

Popa *et al.* [31] proposed the concept of multi-key searchable encryption (MKSE) where authorized users can search over multiple documents encrypted by different keys through a single submission. Yang *et al.* [32] utilized the technique of homomorphism and one-way function to achieve the same goal as [31]. Motivated by the work [31], Tang [33] improved MKSE from granularity of user authorization and server-user collusion. In all these MKSE schemes, a data owner needs to submit an adjusting value to the server for each authorized user when encrypting a file. Given a trapdoor from an authorized user, the server needs to transform it to multiple trapdoors for different files under different owners. As a result, the whole process incurs a huge transmission cost on the owner side, and an expensive storage and computation cost on the server side.

Cui *et al.* [34] proposed a key aggregate searchable encryption (KASE) scheme in the single-owner setting. In their setting, the data owner provides an aggregate searchable encryption key to all users such that each user can generate a single trapdoor with this aggregate key. Given the trapdoor, the server also needs to transform it to multiple trapdoors for different files. If we adopt KASE in crowdsourcing directly, a user needs to generate multiple trapdoors for a single query, which incurs a high computation and communication cost on the user side. Based on the Oblivious Cross-Tag (OXT) protocol [36], Jarecki *et al.* [35] proposed a Multi-Client SSE and Faber *et al.* [37] realized boolean query and range query. However, these schemes only allow a single owner to publish the encrypted data. Moreover, the authorized user needs to interact with the data owner to obtain a search token for each query.

Dong *et al.* [38] and Bao *et al.* [39] proposed two similar searchable encryption schemes based on proxy re-encryption in the multi-owner and multi-user setting. In their schemes, each authorized user or owner has its own distinct key pair (user-side key and server-side key) and the users are allowed to search over encrypted data from all the data owners. Their schemes achieve the efficient user enrollment and revocation without the redistribution of new keys. However, they only support the single-keyword matching. Moreover, trapdoor indeterminacy is not protected in both schemes. In addition, the interactive index generation in [39] incurs an extra communication cost. Although [40] proposed a multi-keyword searchable encryption in the multi-owner model, an extra trusted party is introduced as middleware to transform the index and trapdoors before submitting them to the server.

3 PRELIMINARIES

In this section, we describe some background knowledge before going to the details of our proposed scheme.

3.1 Searchable Encryption

Searchable encryption (SE) is a popular technique that allows users to search over encrypted data stored on a semi-trusted server. A generic SE scheme usually contains three different entities: a data owner, a data user and a server. The entire process of SE can be defined as a tuple SE= ($\mathbf{Setup}$, $\mathbf{Enc}$, $\mathbf{TdGen}$, $\mathbf{Test}$), shown as follows:

- $K \leftarrow \mathbf{Setup}(1^\lambda)$. It is run by the data owner to setup the service. It takes as input a security parameter λ and outputs a secret key K.
- $C_w \leftarrow \mathbf{Enc}(K, w)$. It is run by the data owner to encrypt the keyword. It takes as input a keyword w and the secret key K, and outputs a keyword ciphertext C_w for the keyword w.
- $T_{w'} \leftarrow \mathbf{TdGen}(K, w')$. It is run by the data user to generate the trapdoor. It takes as input a query keyword w' and the secret key K, and outputs a trapdoor $T_{w'}$ for the keyword w'.
- $1/0 \leftarrow \mathbf{Test}(C_w, T_{w'})$. It is run by the server to perform the keyword matching over encrypted data. It takes as input the ciphertext C_w and the trapdoor $T_{w'}$, and outputs 1 if $w = w'$; otherwise, it outputs 0.

An SE scheme is correct, for $\forall K \leftarrow \mathbf{Setup}(1^\lambda)$, $\forall w \in \{0,1\}^*$, $\forall C_w \leftarrow \mathbf{Enc}(K, w)$, $\forall T_w \leftarrow \mathbf{TdGen}(K, w)$, we have $\mathbf{Test}(C_w, T_w) = 1$.

3.2 Asymmetric Scalar-Product-Preserving Encryption

Asymmetric Scalar-Product-Preserving Encryption (ASPE) is an encryption method proposed by [7]. It enables the third party to compute the inner product of two encrypted vectors $E(\overrightarrow{p})$ and $E(\overrightarrow{q})$ without knowing the actual values $\overrightarrow{p}$ and $\overrightarrow{q}$ such that $E(\overrightarrow{p}) \cdot E(\overrightarrow{q}) = \overrightarrow{p} \cdot \overrightarrow{q}$. Through a series of processes like randomly splitting, matrix multiplication and artificial dimension addition, neither $\overrightarrow{p}$ nor $\overrightarrow{q}$ can be recovered by analyzing their corresponding ciphertexts if without the secret key. Due to its security and efficient, ASPE is widely adopted to realize the multi-keyword SE [23], [25].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

TABLE 2
PPTR Notations

Notation	Definition
sk_i, rk_i	secret key and re-encryption key for user i
$\mathcal{K}$	user-key mapping set for storing (i, rk_i)
U	user collection in the system
U_W	worker collection in the system
N	number of workers in U_W
I_i	interest of worker i, denoted as $I_i = \{w_{i,1}, \cdots, w_{i,n_i}\}$
$I_i^*, \tilde{I}_i$	ciphertext and transformed ciphertext for I_i
$\tilde{I}$	searchable index $\tilde{I} = \{\tilde{I}_1, \cdots, \tilde{I}_N\}$
Q	task requirement, denoted as $Q = \{q_1, \cdots, q_l\}, l \leq d$
d	maximum number of keywords allowed in Q
$T^*, \tilde{T}$	trapdoor and transformed trapdoor for Q
$W_{\tilde{T},t}$	set of worker identities where t is a threshold

3.3 Vieta's Formulas

Vieta's formulas relate the coefficients of a polynomial to the sums and products of its roots. Let $P(x)$ be a polynomial of degree n, e.g., $P(x) = a_n x^n + \cdots + a_1 x + a_0$, where the coefficient of x^i is a_i and $a_n \neq 0$. According to the Fundamental Theorem of Algebra, we can have

$$a_n x^n + \cdots + a_1 x + a_0 = a_n (x - x_n) \cdots (x - x_1),$$

where $x_1, \cdots, x_n$ are the roots of $P(x)$. For any integer $0 \leq k \leq n$, the $(n-k)$-th coefficient a_{n-k} is related to a signed sum of all possible subproducts of roots, and it can be computed as follows.

$$\sum_{1 \leq i_1 \leq \cdots \leq i_k \leq n} x_{i_1} \cdots x_{i_k} = (-1)^k \frac{a_{n-k}}{a_n}$$

4 Problem Formulation

In this section, we give a formal description of the targeted problem in the paper. We first introduce the system model and the threat model. Then, we set both utility and security goals for addressing such a problem. For easy reference, we summarize all the notations for PPTR in TABLE 2.

4.1 System Model

As shown in Fig. 1, the system model consists of four different entities: multiple *task requesters*, multiple *task workers*, a crowdsourcing service provider called *broker*, and a key manager server called *KMS*. Their roles are defined as follows:

- A task requester is a user who publishes tasks to the broker.
- A task worker is a user who subscribes the tasks from the broker and completes the tasks.
- The broker is the crowdsourcing service provider that recommends the tasks to suitable workers.
- The KMS is the trusted third party that is responsible for user enrollment and user revocation.

4.1.1 User Registration and Revocation

When a user i (a worker or a requester) registers to the system, the KMS generates a key pair (sk_i, rk_i) and assigns the secret key sk_i to the user i and the re-encryption key rk_i to the broker. The broker keeps a user-key mapping set $\mathcal{K}$ for storing (i, rk_i). If the registered user i quits, the KMS will notify the broker to revoke its re-encryption key rk_i.

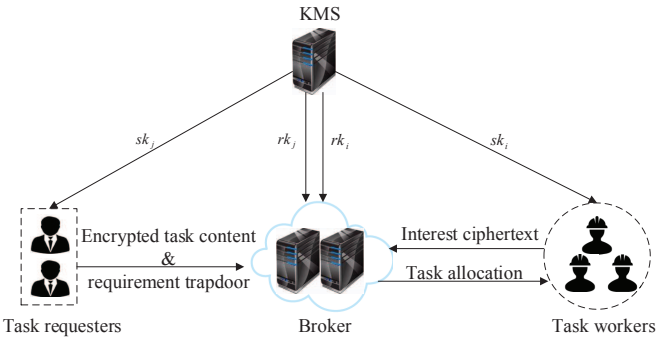


Fig. 1. System model

4.1.2 Task Recommendation

Each worker i has its interest I_i, defined by a set of keywords. Each worker encrypts its interest as I_i^* using its secret key sk_i and registers itself to the broker by specifying its interest I_i^*. Then the broker transforms I_i^* to $\tilde{I}_i$ using the re-encryption key rk_i of the worker i. After that, the broker can build a searchable index $\tilde{I} = \{\tilde{I}_i\}_{1 \leq i \leq N}$. When a task requester j has a new task to be published, it specifies the task requirement Q as a set of keywords, and sets a threshold t that means the interest of each suitable worker for this task must contain at least t requirement keywords. Then it generates the requirement trapdoor T on Q using its secret key sk_j, and sends the trapdoor T and the ciphertext of task content to the broker along with the threshold t.

After receiving the trapdoor T from the requester j, the broker firstly transforms T to $\tilde{T}$ using the corresponding re-encryption key rk_j and then performs a matching between the transformed trapdoor $\tilde{T}$ and each transformed ciphertext $\tilde{I}_i$ in $\tilde{I}$. As a result, the broker can find out a set of suitable workers $W_{\tilde{T},t}$. It then pushes the task in ciphertext to all the workers in $W_{\tilde{T},t}$. The encryption and decryption of task content is out of scope of this paper.

4.2 Threat Model

The threat model for each entity is set as follows:

- The KMS is considered as fully *trusted* and all communications with the KMS are secure.
- Users (requesters and workers) are considered as fully *trusted* in the sense that they could execute the operations properly and protect their secret keys securely. It means that the users wouldn't leak their keys actively and passively to the other entity, e.g., the broker.
- The broker is considered as *honest-but-curious* in the sense that the broker could execute the designed operations honestly while it is curious to infer additional information from trapdoors and ciphertexts. We also assume that the broker wouldn't launch active attacks, such as collusion with users, pretending to be a requester (or a worker).

One possible solution to protect the users' keys is to store their secret keys in the tamper resistant devices, e.g., smarts cards. The assumption for the broker is reasonable, because the broker is usually a large service provider that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

understands the importance of reputation. Active attacks are easy to detect and will damage the broker's reputation once caught.

Since the whole searchable index is built on encrypted interests from different workers, it is difficult for the broker to obtain some background information about the whole index, such as the statistical information of keywords in the index like [23]. Therefore, we only consider the following threat model in this paper.

Known plaintext model. Besides the encrypted data transmitted between workers (or requesters) and the broker, e.g., searchable index $\tilde{I}$ and trapdoor $\tilde{T}$, the broker also knows some plaintext information of worker interests and task requirements, and their corresponding encrypted values.

4.3 Design Goals

To enable the privacy-preserving task recommendation services in the multi-requester and multi-worker setting, A PPTR scheme shall simultaneously satisfy the following utility and security goals.

Utility goals. A valid PPTR scheme should satisfy the following utility goals:

- *Multi-keyword matching for crowdsourcing*. The proposed scheme shall support the multi-keyword matching between multiple requesters and multiple workers. Specifically, the constant-size transformed trapdoor for any requester can be tested with the constant-size transformed ciphertext for any worker.
- *No middleware*. There is no extra middleware to be involved in the generation and transformation of ciphertexts and trapdoors.
- *Non-interaction*. The ciphertext and trapdoor should be independently generated by a single user (worker and requester) without any interaction with the other entities, e.g., the KMS or the broker.
- *User scalability*. The proposed scheme shall support efficient user enrollment and user revocation without redistributing new keys to non-revoked users or updating the encrypted data.

Security goals. We also define interest privacy for workers and trapdoor privacy for tasks.

Interest privacy should meet the following privacy requirements:

- *Interest confidentiality*. Considering worker interests being sensitive, the workers are reluctant to deliver them to the broker in plaintext. To protect privacy, the workers should deal with the worker interests in a cryptographic way.
- *Interest unlinkability*. The encryption function of worker interest should be randomized instead of deterministic. In particular, the broker should not be able to deduce the relationship of worker interests from their corresponding ciphertexts, e.g., whether two worker interests are the same or contain the same keyword(s). Otherwise, the broker can perform a statistic analysis over the ciphertexts and reverse engineer the keywords with background knowledge.

Trapdoor privacy should meet the following privacy requirements:

- *Trapdoor confidentiality*. Task requesters usually prefer to keep the task requirements from being exposed to others including the broker. To protect privacy, the task requesters need to deal with the task requirements in a cryptographic way.
- *Trapdoor unlinkability*. Similar to interest unlinkability, the encryption function of task requirement should also be randomized instead of deterministic. In particular, the broker should not be able to deduce the relationship of task requirements from their corresponding trapdoors, e.g., whether two task requirements are the same or contain the same keyword(s).

5 Privacy-Preserving Task Recommendation

In this section, we first propose a framework for PPTR and then give its security definitions.

5.1 Framework

Based on the system model, we propose a framework PPTR= ($\mathbf{Setup}$, $\mathbf{KeyGen}$, $\mathbf{IntEnc}$, $\mathbf{IntTran}$, $\mathbf{TdGen}$, $\mathbf{TdTran}$, $\mathbf{Match}$, $\mathbf{Revoke}$). It is described as follows:

- $msk \leftarrow \mathbf{Setup}(1^\lambda)$. The system setup algorithm is run by the KMS. It takes as input a security parameter λ and outputs a master secret key msk.
- $(sk_i, rk_i) \leftarrow \mathbf{KeyGen}(msk, i)$. The key generation algorithm is run by the KMS for user enrollment. It takes as input the master secret key msk and an identity i of a worker or requester, and outputs a pair of secret key and re-encryption key (sk_i, rk_i) for the worker or requester.
- $I_i^* \leftarrow \mathbf{IntEnc}(sk_i, I_i)$. The interest encryption algorithm is run by a worker i. It takes as input the secret key sk_i and a worker interest I_i, and outputs a ciphertext I_i^* for the interest I_i.
- $\tilde{I}_i \leftarrow \mathbf{IntTran}(rk_i, I_i^*)$. The interest transformation algorithm is run by the broker. It takes as input the re-encryption key rk_i of the worker i and transforms the ciphertext I_i^* to $\tilde{I}_i$.
- $T \leftarrow \mathbf{TdGen}(sk_j, Q)$. The trapdoor generation algorithm is run by a task requester j. It takes as input the secret key sk_j and a task requirement Q, and outputs a trapdoor T for the task requirement Q.
- $\tilde{T} \leftarrow \mathbf{TdTran}(rk_j, T)$. The trapdoor transformation algorithm is run by the broker. It takes as input the re-encryption key rk_j of the requester j and transforms the trapdoor T to $\tilde{T}$.
- $1/0 \leftarrow \mathbf{Match}(\tilde{I}_i, \tilde{T}, t)$. The matching algorithm is run by the broker. It takes as input the transformed ciphertext $\tilde{I}_i$, the transformed trapdoor $\tilde{T}$ and the threshold t, and outputs 1 if the number of same keywords between them is at least t; otherwise, it outputs 0.
- $\mathbf{Revoke}(i)$. The user revocation algorithm is run by the KMS. It takes as input a user identity i and notifies the broker to revoke the re-encryption key rk_i of the user i.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

5.2 Security Definitions

Before giving the security definitions for PPTR, we first introduce some auxiliary notions that we will use later.

Definition 1 (History). *Let $Q = \{Q_1, \cdots, Q_q\}$ be a sequence of q task requirements from different task requesters. The history of q task requirements is defined as $H(U_W, Q)$ that includes the worker collection and q task requirements.*

Definition 2 (Access Pattern). *The access pattern induced by a history $H(U_W, Q)$ is defined as $\alpha(H) = (U_W(Q_1), \cdots, U_W(Q_q), Y)$, where $U_W(Q_i)$ represents the set of workers that match the task requirement Q_i and $Y = \{Y_{i,j}\}_{1 \le i < N, 1 \le j \le n_i}$ denotes the intermediate information leaked during matching.*

Definition 3 (Search Pattern). *Let $\tilde{T} = \{\tilde{T}_1, \cdots, \tilde{T}_q\}$ denotes a sequence of q trapdoors constructed from Q. The search pattern induced by a history $H(U_W, Q)$ is defined as a $q \times q$ symmetric binary matrix $\sigma(H)$ such that for $1 \le i, j \le q$, $\sigma_{i,j} = 1$ if $\tilde{T}_i = \tilde{T}_j$.*

Definition 4 (Trace). *The trace induced by a history $H(U_W, Q)$ is defined as $\tau(H) = (|I_1|, \cdots, |I_N|, \alpha(H), \sigma(H), |U|)$. Intuitively, the trace $\tau(H)$ contains all the information that the broker is allowed to know.*

Definition 5 (View). *The view induced by a history $H(U_W, Q)$ is defined as $V(H) = (\tilde{I}, \tilde{T}, \mathcal{K})$ which includes the transcript of the interactions between the broker and the involved requesters or workers, together with some knowledge.*

To ensure the security of our proposed scheme, including interest confidentiality and trapdoor confidentiality, we present a simulation-based definition as follows.

Definition 6 (Semantic Security). *A PPTR scheme is semantic secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$, all histories H composed of a worker collection U_W and q task requirements $Q = \{Q_1, \cdots, Q_q\}$, and all functions f, there exists a probabilistic polynomial-time algorithm (simulator) $\mathcal{A}^*$ such that*

$$|Pr[\mathcal{A}(V(H)) = f(H)] - Pr[\mathcal{A}^*(\tau(H)) = f(H)]| \le negl(\lambda)$$

where $V(H)$ is the view based on H, $\tau(H)$ is the trace based on H, and λ is the security parameter.

Intuitively, the notion of semantic security requires that all the information that the broker possesses (e.g., $V(H)$) can also be obtained from the information that the broker is allowed to know (e.g., $\tau(H)$).

To ensure the interest unlinkability, we set the following definition to guarantee that ciphertexts are randomized even if they are generated with the same keyword.

Definition 7 (Randomized Ciphertext). *$\{\tilde{I}_{i,j}\}_{1 \le i \le N, 1 \le j \le n_i}$ is a collection of ciphertexts under the same keyword. A PPTR scheme has (N, ϵ)-randomized ciphertext if for $\forall i_1, i_2 \in [1, N]$, $j_1 \in [1, n_{i_1}]$, $j_2 \in [1, n_{i_2}]$, we have $Pr[\tilde{I}_{i_1,j_1} = \tilde{I}_{i_2,j_2}] < \epsilon$.*

To ensure the trapdoor unlinkability, we set the following definition to guarantee that trapdoors are randomized even if they are generated with the same keyword set.

Definition 8 (Randomized Trapdoor). *Let $\{\tilde{T}_i\}_{1 \le i \le q}$ be a sequence of trapdoors under the same keyword set. A PPTR scheme has (q, σ)-randomized trapdoor if for $\forall i, j \in [1, q]$, we have $Pr[\tilde{T}_i = \tilde{T}_j] < \sigma$.*

6 Detailed Construction of PPTR

6.1 Overview

Motivated by ASPE, we first generate a matrix-based master secret key and randomly decompose it into two matrices for each user, which can be regarded as the secret key and the re-encryption key, respectively. Then, we design a matrix-based proxy re-encryption method to realize the multi-keyword matching for crowdsourcing together with ASPE. Since the original dimensionality of matrix in ASPE or MRSE [23] generally depends on the size of keyword dictionary (e.g., 10,000), such a high dimensional matrix would increase the computation cost. In consideration of efficiency, we adopt a polynomial-based representation method in PPTR to reduce the dimensionality into $(d+1)$ such that each keyword in the worker interest and the whole task requirement can be both transformed into the $(d+1)$-dimensional vectors.

6.2 Construction

The details of the PPTR scheme are constructed as follows.

$\mathbf{Setup}(1^\lambda)$. The KMS randomly generates a $(d+1)$-bit vector S, two $(d+1) \times (d+1)$ invertible matrices $\{M_1, M_2\}$, and a key-based hash function h_s: $\{0,1\}^* \to Z_p^*$ that hashes an arbitrary keyword to a positive integer in Z_p^*. The master secret key is kept as

$$msk = \{M_1, M_2, S, h_s\}.$$

$\mathbf{KeyGen}(msk, i)$. Given a user identity i, the broker randomly chooses two $(d+1) \times (d+1)$ invertible matrices $\{A_{i,1}, B_{i,1}\}$ and computes $A_{i,2} = A_{i,1}^{-1} M_1$, $B_{i,2} = B_{i,1}^{-1} M_2$. Then, the KMS sets the secret key sk_i and the re-encryption key rk_i as follows:

$$sk_i = \{A_{i,1}, B_{i,1}, S, h_s\},$$
$$rk_i = \{A_{i,2}, B_{i,2}\}.$$

The secret key sk_i is transmitted to the user i and the re-encryption key rk_i is sent to the broker along with the user identity i. The broker updates its user-key mapping set $\mathcal{K} = \mathcal{K} \cup (i, rk_i)$.

$\mathbf{IntEnc}(sk_i, I_i)$. To prevent the broker from learning the keywords in the worker interest I_i, the worker i first encrypts each keyword with the key-based hash function h_s and outputs $\{h_s(w_{i,j})\}_{1 \le j \le n_i}$. Then, the worker computes different powers for each $h_s(w_{i,j})$ to generate a $(d+1)$-dimensional vector $(h_s(w_{i,j})^0, \cdots, h_s(w_{i,j})^d)$. At this point, the worker interest I_i can be represented by a matrix below.

$$(\overrightarrow{I}_{i,1}, \cdots, \overrightarrow{I}_{i,n_i}) = \begin{pmatrix} h_s(w_{i,1})^0 & \cdots & h_s(w_{i,n_i})^0 \\ \vdots & \ddots & \vdots \\ h_s(w_{i,1})^d & \cdots & h_s(w_{i,n_i})^d \end{pmatrix}$$

For each keyword vector $\overrightarrow{I}_{i,j} = (h_s(w_{i,j})^0, \cdots, h_s(w_{i,j})^d)^T$, the worker splits it into two random vectors as $\{\overrightarrow{I}^a_{i,j}, \overrightarrow{I}^b_{i,j}\}$: for k from 1 to $d+1$, if $S(k) = 0$, the worker sets $\overrightarrow{I}^a_{i,j}(k) = \overrightarrow{I}^b_{i,j}(k) = \overrightarrow{I}_{i,j}(k)$; if $S(k) = 1$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

the worker randomly sets $\overrightarrow{I}^{a}_{i,j}(k)$ and $\overrightarrow{I}^{b}_{i,j}(k)$ so that $\overrightarrow{I}^{a}_{i,j}(k) + \overrightarrow{I}^{b}_{i,j}(k) = \overrightarrow{I}_{i,j}(k)$. After that, the worker encrypts $\{\overrightarrow{I}^{a}_{i,j}, \overrightarrow{I}^{b}_{i,j}\}$ as $\{A^T_{i,1}\overrightarrow{I}^{a}_{i,j}, B^T_{i,1}\overrightarrow{I}^{b}_{i,j}\}$. Thus, the ciphertext of the keyword $w_{i,j}$ is constructed as $I^*_{i,j} = \{A^T_{i,1}\overrightarrow{I}^{a}_{i,j}, B^T_{i,1}\overrightarrow{I}^{b}_{i,j}\}$ and the whole worker interest I_i is transformed into $I^*_i = (I^*_{i,1}, \cdots, I^*_{i,n_i})$. Finally, the worker i submits the ciphertext I^*_i to the broker together with the identity i.

IntTran(rk_i, I^*_i). When receiving the ciphertext I^*_i from the worker i, the broker finds the corresponding re-encryption key rk_i. For each sub-ciphertext $I^*_{i,j}$, the broker re-encrypts it as follows.

$$\begin{aligned}\tilde{I}_{i,j} &= \{A^T_{i,2}A^T_{i,1}\overrightarrow{I}^{a}_{i,j}, B^T_{i,2}B^T_{i,1}\overrightarrow{I}^{b}_{i,j}\}\\ &= \{M^T_1\overrightarrow{I}^{a}_{i,j}, M^T_2\overrightarrow{I}^{b}_{i,j}\}\end{aligned}$$

After that, the ciphertext of the worker interest I_i is transformed into $\tilde{I}_i = (\tilde{I}_{i,1}, \cdots, \tilde{I}_{i,n_i})$. With the interests from N workers in the system, the broker can build a searchable index $\tilde{I} = \{\tilde{I}_i\}_{1\le i\le N}$.

TdGen(sk_j, Q). The task requester j specifies the task requirement as $Q = \{q_1, \cdots, q_l\}$ consisting of l keywords. To make the number of keywords in the task requirement consistent, $d-l$ dummy keywords $\{q_{l+1}, \cdots, q_d\}$ are added to Q. Note that each dummy keyword is different from any real dictionary word and thus it has no impact on the matching result. Then the task requirement is encrypted with the key-based hash function h_s as

$$h_s(Q) = \{h_s(q_1), \cdots, h_s(q_l), h_s(q_{l+1}), \cdots, h_s(q_d)\}.$$

Considering the intense computation of polynomial root tracking, which requires $2d^2$ complex floating point operations to find all roots of a d-th order polynomial [43], we construct a polynomial function to hide the keywords of the task requirement. Specifically, a polynomial function of degree d is constructed as

$$\begin{aligned}f(x) &= (x - h_s(q_1)) \times \cdots \times (x - h_s(q_d))\\ &= b_0 + b_1 x + \cdots + b_d x^d.\end{aligned}$$

Subsequently, the coefficients of the polynomial function can be extracted to construct the query vector as

$$\overrightarrow{Q} = (b_1, \cdots, b_d)^T.$$

After that, the task requester splits $\overrightarrow{Q}$ into two random vectors as $\{\overrightarrow{Q}^a, \overrightarrow{Q}^b\}$: for k from 1 to $d+1$, if $S(k) = 0$, the task requester randomly sets $\overrightarrow{Q}^a$ and $\overrightarrow{Q}^b$ such that $\overrightarrow{Q}^a(k) + \overrightarrow{Q}^b(k) = \overrightarrow{Q}(k)$; if $S(k) = 1$, the task requester sets $\overrightarrow{Q}^a(k) = \overrightarrow{Q}^b(k) = \overrightarrow{Q}(k)$. Then the task requester encrypts $\{\overrightarrow{Q}^a, \overrightarrow{Q}^b\}$ as $\{A^{-1}_{j,1}\overrightarrow{Q}^a, B^{-1}_{j,1}\overrightarrow{Q}^b\}$. At this point, the trapdoor of the task requirement is built as $T = \{A^{-1}_{j,1}\overrightarrow{Q}^a, B^{-1}_{j,1}\overrightarrow{Q}^b\}$. Finally, the requester j submits the trapdoor T to the broker together with the identity j.

TdTran(rk_j, T). Upon receiving the trapdoor T from the requester j, the broker re-encrypts it with the corresponding re-encryption key and outputs the transformed trapdoor $\tilde{T}$ as

$$\begin{aligned}\tilde{T} &= \{A^{-1}_{j,2}A^{-1}_{j,1}\overrightarrow{Q}^a, B^{-1}_{j,2}B^{-1}_{j,1}\overrightarrow{Q}^b\}\\ &= \{M^{-1}_1\overrightarrow{Q}^a, M^{-1}_2\overrightarrow{Q}^b\}.\end{aligned}$$

Match$(\tilde{I}_i, \tilde{T}, t)$. For each transformed ciphertext $\tilde{I}_i$ in $\tilde{I}$, the broker tests $\tilde{T}$ with each sub-interest $\tilde{I}_{i,j}$ in $\tilde{I}_i$, and determines whether the keyword $w_{i,j}$ is included in the task requirement by computing $Y_{i,j}$ as follows.

$$\begin{aligned}Y_{i,j} &= \tilde{T}^T \cdot \tilde{I}_{i,j}\\ &= \{M^{-1}_1\overrightarrow{Q}^a, M^{-1}_2\overrightarrow{Q}^b\}^T \cdot \{M^T_1\overrightarrow{I}^{a}_{i,j}, M^T_2\overrightarrow{I}^{b}_{i,j}\}\\ &= (\overrightarrow{Q}^a)^T\overrightarrow{I}^{a}_{i,j} + (\overrightarrow{Q}^b)^T\overrightarrow{I}^{b}_{i,j}\\ &= \overrightarrow{Q}^T\overrightarrow{I}_{i,j}\end{aligned}$$

We have $Y_{i,j} = 0$ if the keyword $w_{i,j}$ is included in the task requirement Q. The number of zero values in $Y_i = (Y_{i,1}, \cdots, Y_{i,n_i})$ is used as the matching score $score_i$. It indicates the number of matched keywords between the task requirement Q and the worker interest I_i. If the matching score $score_i \ge t$, the broker adds the worker identity i into the result $W_{\tilde{T},t}$. Finally, the broker gets a set of suitable worker identities $W_{\tilde{T},t} = \{i\}_{score_i \ge t}$ after computing the matching scores of each transformed ciphertext $\tilde{I}_i$ in $\tilde{I}$.

Revoke(i). Given a user identity i, the KMS notifies the broker to update its user-key mapping set $\mathcal{K} = \mathcal{K} \setminus (i, rk_i)$. Once the re-encryption key rk_i is deleted, the transformed ciphertext and the transformed trapdoor can be no longer correctly generated by the revoked user i. Thus, user revocation is effectively achieved without the renewal of keys or the update of interest ciphertexts.

6.3 Discussion

Since each sub-interest in the interest ciphertext is independently stored on the broker, our PPTR scheme supports the efficient update of worker interest. In order to reduce the communication cost, each worker locally stores a copy of its worker interest in a sequential manner and the broker likewise stores the ciphertext in the same order. When the worker wants to delete a keyword $w_{i,j}$ from its worker interest, it only needs to send the broker a sequential number j, which represents the order of the keyword in the worker interest. When receiving the deletion request, the broker finds the sub-interest $\tilde{I}_{i,j}$ and deletes it. If the worker needs to insert a new keyword $w_{i,j}$ into its worker interest, it performs the interest encryption on the keyword and submits the keyword ciphertext to the broker. On the insertion request, the broker transforms the received keyword ciphertext and inserts the transformed keyword ciphertext to the end of $\tilde{I}_i$. Both keyword deletion and insertion will not affect other keywords in the interest ciphertext.

7 Security Analysis

Theorem 1. *Index confidentiality and trapdoor confidentiality are preserved in the known plaintext model if the broker cannot obtain the splitting vector S and the key-based hash function h_s.*

Proof. In the known plaintext model, the broker only knows the encrypted values $\{\tilde{I}^a_{i,j}, \tilde{I}^b_{i,j}\}$ and the corresponding keyword $w_{i,j}$. Without the key-based hash function h_s and the splitting vector S, the broker cannot obtain the vector $\overrightarrow{I}_{i,j}$ and thereby can only set $\overrightarrow{I}^{a}_{i,j}$ and $\overrightarrow{I}^{b}_{i,j}$ as two random $(d+1)$-dimensional vectors. Thus, linear equations can be created as follows: $M^T_1\overrightarrow{I}^{a}_{i,j} = \tilde{I}^a_{i,j}$ and $M^T_2\overrightarrow{I}^{b}_{i,j} = \tilde{I}^b_{i,j}$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

where M_1 and M_2 are both unknown matrices. There are $2(d+1)\sum_{1\leq i\leq N} n_i$ unknowns in $\overrightarrow{I}_{i,j}^{a}$ and $\overrightarrow{I}_{i,j}^{b}$, and $2(d+1)^2$ unknowns in M_1 and M_2. Since there are only $2(d+1)\sum_{1\leq i\leq N} n_i$ equations which are less than the number of unknowns, the broker doesn't have enough information to solve the equations. For the linear equations created for the trapdoor, there are $2(d+1)$ unknowns in the requirement vector and $2(d+1)^2$ unknowns in M_1^{-1} and M_2^{-1}. Since there are only $2(d+1)$ equations, it is also impossible for the broker to solve the equations. Even chosen-plaintext attacks launched in [42] cannot break the confidentiality, bacause the initial vectors for interests and trapdoor are concealed by the key-based hash function h_s, dummy keywords injection and polynomial factoring. Therefore, without S and h_s, index confidentiality and trapdoor confidentiality are preserved. □

Theorem 2. *The PPTR scheme achieves the semantic security if h_s is a pseudorandom function.*

Proof. Given a history H, there exists a probabilistic polynomial-time simulator $\mathcal{A}^*$ that takes $\tau(H)$ as input and outputs the view $V(H^*) = (\tilde{I}^*, \tilde{T}^*, \mathcal{K}^*)$ with the restriction $\tau(H) = \tau(H^*)$.

Given the leaked immediate information Y in the access pattern $\sigma(H)$, $\mathcal{A}^*$ can represent Y as a $(\sum_{i=1}^{N} n_i) \times q$ matrix. Since there are $(\sum_{i=1}^{N} n_i) \times (d+1)$ unknown variables in the interest vectors and $(d+1) \times q$ unknown variables in the requirement vectors, $\mathcal{A}^*$ can represent them as a $(\sum_{i=1}^{N} n_i) \times (d+1)$ matrix M_I and a $(d+1) \times q$ matrix M_Q, respectively. Therefore, we have $M_I \times M_Q = Y$. Let $Y(i:j)$ be the sub-matrix from the i-th row to the j-th row in Y. Since $rank(M_I) \leq d+1$ and $rank(M_Q) \leq d+1$, we have $rank(Y) \leq d+1$. Therefore, there exists a $(\sum_{i=1}^{N} n_i - d - 1) \times (d+1)$ matrix L such that $Y(d+2 : \sum_{i=1}^{N} n_i) = L \times Y(1:d+1)$. Let E be a $(d+1) \times (d+1)$ unit matrix, $M_I = (E, L)^T$, $M_Q = Y(1:d+1)$, we also have $M_I \times M_Q = Y$. By this observation, the simulator $\mathcal{A}^*$ randomly generates a $(d+1) \times (d+1)$ matrix M_r and sets $M_I^* = M_I M_r$, $M_Q^* = M_r^{-1} M_Q$. After that, the simulator $\mathcal{A}^*$ randomly generates a $(d+1)$-bit vector S^*. For $1 \leq i \leq |U|$, the simulator $\mathcal{A}^*$ randomly selects two $(d+1) \times (d+1)$ invertible matrices $A_{i,1}^*$, $B_{i,1}^*$, and sets $M_1^* = A_{1,1}^* \times \cdots \times A_{|U|,1}^*$, $M_2^* = B_{1,1}^* \times \cdots \times B_{|U|,1}^*$.

- (Generating $\mathcal{K}^*$): Since each $(A_{i,1}^*, B_{i,1}^*)$ is actually associated with a user in U as its secret key, the corresponding re-encryption key can be computed as $(A_{i,1}^{*-1} M_1^*, B_{i,1}^{*-1} M_2^*)$. Therefore, the user-mapping key $\mathcal{K}^*$ can be simulated correctly.
- (Generating $\tilde{T}^*$): For $1 \leq i \leq q$, the simulator $\mathcal{A}^*$ extracts the i-th column of M_Q^* as the task requirement vector $\overrightarrow{Q}_i^*$. Then $\mathcal{A}^*$ randomly selects a user identity i as the corresponding task requester, and picks up an element from $\{(A_{i,1}^*, B_{i,1}^*)\}_{1\leq i\leq |U|}$ as the secret key. After that, $\mathcal{A}^*$ simulates the operations of randomly splitting and matrix multiplication as **TdGen**, followed by **TdTran**. Finally, $\mathcal{A}^*$ figures out the trapdoor $\tilde{T}^*$.
- (Generating $\tilde{I}^*$): For each row in M_I^*, the simulator $\mathcal{A}^*$ treats it as a keyword vector. Then $\mathcal{A}^*$ can simulate the operations of randomly splitting and matrix multiplication as **IntEnc**, followed by **IntTran**. Therefore, the index $\tilde{I}^*$ can be constructed correctly.

At last, the simulator $\mathcal{A}^*$ outputs the view $V(H^*) = (\tilde{I}^*, T^*, \mathcal{K}^*)$. Then, we show that $V(H)$ and $V(H^*)$ are indistinguishable by comparing component by component.

- For $\mathcal{K}$ and $\mathcal{K}^*$, since $(A_{i,1}, B_{i,1})$ and $(A_{i,1}^*, B_{i,1}^*)$ are both random, a real re-encryption key $(A_{i,1}^{-1} M_1, B_{i,1}^{-1} M_2)$ and a simulated re-encryption key $(A_{i,1}^{*-1} M_1^*, B_{i,1}^{*-1} M_2^*)$ are indistinguishable.
- For $\tilde{T}$ and $\tilde{T}^*$, let us consider a real task requirement vector $\overrightarrow{Q}$ which is the coefficients based on $h_s(w)$ and a simulated task requirement vector $\overrightarrow{Q}^*$ which is one column in a random matrix M_Q^*, $\overrightarrow{Q}$ and $\overrightarrow{Q}^*$ are computationally indistinguishable if h_s is a pseudo-random function. Through a sequence of operations like randomly splitting and matrix multiplication, $\tilde{T}$ and $\tilde{T}^*$ are also computationally indistinguishable.
- Similarly, $\tilde{I}$ and $\tilde{I}^*$ are computationally indistinguishable.

□

Theorem 3. *The PPTR scheme achieves the randomized ciphertext for worker interest.*

Proof. Suppose two ciphertexts $\tilde{I}_{i_1,j_1}$ and $\tilde{I}_{i_2,j_2}$ are both generated from the same keyword w through the algorithms of **IntEnc** and **IntTran**. Their corresponding keyword vectors for w are both defined as $\overrightarrow{I}_{i_1,j_1} = \overrightarrow{I}_{i_2,j_2} = (h_s(w)^0, \cdots, h_s(w)^d)^T$. After that, for each dimension k where $S(k)$ is equal to 1, $\overrightarrow{I}_{i_1,j_1}(k)$ and $\overrightarrow{I}_{i_2,j_2}(k)$ need to be split into two random values. Suppose $Pr[\{\overrightarrow{I}_{i_1,j_1}^{a}(k), \overrightarrow{I}_{i_1,j_1}^{b}(k)\} = \{\overrightarrow{I}_{i_2,j_2}^{a}(k), \overrightarrow{I}_{i_2,j_2}^{a}(k)\}] = \xi$ for any dimension k where $S(k)$ is equal to 1, we have $Pr[\tilde{I}_{i_1,j_1} = \tilde{I}_{i_2,j_2}] = \xi^{s_1}$ where s_1 are the numbers of 1 in the vector S. Since $\xi \to 0$, then $Pr[\tilde{I}_{i_1,j_1} = \tilde{I}_{i_2,j_2}] \to 0$. Hence, Theorem 3 is proved. □

Theorem 4. *The PPTR scheme achieves the randomized trapdoor for task requirement.*

Proof. Dummy keyword injection: suppose $D_i, D_j \subset D$ are two dummy keyword sets injected into the same keyword set $Q = \{q_1, \cdots, q_l\}$, respectively, then $Pr[D_i = D_j] = \frac{1}{C_n^{d-l}}$ where n is the size of dummy keyword dictionary D.

Randomly splitting: given two same coefficient vectors $\overrightarrow{Q}_i$ and $\overrightarrow{Q}_j$ where $\overrightarrow{Q}_i = \overrightarrow{Q}_j = \{b_1, \cdots, b_d\}$, for each dimension k where $S(k)$ is equal to 0, $\overrightarrow{Q}_i(k)$ and $\overrightarrow{Q}_j(k)$ need to be split into two random values. Suppose $Pr[\{\overrightarrow{Q}_i^{a}(k), \overrightarrow{Q}_i^{b}(k)\} = \{\overrightarrow{Q}_j^{a}(k), \overrightarrow{Q}_j^{a}(k)\}] = \varrho$ for any dimension k where $S(k)$ is equal to 0, we have $Pr[\tilde{T}_i = \tilde{T}_j] = \varrho^{s_0}$ where s_0 are the numbers of 0 in the vector S.

Combining dummy keyword injection with randomly splitting, we get a conclusion defined as follows:

Given two trapdoors $\tilde{T}_i$ and $\tilde{T}_j$, both constructed through the algorithms of **TdGen** *and* **TdTran** *with the same keyword set $Q = \{q_1, \cdots, q_l\}$, we have $Pr[\tilde{T}_i = \tilde{T}_j] = \frac{1}{C_n^{d-l}} \cdot \varrho^{s_0}$.*

Since $\varrho \to 0$, then $Pr[\tilde{T}_i = \tilde{T}_j] \to 0$. Hence, Theorem 4 is proved. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

TABLE 3
Notations

Notation	Definition
e	exponentiation operation on group G
e_z	exponentiation operation on Z_p^*
p	pairing operation on group G
h_s	key-based hash function: $\{0,1\}^* \rightarrow Z_p^*$
H	collision-resistant hash function: $\{0,1\}^* \rightarrow G$
h	collision-resistant hash function: $G \rightarrow \{0,1\}^k$
M	multiplication
SE	symmetric encryption operation
SD	symmetric decryption operation
$\|\mathcal{M}\|$	plaintext size under a symmetric key encryption
$\|\mathcal{C}\|$	ciphertext size under a symmetric key encryption
$\|G\|$	element size in G
$\|Z_p^*\|$	element size in Z_p^*
$\|V_k\|$	size of k-dimensional vector
n	size of keyword dictionary D
n_i	number of keywords in worker interest
Q	task requirement, denoted as $Q = \{q_1, \cdots, q_l\}$, $l \leq d$
d	maximum number of keywords allowed in Q

TABLE 4
Communication Cost

Scheme	Ciphertext	Trapdoor
MSDE	$\|Z_p^*\| + 2n_i \cdot \|G\| + n_i \cdot k$	$\|Z_p^*\| + 2l \cdot \|G\|$
MuED	$\|Z_p^*\| + 2n_i \cdot \|G\| + \|\mathcal{M}\| + n_i \cdot \|\mathcal{C}\|$	$\|Z_p^*\| + l \cdot \|G\|$
MRSE	$2\|V_{n+2}\|$	$2\|V_{n+2}\|$
PPTR	$\|Z_p^*\| + 2n_i \cdot \|V_{d+1}\|$	$\|Z_p^*\| + 2\|V_{d+1}\|$

8 Performance Evaluation

In this section, we measure the performance of PPTR and compare it with three related works: two multi-owner multi-user single-keyword searchable encryption schemes: MSDE [38] and MuED [39], and a most popular single-user multi-keyword searchable encryption scheme: MRSE [23]. All the notations used in the following parts can be referred in TABLE 3.

8.1 Theoretical Analysis

TABLE 4 and TABLE 5 give the communication cost and computation complexity for PPTR in comparison with MSDE, MuED and MRSE. In PPTR, each keyword in the worker interest needs to go though one key-based hash operation h_s, $(d+1)$ exponentiation operations e_z and $2(d+1)^2$ multiplication operations with the secret key. After interest encryption, it outputs 2 $(d+1)$-dimensional vectors as the ciphertext for each keyword. Therefore, the total ciphertext transmitted to the broker is $2n_i$ $(d+1)$-dimensional vectors together with a user identity Z_p^*. Interest transformation is to multiply 2 $(d+1)$-dimensional vectors with 2 $(d+1) \times (d+1)$ matrices (re-encryption key) respectively for each keyword ciphertext. In the trapdoor generation, the whole task requirement needs to go through d key-based hash operations h_s, $\sum_{j=1}^{j=d} C_d^j (j-1)$ multiplication operations to compute the coefficients in the Vieta's formulas of degree d, and $2(d+1)^2$ multiplication operations with the secret key to generate 2 $(d+1)$-dimensional vectors as the trapdoor. Therefore, the trapdoor transmitted to the broker is 2 $(d+1)$-dimensional vectors together with a user identity Z_p^*. Similar to interest transformation, trapdoor transformation is to multiply 2 $(d+1)$-dimensional vectors (trapdoor) with 2 $(d+1) \times (d+1)$ matrices (re-encryption key) respectively. In the matching, the broker needs to compute the inner product between the trapdoor (2 $(d+1)$-dimensional vectors) and each keyword ciphertext (2 $(d+1)$-dimensional vectors) in the worker interest.

Compared with MSDE and MuED, we adopt the lightweight operation (e.g., multiplication), instead of using time-consuming operations (e.g., exponentiation and pairing), in designing an efficient PPTR and meanwhile preserve the trapdoor indeterminacy. Compared with MRSE, we exploit the polynomial function to express task requirements and worker interests, which eliminates the predefined keyword dictionary and further reduces the computation cost. Such advantages make PPTR more applicable to practical crowdsourcing systems.

8.2 Experimental Study

Dataset and setting. Due to the lack of public real-world dataset on crowdsourcing tasks, we obtain the task data from a third-party platform, called MTurk Tracker [41], which regularly crawls the Human Intelligence Tasks (HITs) from Amazon Mechanical Turk. We collect 1,000 different HITs from different task requesters as our task dataset through available APIs, and the keywords of a HIT can be regarded as a task requirement. According to our statistics, the number of the keywords in a task requirement ranges from 1 to 15 and its average is 3.74. To compare with MRSE, we set a keyword dictionary D which contains 4,744 unique keywords from all the task requirements. Moreover, we synthesize an interest dataset with 10,000 workers and each worker interest contains 1-10 keywords randomly chosen from D. All the schemes are implemented in java on a Windows 7 PC with an Intel Core i7 at 3.60GHz and 8G RAM. Implementations of MSDE and MuED rely on a symmetric elliptic curve in the JPBC library [44] with base filed size 512-bit and embedding degree 2, and a 160-bit prime p.

Interest encryption. Fig. 2 shows the time cost of interest encryption for four schemes with varying number of keywords in the worker interest. The time cost of interest encryption in MSDE, MuED and PPTR all increases as the number of keywords increases, as shown in Fig. 2(a) and Fig. 2(c). This is because the larger number of keywords in the worker interest, the more sub-indexes need to be built in these three schemes, which is verified by the computation complexity of **IntEnc** for these three schemes. From Fig. 2(b), we can see that the time cost of interest encryption in MRSE is insensitive to the number of keywords when the size of keyword dictionary is fixed, while it will suffer a quadratic growth with the size of keyword dictionary, which can be easily proved by the computation complexity of **IntEnc** for MRSE. Moreover, we also vary different value of d to evaluate the efficiency of interest encryption for PPTR, as shown in Fig. 2(c). It demonstrates that the time cost of interest encryption in PPTR also increases as d increases. But the time cost to encrypt 10 interest keywords in PPTR is at most 1.5ms, which is less than 2ms in MRSE when the size of keyword dictionary is set as small as 1,000. Therefore, interest encryption in PPTR is most efficient among the four schemes, which demonstrates that PPTR is quite suitable for workers with the limited computing power.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

TABLE 5
Computation Complexity

Scheme	IntEnc	IntTran	TdGen	TdTran	Match
MSDE	$(2n_i+1)\cdot e+n_i\cdot h_s+h$	$n_i\cdot e$	$(3+2l)\cdot e+l\cdot h_s$	$l\cdot e$	$l\cdot e+n_i\cdot l\cdot h$
MuED	$2n_i\cdot e+n_i\cdot H+n_i\cdot h+n_i\cdot SE$	$n_i\cdot p$	$l\cdot e+l\cdot H$	$l\cdot p$	$l\cdot h+n_i\cdot l\cdot SD$
MRSE	$2(n+2)^2\cdot M$	-	$2(n+2)^2\cdot M$	-	$2(n+2)\cdot M$
PPTR	$n_i\cdot h_s+n_i(d+1)\cdot e_z+2n_i(d+1)^2\cdot M$	$2n_i(d+1)^2\cdot M$	$d\cdot h_s+\left(\sum_{j=1}^{j=d}C_d^j(j-1)+2(d+1)^2\right)\cdot M$	$2(d+1)^2\cdot M$	$2n_i(d+1)\cdot M$

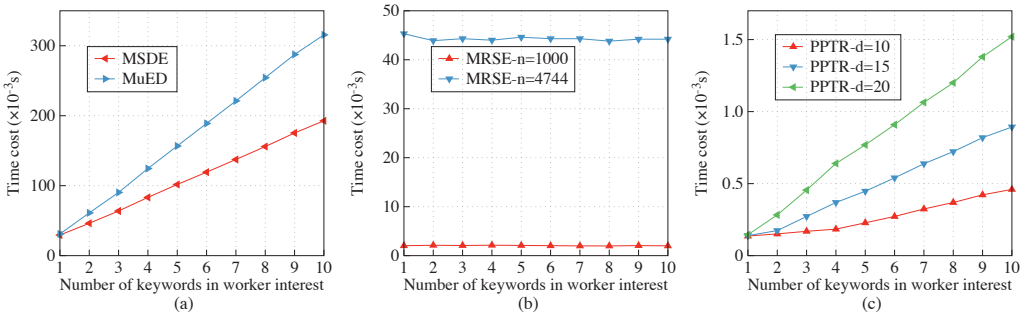


Fig. 2. Time cost of interest encryption

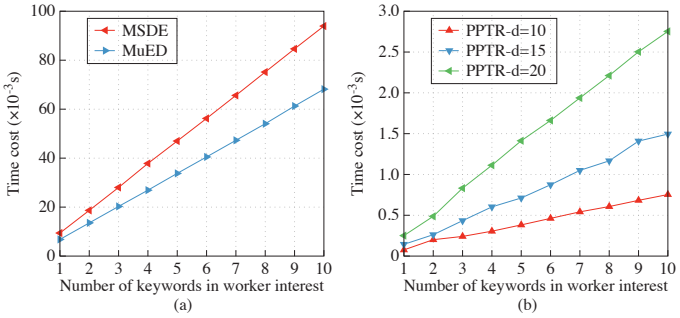


Fig. 3. Time cost of interest transformation

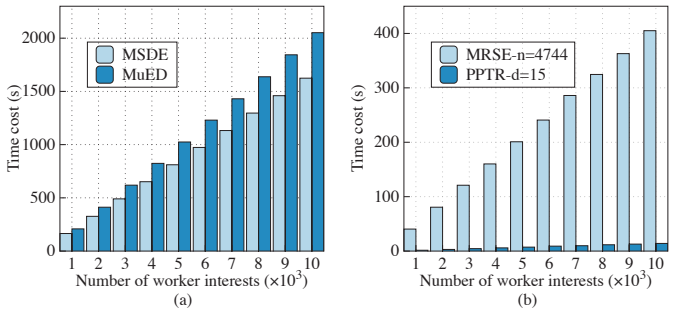


Fig. 4. Time cost of index construction

Interest transformation. We also evaluate the efficiency of interest transformation for PPTR in comparison with MSDE and MuED, as shown in Fig. 3. Similar to interest encryption above, the time cost in these three schemes all increases as the number of keywords increases. Interest transformation in PPTR is much more efficient than these two exponentiation-based or pairing-based schemes, MSDE and MuED. Therefore, PPTR will not pose a burden on the broker side.

Index construction. Combining interest encryption with interest transformation, we can get the total time cost for index construction. Fig. 4 shows the time cost of index construction for four schemes under different number of worker interests. When $d = 15$ in PPTR, building the whole index for 10,000 worker interests only takes about 14s. It shows that PPTR is more efficient than the other three schemes in the terms of index construction.

Trapdoor generation. We vary different number of keywords in the task requirement to evaluate the efficiency of trapdoor generation for four schemes, as shown in Fig. 5. It shows that the time cost of trapdoor generation in MSDE and MuED both increases with the increasing number of keywords, while MRSE and PPTR are both insensitive when n and d are fixed. Fig. 5(c) demonstrates that, given the same task requirement, the time cost of trapdoor generation in PPTR when $d = 10$, $d = 15$, $d = 20$ is about 0.3ms, 12ms, 500ms, respectively. In consideration of efficiency, we can set $d = 15$ in practice. This setting is enough for our experiment, because the maximum number of keywords in the task requirement in the dataset is 15. When $d = 15$, trapdoor generation in PPTR is more efficient than MSDE, MuED and MRSE-n=4744, and it will not pose a burden on the task requester.

Trapdoor transformation. Fig. 6 shows the time cost of trapdoor transformation for PPTR in comparison with MSDE and MuED. It shows that the time cost of trapdoor transformation in MSDE and MuED both increases with the increasing number of keywords in the task requirement, while PPTR is insensitive to the number of keywords when d is fixed. When $d = 15$, PPTR takes about 0.11ms to re-encrypt a trapdoor, which is much less than the time cost (tens of milliseconds) in both MSDE and MuED.

Match. Fig. 7 shows the time cost of matching for four schemes. It shows that given the same task requirement, the time cost of matching in MSDE, MuED and PPTR is all affected by the number of keywords in the worker interest, while MRSE is insensitive, as shown in Fig. 7(a). Fig. 7(b) shows that the time cost of matching in both PPTR and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

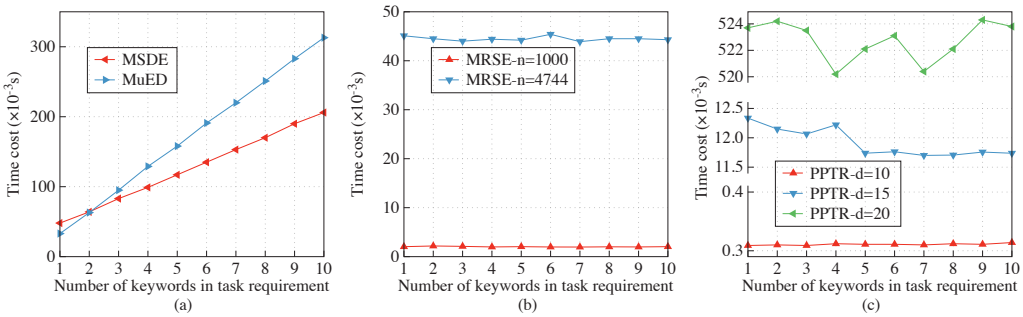


Fig. 5. Time cost of trapdoor generation

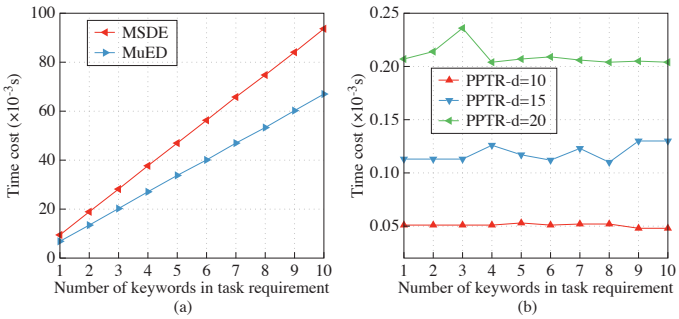


Fig. 6. Time cost of trapdoor transformation

MRSE is insensitive to the number of keywords in the task requirement. From Fig. 7(a) and Fig. 7(b), we can observe that the matching in PPTR costs less time than the other three schemes. We also evaluate the time cost of matching with different number of worker interests in index, as shown in Fig. 7(c). It shows that it only takes about 1.17s for PPTR to match a trapdoor with the whole index consisting of 10,000 worker interests. This further demonstrates that the matching process in PPTR is most efficient among four schemes.

9 Other Applications

Besides the targeted crowdsourcing application in this paper, the underlying method in the proposed PPTR can be also applied into many other practical multi-owner and multi-user applications.

9.1 Privacy-Preserving Publish-Subscribe

In a content-based publish-subscribe system, there are multiple publishers that publish contents as notifications, multiple subscribers that express their interests as subscriptions, and a platform that disseminates the notifications from the publishers to all the interested subscribers. Publishers and subscribers may join and leave the system frequently. When data privacy is considered against the non-trusted platform, both notifications and subscriptions need to be encrypted first and then outsourced to the platform. In this case, our proposed method can be utilized to allow the platform perform the matching between encrypted notifications and encrypted subscriptions in the multi-publisher and multi-subscriber environment.

9.2 Encrypted Data Search

In order to reduce the IT cost, a company outsources its data management to a service provider such that every employee can share his/her own data to others via the service provider. Employees may join and leave the company frequently. When data privacy is considered against the non-trusted service provider, shared data needs to be encrypted by the employees first before outsourcing. When the service provider stores the encrypted data, it introduces a new problem that how to retrieve the data of interest. Due to the large scale of data, it's impractical to download all the data and decrypt it locally. In this case, our proposed method can be adopted to realize the encrypted data search in the multi-owner and multi-user setting.

10 Conclusion

In the paper, we formulated the problem of privacy-preserving task recommendation services for crowdsourcing and established a set of utility and security goals. To solve the problem, we proposed a non-interactive PPTR scheme without relying on any middleware. In PPTR, we exploited the polynomial function to express task requirements and worker interests, eliminating the predefined keyword dictionary and reducing the computation cost. We also designed a key derivation method based on matrix decomposition, which realizes the multi-keyword matching for crowdsourcing with utilizing the proxy re-encryption and ASPE. User accountability and user revocation are achieved effectively and efficiently. Through a detailed privacy analysis, both task privacy and worker privacy are preserved simultaneously. Finally, we validated the practicality of PPTR by conducting comprehensive comparisons with MSDE, MuED and MRSE. Through theoretical analysis and experimental study, the proposed PPTR scheme outperforms the other three schemes in many aspects.

Acknowledgment

This work was supported by grants from NSF China Grant No. 61672195 and Research Grants Council of Hong Kong [Project No. CityU 11205014 and CityU 11204215]. Xiaohua Jia is the corresponding author of this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2791601, IEEE Transactions on Services Computing

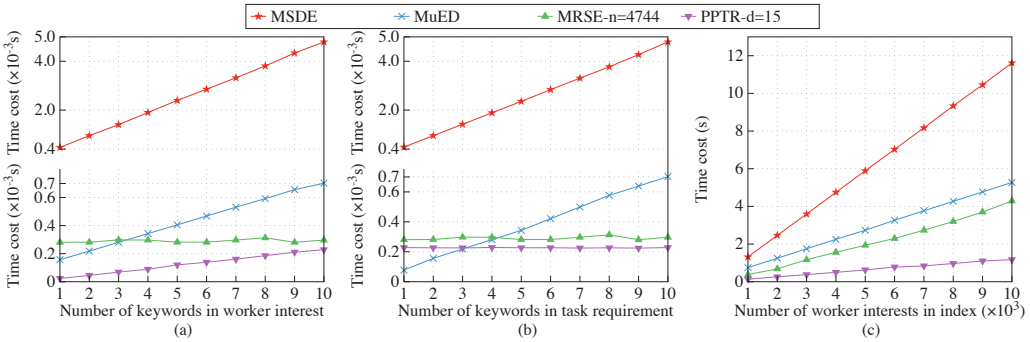


Fig. 7. Time cost of match. (a) under different number of keywords in worker interest when the number of keywords in task requirement is 10; (b) under different number of keywords in task requirement when the number of keywords in worker interest is 10; (c) under different number of worker interests in index when the number of keywords in task requirement is 10.

REFERENCES

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1-4, 2006.

[2] T. Liu, Y. Zhu, Q. Zhang, and A. Vasilakos, "Stochastic optimal control for participatory sensing systems with heterogenous requests," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1619-1631, 2016.

[3] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, pp. 88, 2016.

[4] M. Ali, R. Dhamotharan, E. Khan, S. U. Khan, A. V. Vasilakos, K. Li, and A. Y. Zomaya, "SeDaSC: secure data sharing in clouds," *IEEE Systems Journal*, vol. 11, no. 2, pp. 395-404, 2017.

[5] Z. Yan, X. Li, M. Wang, and A. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Transactions on Cloud Computing*, 2017.

[6] Q. Liu, H. Zhang, J. Wan, and Chen, X. "An Access Control Model for Resource Sharing based on the Role-Based Access Control Intended for Multi-domain Manufacturing Internet of Things," *IEEE Access*, 2017.

[7] W. K. Wong, D. W. L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 139-152.

[8] V. Ambati, S. Vogel, and J. G. Carbonell, "Towards Task Recommendation in Micro-Task Markets," in *Proceedings of Human Computation*, 2011, pp. 1-4.

[9] M. C. Yuen, I. King, and K. S. Leung, "Task recommendation in crowdsourcing systems," in *Proceedings of the First International Workshop on Crowdsourcing and Data Mining*, 2012, pp. 22-26.

[10] D. E. Difallah, G. Demartini, and P. Cudr-Mauroux, "Pick-a-crowd: tell me what you like, and i'll tell you what to do," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 367-374.

[11] D. He, S. Chan, and M. Guizani, "User privacy and data trustworthiness in mobile crowd sensing," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 28-34, 2015.

[12] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos, "TRAC: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in *Proceedings of IEEE INFOCOM 2014*, 2014, pp. 1231-1239.

[13] Z. Yan, W. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," *Future Generation Computer Systems*, vol. 62, pp. 175-189, 2016.

[14] Z. Yan, P. Zhang, and A. V. Vasilakos, "A security and trust framework for virtualized networks and software-defined networking," *Security and communication networks*, vol. 9, no. 16, pp. 3059-3069, 2016.

[15] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: challenges, *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26-33, 2017.

[16] N. Xiong, A. V. Vasilakos, L. T. Yang, L. Song, Y. Pan, R. Kannan, and Y. Li, "Comparative analysis of quality of service and memory usage for adaptive failure detectors in healthcare systems," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, pp. 495-509, 2009.

[17] K. Yang, K. Zhang, J. Ren, and X. Shen, "Security and privacy in mobile crowdsourcing networks: challenges and opportunities," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 75-81, 2015.

[18] Y. Gong, Y. Guo, and Y. Fang, "A privacy-preserving task recommendation framework for mobile crowdsourcing," in *Proceedings of 2014 IEEE Global Communications Conference*, 2014, pp. 588-593.

[19] J. Shu and X. Jia, "Secure Task Recommendation in Crowdsourcing," in *Proceedings of GLOBECOM 2016*, 2016, pp. 1-6.

[20] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of IEEE S&P 2000*, 2000, pp. 588-593.

[21] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proceedings of IEEE INFOCOM 2010*, 2010, pp. 1-5.

[22] C. Wang, N. Cao, J. Li, K. Ren, and W. J. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," in *Proceedings of IEEE ICDCS 2010*, 2010, pp. 253-262.

[23] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222-233, 2014.

[24] Z. Shen, J. Shu, and W. Xue, "Preferred keyword search over encrypted data in cloud computing," in *Proceedings of 21st International Symposium on Quality of Service*, 2013, pp. 1-6.

[25] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C. N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Services Computing*, 2016.

[26] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of Advances in Cryptology-Eurocrypt 2004*, 2004, pp. 506-522.

[27] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895-934, 2011.

[28] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient Encrypted Keyword Search for Multi-user Data Sharing," in *Proceedings of European Symposium on Research in Computer Security*, 2016, pp. 173-195.

[29] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Proceedings of Annual International Cryptology Conference*, 2005, pp. 258-275.

[30] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Proceedings of Annual International Cryptology Conference*, 2006, pp. 290-307.

[31] R. A. Popa and N. Zeldovich, "Multi-Key Searchable Encryption," *IACR Cryptology ePrint Archive*, Report 2013/508, 2013.

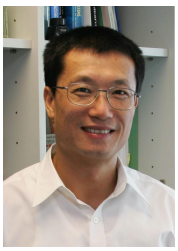
[32] J. Yang, C. Fu, N. Shen, Z. Liu, C. Jia, and J. Li, "General Multi-key Searchable Encryption," in *Proceedings of 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 2015, pp. 89-95.

[33] Q. Tang, "Nothing is for free: Security in searching shared and encrypted data," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1943-1952, 2014.

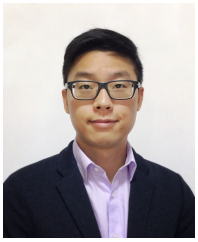
[34] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2374-2385, 2016.

[35] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proceedins of 2013 ACM SIGSAC conference on Computer and communications security*, 2013, pp. 875-888.
[36] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proceedings of Advances in Cryptology-CRYPTO 2013*, 2013, pp. 353-373.
[37] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *Proceedings of European Symposium on Research in Computer Security*, 2015, pp. 123-145.
[38] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *Journal of Computer Security*, vol. 19, no. 3, pp. 367-397, 2011.
[39] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proceedings of International Conference on Information Security Practice and Experience*, 2008, pp. 71-85.
[40] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1566-1577, 2016.
[41] D. E. Difallah, M. Catasta, G. Demartini, P. G. Ipeirotis, and P. Cudr-Mauroux, "The dynamics of micro-task crowdsourcing: The case of amazon mturk," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 238-247.
[42] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proceedings of 2013 IEEE 29th International Conference on Data Engineering*, 2013, pp. 733-744.
[43] S. T. Alexander and V. L. Stonick, "Fast adaptive polynomial root tracking using a homotopy continuation method," in *Proceedings of 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1993, pp. 480-483.
[44] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of 2011 IEEE Symposium on In Computers and communications*, 2011, pp. 850-855.

Jiangang Shu received the BE degree and the MS degree from Nanjing University of Information Science and Technology, Nanjing, China, in 2012 and in 2015, respectively. Since 2015, he has been working toward the PhD degree in computer science at the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include security and privacy in crowdsourcing, cloud computing security, and steganography. He is a student member of the IEEE.



Xiaohua Jia received his BSc (1984) and MEng (1987) from University of Science and Technology of China, and DSc (1991) in Information Science from University of Tokyo. He is currently Chair Professor with Dept of Computer Science at City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks and mobile computing. Prof. Jia is an editor of IEEE Internet of Things, IEEE Transactions on Parallel and Distributed Systems (2006-2009), Wireless Networks, Journal of World Wide Web, Journal of Combinatorial Optimization, etc. He is the General Chair of ACM MobiHoc 2008, TPC Co Chair of IEEE GlobeCom 2010 Ad Hoc and Sensor Networking Symposium, Area-Chair of IEEE INFOCOM 2010 and 2015. He is Fellow of IEEE.



Kan Yang received the B.Eng. degree in information security from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, China, in August 2013. He is currently a tenure-track assistant professor in the Dept. of Computer Science at the University of Memphis, Memphis, TN, USA. His research interests include security and privacy issues in cloud computing, big data, crowdsourcing, and Internet of Things, applied cryptography, wireless communication and networks, and distributed systems.

Hua Wang received the Ph.D. degree from the University of Southern Queensland, Australia. He is a full-time Professor with Victoria University, Victoria, Australia. He was a Professor with the University of Southern Queensland before he joined Victoria University. He has more than ten years of teaching and working experience in applied informatics in both industry and academia. He has expertise in electronic commerce, business process modeling and enterprise architecture. As an Chief Investigator, three Australian Research Council (ARC) Discovery grants have been awarded since 2006, and 155 peer-reviewed scholar papers have been published. Six Ph.D. students have already graduated under his principal supervision.