8-2018

Anonymous privacy-preserving task matching in crowdsourcing

Jiangang SHU

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Xiaohua JIA

Kan YANG

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/JIOT.2018.2830784

Published in IEEE Internet of Things Journal, August 2018, Volume 5, Issue 4, pp. 3068-3078.
https://doi.org/10.1109/JIOT.2018.2830784

Anonymous Privacy-Preserving Task Matching in Crowdsourcing

Jiangang Shu, *Graduate Student Member, IEEE*, Ximeng Liu, *Member, IEEE*, Xiaohua Jia, *Fellow, IEEE*, Kan Yang, *Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*—With the development of sharing economy, crowdsourcing as a distributed computing paradigm has become increasingly pervasive. As one of indispensable services for most crowdsourcing applications, task matching has also been extensively explored. However, privacy issues are usually ignored during the task matching and few existing privacy-preserving crowdsourcing mechanisms can simultaneously protect both task privacy and worker privacy. This paper systematically analyzes the privacy leaks and potential threats in the task matching and proposes a single-keyword task matching scheme for the multirequester/multiworker crowdsourcing with efficient worker revocation. The proposed scheme not only protects data confidentiality and identity anonymity against the crowd-server, but also achieves query traceability against dishonest or revoked workers. Detailed privacy analysis and thorough performance evaluation show that the proposed scheme is secure and feasible.**

***Index Terms*—Anonymity, crowdsourcing, privacy, revocation, task matching, traceability.**

I. Introduction

CROWDSOURCING [1] has emerged as an effective way to deal with complex tasks that require human intelligence or machine computation. Many Web-based and mobile-based crowdsourcing platforms, e.g., Amazon Mechanical MTurk,[1] CrowdFlower,[2] and TaskRabbit,[3] have been established for a vast number of tasks ranging from house improvement to text translation. In such a crowdsourcing platform, *task requesters* can publish tasks to the platform (*crowd-server*) and *task workers* can query the tasks of their interests.

Manuscript received March 16, 2018; revised April 18, 2018; accepted April 24, 2018. Date of publication April 27, 2018; date of current version August 9, 2018. This work was supported in part by the Research Grants Council of Hong Kong under Grant GRF CityU 11208917 and Grant CRF CityU C1008-16G and in part by the NSF China under Grant 61732022 and Grant 61702105. *(Corresponding author: Xiaohua Jia.)*

J. Shu and X. Jia are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: jgshu2-c@my.cityu.edu.hk; csjia@cityu.edu.hk).

X. Liu and R. H. Deng are with the School of Information Systems, Singapore Management University, Singapore (e-mail: snbnix@gmail.com; robertdeng@smu.edu.sg).

K. Yang is with the Department of Computer Science, University of Memphis, Memphis, TN 38152 USA (e-mail: kan.yang@memphis.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. This includes definitions, theorems, and Proofs related to the main paper. This material is 0.169 MB in size.

[1] [Online]. Available: https://www.mturk.com/mturk/welcome

[2] [Online]. Available: https://www.crowdflower.com

[3] [Online]. Available: https://www.taskrabbit.com

As a key service of crowdsourcing, task matching has attracted a lot of attention from both research community and industry. In the current solutions [2]–[4], the crowd-server performs accurate task-worker matching based on task requirements specified by requesters and queries submitted by workers. Since the requirements and queries usually contain sensitive information and yet the crowd-server is not fully trusted, such solutions will inevitably disclose the sensitive information of tasks and workers to the crowd-server. Existing privacy-preserving mechanisms, especially in spatial crowdsourcing, only preserve worker information but ignore the protection of task information [5]–[7]. The crowd-server can infer the workers' information by combining the task-worker matching result with the task information, and thus these one-sided mechanisms cannot fully preserve the worker privacy in the end. Therefore, it is necessary to protect both task privacy and worker privacy against the crowd-server during the task matching.

Encryption-before-outsourcing is a fundamental method to protect the privacy. Searchable encryption (SE) is an important technique that seems to provide a good solution for the task-worker matching over the encrypted data in crowdsourcing. Most of SE schemes [10]–[18] only allow the queries from a single user holding the secret key. However, there are multiple requesters and multiple workers in crowdsourcing. It is infeasible to let all the users (requesters and workers) share the same secret key, as every user revocation will incur the update of the stored encrypted data and the key redistribution to all the non-revoked users. And meanwhile, user accountability cannot be achieved in a provable manner when the secret key is leaked. It does not work either to simply let each worker have its own secret key and share this key with all the requesters. To make published tasks searchable by all the workers, in this case a requester has to encrypt a task with each worker's key and submit multiple copies of encrypted tasks to the crowd-server. This will incur a huge amount of computation and transmission overhead. Therefore, the single-user SE cannot be directly applied in the multiuser task matching in crowdsourcing.

Proxy re-encryption is a major technique to achieve multiuser SE [8], [9], [24], [25]. However, in these proxy-based solutions, users' identities need to be explicitly transmitted to the server together with the encrypted data for server-side re-encryption, which will lead to the identity leakage. Another alternative solution is to utilize broadcast encryption to generate a distinct secret key for each user, and thus every user can query the encrypted data with its own

key [28]. But, since the user secret keys are all derived from a common master secret key, user revocation will incur a high overhead for recomputing and redistribution of the new keys. It is difficult to design a privacy-preserving task matching scheme that can simultaneously achieve identity anonymity and efficient revocation.

In this paper, we design an anonymous privacy-preserving task matching scheme with efficient worker revocation in the multirequester/multiworker crowdsourcing systems. Our scheme not only protects data confidentiality and identity anonymity against the crowd-server, but also achieves traceability against the dishonest workers and revoked workers. We also analyze its security and performance through detailed security analysis and performance evaluation, and the results show that our scheme is secure and feasible. The main contributions of this paper can be summarized as follows.

1) This paper systematically analyzes the privacy leaks and potential threats in the task matching for crowdsourcing and defines a set of privacy requirements against the crowd-server, dishonest workers, and revoked workers.
2) Compared with the proxy-based solutions [8], [9], [24], [25], the proposed scheme achieves the task matching without leaking identity privacy.
3) Compared with the broadcast-based solutions [28], the proposed scheme supports efficient worker revocation with minimal overhead on the crowd-server, and meanwhile without recomputing and redistributing new keys to the nonrevoked workers.

The rest of this paper is organized as follows. Section II presents the related works and Section III provides the preliminaries. In Section IV, we formulate the problem of crowdsourcing task matching with the system model, threat model, privacy requirements, and design goals. In Section V, we describe the detailed construction of a single-keyword task matching scheme and also analyze its security. In Section VI, we implement the proposed scheme and conduct a detailed performance analysis in comparison with the state-of-the-art works theoretically and experimentally. Finally, we conclude this paper in Section VII.

II. Related Work

There are three research topics involved in this paper: 1) task matching in crowdsourcing; 2) SE; and 3) group signature.

A. Task Matching in Crowdsourcing

Task matching has been extensively studied since the introduction of crowdsourcing [1]. Crowdsourcing tasks are usually assigned to workers based on their interests [2], search histories [3], or social profiles [4]. However, privacy issue during task matching was ignored by these works. Considering the crowd-server being untrusted, worker location privacy was first considered in spatial crowdsourcing. To *et al.* [5] introduced a trusted third party to protect worker location privacy based on differential privacy. Shen *et al.* [6] designed a secure task assignment protocol using additive homomorphic encryption with the introduction of a semi-honest third party. Extending spatial crowdsourcing to mobile crowdsourcing, Gong *et al.* [7] considered worker context privacy (e.g., activity) and proposed a flexible framework to optimize the tradeoffs among utility, privacy, and efficiency. All these works rely on a trusted or semi-trusted proxy to collect and process the sensitive worker information, and moreover they only consider worker privacy. For these reasons, we first considered both worker privacy and task privacy simultaneously, and utilized proxy re-encryption to achieve the single-keyword and multikeyword matching [8], [9], without any interactive help from the proxy during the task matching. However, due to the adoption of proxy re-encryption, our previous schemes cannot protect workers' identity privacy.

B. Searchable Encryption

SE is a potential way to achieve the encrypted task-worker matching. According to its application scenarios, the model of SE can be generally classified into four categories: 1) single-owner/single-user; 2) multiowner/single-user; 3) single-owner/multiuser; and 4) multiowner/multiuser.

In the single-owner/single-user model, both searchable ciphertexts and trapdoors are created by a single data owner (also as user) with a secret key. Since the first searchable symmetric encryption (SSE) [10], fuzzy query [11], ranked query [12], personalized query [13], Boolean query [14], and pattern query [15] have been extensively explored in this model.

In the multiowner/single-user model, multiple data owners can create searchable ciphertexts using a common public key while only a private key holder (as user) is allowed to query. Since the first pubic key encryption with keyword search (PEKS) [16], conjunctive, subset, and range query [17], [18] have also been extensively studied in this model.

In the single-owner/multiuser model, only a secret key owner can create searchable ciphertexts, whereas a group of users authorized by the owner are allowed to query. To achieve this goal, Curtmola *et al.* [19] first proposed a general construction which combines broadcast encryption with a single-owner/single-user SE scheme. Inspired by oblivious cross-tag [20], Jarecki *et al.* [21] proposed a multiclient SSE and Faber *et al.* [22] further realized rich queries. To remove the per-query interaction between the owner and the user in [21], Sun *et al.* [23] proposed a noninteractive multiclient SSE. If we introduce a single-owner/multiuser SE scheme directly into the multirequester/multiworker crowdsourcing, the system usability will be largely affected, as multiple trapdoors need to be generated for a query.

In the multiowner/multiuser model, every user is allowed to search over encrypted data outsourced by multiple owners. Proxy re-encryption is first introduced by Dong *et al.* [24] and Bao *et al.* [25] to achieve this goal. However, these proxy re-encryption-based SE schemes cannot protect identity privacy. Moreover, collusion attack vulnerability and interaction cost make them not applicable to the crowdsourcing-based task matching. Zhang *et al.* [26] introduced an extra third party as middleware for transforming indexes and trapdoors before outsourcing, which is not feasible to be deployed in crowdsourcing. Besides the above conventional multiowner/multiuser SE schemes, attribute-based keyword search (ABKS) [27] and identity-based keyword search (IBKS) [28] have also been

studied. In ABKS or IBKS, if and only if a user' attributes or identity satisfy a file's access policy specified by the data owner, the user will gain the search authorization to that file. Such mechanisms based on owner-enforced search authorization are not applicable to crowdsourcing. Recently, policy-hiding ABE [29], [30] has been proposed to achieve the anonymity of policy during the decryption on the user side, which is different from our goal to achieve the query anonymity during the task matching on the crowd-server side.

C. Group Signature

Group signature, introduced by Chaum and Van Heyst [31], allows each member of a group anonymously sign messages on behalf of the group. Due to its anonymity, traceability, and nonframeability features, group signatures have been applied in anonymous online communications, anonymous credential systems, and digit rights management. Membership revocation is an important research topic for dynamic groups, and there are generally three different revocation mechanisms. The first mechanism enables the group manager to update the group public key, recompute, and reissue new private signing keys to all the remaining members after each revocation. It will put a heavy burden on the group manager. An alternate mechanism is called verifier-local revocation [32], where revocation messages are only sent to signature verifiers. In this mechanism, the verifiers need to check whether a candidate signature is generated by a revoked member before signature verification. Since the complexity of revocation checking is linear to the number of revoked members, it is quite inefficient and will add computation overheads on the verifiers. The last mechanism is to enable the group manager to update the group public key and the signers to update their private signing keys by themselves [33]. This mechanism keeps constant the verifiers's computation costs.

III. Preliminaries

A. Bilinear Pairings

$\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are all multiplicative cyclic groups of prime order p. Let g_i denotes a generator of $\mathbb{G}_i$. A bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ has the following properties.

1) *Bilinearity:* For all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$.
2) *Nondegeneracy:* $e(g_1, g_2) \neq 1$.
3) *Computability:* It is efficient to compute e for any input.

In bilinear pairings, there are three basic variants as follows.

1) *Type 1:* $\mathbb{G}_1 = \mathbb{G}_2$.
2) *Type 2:* $\mathbb{G}_1 \neq \mathbb{G}_2$, and there is a computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(g_2) \rightarrow g_1$.
3) *Type 3:* $\mathbb{G}_1 \neq \mathbb{G}_2$, and there is no isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$.

B. Complexity Assumptions

1) Bilinear Diffie–Hellman Problem: The bilinear Diffie–Hellman (BDH) problem in $\mathbb{G}_1$ is stated as follows: given (g, g^a, g^b, g^c) as input, output $e(g, g)^{abc} \in \mathbb{G}_T$, where $g \in \mathbb{G}_1, a, b, c \in \mathbb{Z}_p^*$ chosen uniformly and independently at random. We say that the BDH assumption holds if all probabilistic polynomial time algorithms have a negligible advantage to solve the BDH problem.

2) q-Strong Diffie–Hellman Problem: The q-Strong Diffie–Hellman (q-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is stated as follows: given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, \ldots, g_2^{\gamma^q})$ as input, output a pair $(g_1^{[1/(\gamma+x)]}, x)$, where $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, q, \gamma, x \in \mathbb{Z}_p^*$ chosen uniformly and independently at random. We say that the q-SDH assumption holds if all probabilistic polynomial time algorithms have a negligible advantage to solve the q-SDH problem.

3) Decisional Linear Problem: The decisional linear (DLIN) problem in $\mathbb{G}_1$ is stated as follows: given (u, v, h, u^a, v^b), distinguish h^{c+d} from a random element, where $u, v, h \in \mathbb{G}_1, a, b \in \mathbb{Z}_p^*$ chosen uniformly and independently at random. The DLIN assumption holds if all probabilistic polynomial time algorithms have a negligible advantage to solve the DLIN problem.

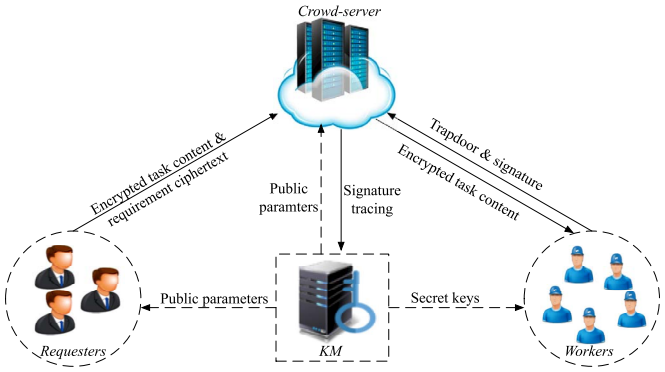


Fig. 1. System model.

C. Linear Encryption

Linear encryption is an extension of ElGamal encryption. It is proved to be semantically secure against chosen-plaintext attack, assuming the DLIN assumption holds. In the linear encryption, the public key is $u, v, h \in \mathbb{G}_1$ and the private key is $x, y \in \mathbb{Z}_p^*$ such that $u^x = v^y = h$. Its encryption and description processes are stated as follows.

1) To encrypt a message $m \in \mathbb{G}_1$, randomly choose $a, b \in \mathbb{Z}_p^*$, output $(u^a, v^b, m \cdot h^{a+b})$.
2) To recover the message from a ciphertext (C_1, C_2, C_3), compute $C_3/(C_1^x \cdot C_2^y)$.

IV. Problem Formulation

A. System Model

We consider a dynamic crowdsourcing system, where any task requester can publish its encrypted tasks on an untrusted crowdsourcing server such that only authenticated task workers can search over the tasks of their interests. In the system, the workers may join and leave the system dynamically. For a revoked worker, it will no longer have permission to query the tasks.

As shown in Fig. 1, there are four entities in the crowdsourcing system: 1) a key manager (KM); 2) a

crowdsourcing service provider (crowd-server); 3) multiple requesters; and 4) multiple workers. The KM is in charge of system initialization, worker enrollment, and revocation. Initially, the KM setups the system to publicize public parameters and assign a distinct secret key to each participating worker. When publishing a task, a requester encrypts the requirement for the task, and then publishes the requirement ciphertext to the crowd-server, together with the task content in encryption form. To retrieve the tasks of its interest, a worker generates the trapdoor and the signature on a query using its secret key, and submits them to the crowd-server. When receiving the query request from a worker, the crowd-server authenticates the worker and sends the matched tasks to the worker by matching the requirements with the trapdoor. After that, the worker can decrypt the task contents and carry out them. Note that the encryption and decryption of task content is out of scope of this paper. The KM can also trace back the identities from the suspicious signatures.

B. Threat Model and Privacy Requirements

In the above system model, we consider the KM and all the requesters as honest (the requesters can be guaranteed by payments).

Crowd-server is assumed as "honest-but-curious," i.e., it will honestly execute the protocol but it is curious to obtain the private and sensitive information from the ciphertexts, trapdoors, and signatures. Besides, worker identities and query frequencies are also very sensitive, i.e., the crowd-server can launch linking attacks to decide whether any two queries (including trapdoors and signatures) come from a same worker. If the linking attacks are successful, the crowd-server can obtain the query frequency of each worker and easily identify the worker identities if with some external knowledge. The crowd-server is usually a reputable company in practice and is assumed to not collude with the requesters and workers, which is a reasonable assumption as [24] and [25].

Workers are not always fully trusted. They can be categorized into two classes.

1) *Dishonest worker* is a legitimate worker in the system but may be dishonest in the sense that it may leak its secret key to other illegitimate (outside) workers to make profit.
2) *Revoked worker* was a legitimate worker but now it has no permission to search over the encrypted tasks on the crowd-server. After revocation, it may forge the current legitimate workers to send the queries to the crowd-server.

Based on the above adversaries, we set the following privacy requirements.

1) *Confidentiality:* Ciphertexts and trapdoors should be protected from the crowd-server.
2) *Anonymity:* Given the queries from the legitimate workers, the crowd-server and other inside or outside workers cannot discover their identities, and decide whether any two queries come from a same worker.
3) *Traceability:* Underlying identities of the queries can always be recognized by the KM. It includes unforgeability that queries from a legitimate worker cannot be forged by any outside worker, and revocability that revoked workers no longer have permissions to query.

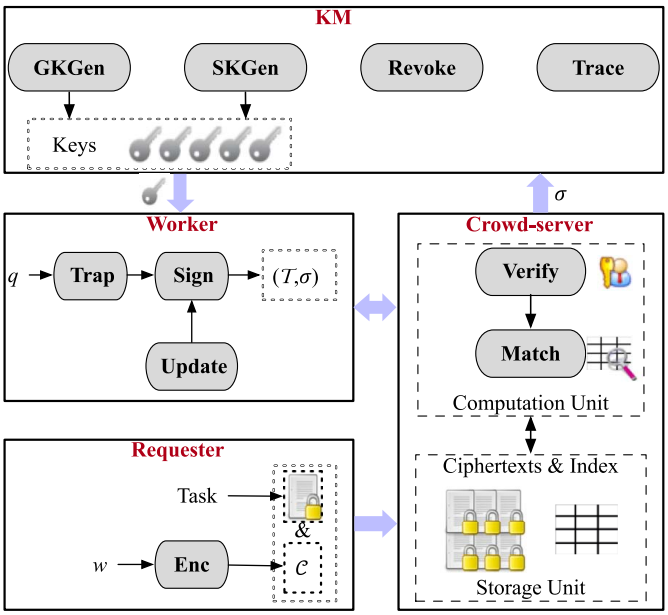


Fig. 2. Work flow.

C. Design Goals

Besides the aforementioned privacy requirements, the proposed scheme shall also meet the following utility goals.

1) *Scalability:* To fit the multirequester/multiworker crowdsourcing, it should have constant-size secret keys, ciphertexts and trapdoors. Specifically, their sizes should be all independent of the number of requesters and workers in the system.
2) *Efficient Revocation:* The proposed scheme should support the efficient worker revocation with minimal overheads on the crowd-server and the remaining nonrevoked workers.

V. Scheme Construction

In this section, we construct a single-keyword task matching scheme based on short group signature [33] and PEKS [16]. In the short group signature, we consider three bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p with a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, and use a hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$. In the SE, we consider two bilinear groups $\mathbb{G}$ and $\mathbb{G}_{T'}$ of prime order p' with a mapping $e' : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{T'}$ and employ two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G}_{T'} \to \{0,1\}^*$. As shown in Fig. 2, the work flow of the scheme proceeds as follows.

1) *System Initialization:* The KM initializes the system and assigns keys for requesters and registered workers.
2) *Task Matching:* Given the encrypted task requirements and trapdoors from requesters and workers, respectively, the crowd-server performs the task matching process.

Algorithm 1 Group Key Generation

procedure GKGen($1^\lambda, n$)
  Select $g_2 \in_R \mathbb{G}_2$
  $g_1 \leftarrow \psi(g_2)$
  Select $h \in_R \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$, $\alpha, \beta, \gamma \in_R \mathbb{Z}_p^*$
  Set $u, v \in \mathbb{G}_1$ such that $u^\alpha = v^\beta = h$
  $w \leftarrow g_2^\gamma$
  for $i = 1$ to n **do**
    Select $x_i \in_R \mathbb{Z}_p^*$
    Compute y_i
    $gsk[i] \leftarrow (x_i, y_i)$
  end for
  $RL \leftarrow \{\}$, $X \leftarrow 1$
  $gpk \leftarrow (g_1, g_2, h, u, v, w)$
  $gmsk \leftarrow (\alpha, \beta, \gamma, X)$
  return $(gpk, gmsk, gsk[1{:}n], RL)$
end procedure

3) *Query Verification and Tracing:* Workers sign their queries and submit the signed queries to the crowd-server. The crowd-server verifies the validity of received queries and traces the underlying identities of suspicious queries in cooperation with the KM.
4) *Worker Revocation:* The KM revokes the leaving workers, and the remaining nonrevoked workers independently update their private keys.

A. System Initialization

The KM sets up the system by calling the **GKGen** and **SKGen** algorithms to generate the parameters for group signature and SE, respectively.

GKGen$(1^\lambda, n) \rightarrow (gpk, gmsk, gsk[1{:}n], RL)$. Taking as input the security parameter λ and the maximum number of workers n, it performs as Algorithm 1 to output the group public key $gpk = (g_1, g_2, h, u, v, w)$, where $u^\alpha = v^\beta = h$, the group master secret key $gmsk = (\alpha, \beta, \gamma, X)$, n group private signing keys $gsk[1{:}n] = \{(x_i, y_i = g_1^{1/(\gamma + x_i)})\}$, and an empty revocation list RL.

SKGen$(1^\lambda) \rightarrow (spk, ssk)$. Taking as input the security parameter λ, it selects $\delta \in_R \mathbb{Z}_{p'}^*$ and a generator $g \in_R \mathbb{G}$. It outputs the searchable public key $spk = (g, g^\delta)$ and the searchable secret key $ssk = \delta$.

After that, the KM transmits $(gsk[i], ssk)$ to the worker i for each $i \in [1, n]$. It also publicizes the group public key gpk, the searchable public key spk and the revocation list RL while keeping the group master secret key $gmsk$.

B. Task Matching

With the public parameters, the requester encrypts a requirement keyword w by calling the **Enc** algorithm and submits the ciphertext $\mathcal{C}$ of w to the crowd-server. The crowd-server can build a task index table to store the requirement ciphertexts submitted by the requesters. To retrieve the tasks matching with some keyword q, the worker generates the trapdoor $\mathcal{T}$ from q by calling the **Trap** algorithm with its secret key and submits $\mathcal{T}$ to the crowd-server. Given the trapdoor $\mathcal{T}$, the crowd-server can match it with the ciphertexts in the index table by performing the **Match** algorithm.

Enc$(spk, w) \rightarrow \mathcal{C}$. Given a requirement keyword w, it chooses $r \in_R \mathbb{Z}_{p'}^*$ and computes $t = e'(H_1(w), (g^\delta)^r)$. It outputs $\mathcal{C} = (g^r, H_2(t))$ as the ciphertext of w.

Trap$(ssk, q) \rightarrow \mathcal{T}$. Given a query keyword q, it computes the trapdoor $\mathcal{T} = H_1(q)^{ssk}$.

Match$(spk, \mathcal{C}, \mathcal{T}) \rightarrow 1/0$. Let $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$, where $\mathcal{C}_1 = g^r$ and $\mathcal{C}_2 = H_2(t)$, if $H_2(e'(\mathcal{T}, \mathcal{C}_1)) = \mathcal{C}_2$, it outputs 1; otherwise, it outputs 0.

Theorem 1 (Matching Correctness): The task matching process is correct. That is, **Match**$(spk, \mathcal{C}, \mathcal{T}) = 1$ iff $q = w$.

Proof: Suppose **Match**$(spk, \mathcal{C}, \mathcal{T}) = 1$, we have

$$\begin{aligned} & H_2\big(e'(\mathcal{T}, \mathcal{C}_1)\big) = \mathcal{C}_2 \\ \Leftrightarrow\ & H_2\big(e'(\mathcal{T}, \mathcal{C}_1)\big) = H_2\Big(e'\Big(H_1(w), \big(g^\delta\big)^r\Big)\Big). \end{aligned}$$

Since H_2 is a hash funcation, we have

$$\begin{aligned} & e'(\mathcal{T}, \mathcal{C}_1)) = e'\Big(H_1(w), \big(g^\delta\big)^r\Big) \\ \Leftrightarrow\ & e'\Big(H_1(q)^{ssk}, g^r\Big)\Big) = e'\Big(H_1(w), \big(g^\delta\big)^r\Big) \\ \Leftrightarrow\ & e'(H_1(q), g)^{\delta r} = e'(H_1(w), g)^{\delta r} \\ \Rightarrow\ & H_1(q) = H_1(w). \end{aligned}$$

Since H_1 is a hash function, we have $q = w$. This completes the proof. ■

Algorithm 2 Sign a Trapdoor

procedure Sign($gpk, gsk[i], \mathcal{T}$)
  Select $a, b \in_R \mathbb{Z}_p^*$
  Compute A_1, A_2, A_3
  Compute θ_1, θ_2
  Select $r_a, r_b, r_x, r_{\theta_1}, r_{\theta_2} \in_R \mathbb{Z}_p^*$
  Compute R_1, R_2, R_3, R_4, R_5
  Compute c
  Compute $s_a, s_b, s_x, s_{\theta_1}, s_{\theta_2}$
  return $\sigma \leftarrow (A_1, A_2, A_3, c, s_a, s_b, s_x, s_{\theta_1}, s_{\theta_2})$
end procedure

C. Query Verification and Tracing

To prevent outside illegitimate workers from requesting the query services, the crowd-server needs to verify the validity of the received queries. Only when the queries are from the legitimate workers, the crowd-server will conduct the above task matching process. With utilizing group signature, the worker generates the signature σ of the trapdoor $\mathcal{T}$ by calling **Sign** with its own group private signing key $gsk[i]$ and submits $(\mathcal{T}, \sigma)$ to the crowd-server. Given $(\mathcal{T}, \sigma)$, the crowd-server can authenticate σ with the **Verify** algorithm.

Sign$(gpk, gsk[i], \mathcal{T}) \rightarrow \sigma$. Taking the trapdoor $\mathcal{T}$ as input, it outputs the signature σ of $\mathcal{T}$ as Algorithm 2,

Algorithm 3 Verify Signature of a Trapdoor

```
procedure Verify(gpk, T, σ)
    Compute R̃1, R̃2, R̃3, R̃4, R̃5
    Compute c' ← H(T, A1, A2, A3, R̃1, R̃2, R̃3, R̃4, R̃5)
    if c' = c then
        return 1
    else
        return 0
    end if
end procedure
```

denoted as $(A_1, A_2, A_3, c, s_a, s_b, s_x, s_{\theta_1}, s_{\theta_2})$

$$
\begin{aligned}
A_1 &= u^a \qquad A_2 = v^b \qquad A_3 = y_i h^{a+b} \\
c &= H(\mathcal{T}, A_1, A_2, A_3, R_1, R_2, R_3, R_4, R_5) \\
s_a &= r_a + ca \qquad s_b = r_b + cb \qquad s_x = r_x + cx_i \\
s_{\theta_1} &= r_{\theta 1} + c\theta_1 \qquad s_{\theta_2} = r_{\theta 2} + c\theta_2
\end{aligned}
$$

where $\theta_1 = x_i a$, $\theta_2 = x_i b$, $R_1 = u^{r_a}$, $R_2 = v^{r_b}$, $R_3 = e(A_3, g_2)^{r_x} e(h, w)^{-r_a - r_b} e(h, g_2)^{-r_{\theta_1} - r_{\theta_2}}$, $R_4 = A_1^{r_x} \cdot u^{-r_{\theta_1}}$, $R_5 = A_2^{r_x} \cdot v^{-r_{\theta_2}}$, and a, b, r_a, r_b, r_x, r_{θ_1}, $r_{\theta_2} \in_R Z_p^*$.

Verify$(gpk, \mathcal{T}, \sigma) \rightarrow 1/0$. Given the trapdoor $\mathcal{T}$ and the signature σ, it verifies the validity of σ as Algorithm 3. It first computes $(\tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4, \tilde{R}_5)$ as

$$
\begin{aligned}
\tilde{R}_1 &= u^{s_a} \cdot A_1^{-c} \qquad \tilde{R}_2 = v^{s_b} \cdot A_2^{-c} \\
\tilde{R}_3 &= e(A_3, g_2)^{s_x} \cdot e(h, w)^{-s_a - s_b} \cdot e(h, g_2)^{-s_{\theta_1} - s_{\theta_2}} \\
&\quad \times (e(A_3, w)/e(g_1, g_2))^c \\
\tilde{R}_4 &= A_1^{s_x} \cdot u^{-s_{\theta_1}} \qquad \tilde{R}_5 \leftarrow A_2^{s_x} \cdot v^{-s_{\theta_2}}.
\end{aligned}
$$

Then it computes

$$
c' = H\big(\mathcal{T}, A_1, A_2, A_3, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4, \tilde{R}_5\big)
$$

and checks whether $c' \stackrel{?}{=} c$. If $c' = c$, σ is valid.

If the crowd-server finds a dishonest worker by detecting abnormal operations (e.g., frequent queries by a worker), it will transmit the received suspicious queries to the KM which can trace back its identity by calling the **Trace** algorithm.

Trace$(gmsk, \mathcal{T}, \sigma) \rightarrow i/0$. When σ is valid, it computes $y_i = A_3/(A_1^{\alpha} A_2^{\beta})$ and finds the worker identity i who has y_i. If no such i exists, it outputs 0.

Theorem 2 (Signature Correctness): The group signature is correct. That is, **Verify**$(gpk, \mathcal{T}, \textbf{Sign}(gpk, gsk[i], \mathcal{T})) = 1$ and **Trace**$(gmsk, \mathcal{T}, \textbf{Sign}(gpk, gsk[i], \mathcal{T})) = i$.

Proof: Suppose $\sigma = (A_1, A_2, A_3, c, s_a, s_b, s_x, s_{\theta_1}, s_{\theta_2})$ is a valid signature, we have the following five equations:

$$
\begin{aligned}
\tilde{R}_1 &= u^{s_a} \cdot A_1^{-c} = u^{r_a} = R_1 \\
\tilde{R}_2 &= v^{s_b} \cdot A_2^{-c} = u^{r_b} = R_2 \\
\tilde{R}_3 &= e(g_1, g_2)^{\frac{r_x}{(\gamma + x_i)}} e(h, g_2)^{r_x(a+b) - \gamma(r_a + r_b) - r_{\theta_1} - r_{\theta_2}} = R_3 \\
\tilde{R}_4 &= A_1^{s_x} \cdot u^{-s_{\theta_1}} = u^{a r_x - r_{\theta_1}} = R_4 \\
\tilde{R}_5 &= A_2^{s_x} \cdot u^{-s_{\theta_2}} = v^{b r_x - r_{\theta_2}} = R_5.
\end{aligned}
$$

Algorithm 4 Worker Revocation

```
procedure Revoke(gpk, gmsk, RL, I)
    for i ∈ I do
        Compute y_i*
        grk[i] ← (x_i, y_i*)          ▷ Set group revocation key
    end for
    Compute X̂
    Compute ĝ1, ĝ2, ŵ
    gpk ← (ĝ1, ĝ2, h, u, v, ŵ)        ▷ Update gpk
    gmsk ← (α, β, γ, X̂)               ▷ Update gmsk
    RL ← RL ∪ {grk[i]}_{i∈I}          ▷ Update RL
    return (gpk, gmsk, RL)
end procedure
```

Then we have

$$
\begin{aligned}
\big(\tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4, \tilde{R}_5\big) &= (R_1, R_2, R_3, R_4, R_5) \\
&\Leftrightarrow c' = c \\
\Leftrightarrow \textbf{Verify}(gpk, \mathcal{T}, \sigma) &= 1.
\end{aligned}
$$

Moreover, the first three components of any valid signature σ, $(A_1, A_2, A_3) = (u^a, v^b, y_i \cdot h^{a+b})$, form a linear encryption of y_i under the public key (u, v, h). Since the KM possesses the private keys (α, β), it can always trace the identities from the valid signatures. ■

D. Worker Revocation

To prevent a revoked worker from requesting the query services, we need to revoke its query permission by making its secret keys invalid. Considering the efficiency on the crowd-server, we design a worker revocation mechanism that can keep the verification cost on the crowd-server a constant. In this mechanism, the KM updates the group public key *gpk* and the revocation list RL by **Revoke** when the worker revocation occurs, and the remaining nonrevoked workers can independently update their private signing keys with the updated RL by **Update**. In this way, the crowd-server can verify the signatures using the updated *gpk* without any extra computation overhead. Considering the general case that $r \in [1, n]$ workers leave the system at the same time, the proposed worker revocation mechanism can revoke r workers at once.

Revoke$(gpk, gmsk, RL, I) \rightarrow (gpk, gmsk, RL)$. Given the leaving worker identities $I = \{1, \ldots, r\}$, the KM performs as Algorithm 4 to update the group public key *gpk*, the group master secret key *gmsk* and the revocation list RL as

$$
\begin{aligned}
gpk &= \big(\hat{g}_1, \hat{g}_2, h, u, v, \hat{w}\big) \\
gmsk &= \big(\alpha, \beta, \gamma, \hat{X}\big) \\
RL &= RL \cup \{grk[i]\}_{i \in I}
\end{aligned}
$$

where $\hat{X} = X \prod_{j=1}^{r} (\gamma + x_j)$, $\hat{g}_1 = g_1^{1/\hat{X}}$, $\hat{g}_2 = g_2^{1/\hat{X}}$, $\hat{w} = (\hat{g}_2)^{\gamma}$, and $grk[i] = (x_i, y_i^* = g_2^{1/X \prod_{j=1}^{i} (\gamma + x_j)})$ is the corresponding group revocation key for the worker i.

Update$(RL, gsk[i]) \rightarrow gsk[i]$. Given the revocation list RL with the newly added entries $\{grk[j]\}_{1 \le j \le r}$, a nonrevoked

Algorithm 5 Signing Key Update

procedure Update(RL, $gsk[i]$)
  for $(x_j, y_j^*) \in \{grk[j]\}_{1 \le j \le r}$, **do**
    $y_i \leftarrow \psi(y_j^*)^{1/x_i - x_j} / y_i^{1/x_i - x_j}$ ▷ Update y_i
  end for
  $\hat{y}_i \leftarrow y_i$
  $gsk[i] \leftarrow (x_i, \hat{y}_i)$ ▷ Update $gsk[i]$
  return $gsk[i]$
end procedure

worker i can independently update its private signing key as $gsk[i] = (x_i, \hat{y}_i = g_1^{1/X(\gamma+x_i)\prod_{j=1}^{r}(\gamma+x_j)})$ with Algorithm 5.

Remark 1: In practice, we need not require the renewal of private signing key immediately upon receiving every event of worker revocation, and the update process only happens when the worker logs in the system or wants to query. When the worker logs in the system, the key can be updated once on a large number of newly revoked workers such that the key update on a small number of newly revoked workers during the query will not affect the efficiency on the worker side too much.

Theorem 3 (Revocation Correctness): The worker revocation process is correct.

Proof: When revoking r workers at once, the KM outputs the group public key $(\hat{g}_1 = g_1^{1/X\prod_{j=1}^{r}(\gamma+x_j)}, \hat{g}_2 = g_2^{1/X\prod_{j=1}^{r}(\gamma+x_j)}, h, u, v, \hat{w} = g_2^{1/X})$ and the revocation list $RL = \{grk[j]\}_{1 \le j \le r}$. Given the first revocation key $(x_1, y_1^* = g_2^{1/X(\gamma+x_1)})$, the worker holding (x, y) updates

$$y = \psi(y_1^*)^{1/x-x_1} / y^{1/x-x_1} = g_1^{\frac{1}{X(\gamma+x)(\gamma+x_1)}}.$$

Then, given the second revocation key $(x_2, y_2^* = g_2^{1/X(\gamma+x_1)(\gamma+x_2)})$, the worker continues updating

$$y = \psi(y_2^*)^{1/x-x_2} / y^{1/x-x_2} = g_1^{\frac{1}{X(\gamma+x)(\gamma+x_1)(\gamma+x_2)}}.$$

After repeating r times for all the entries in $\{grk[j]\}_{1 \le j \le r}$, the worker can compute $\hat{y} = y = g_1^{1/X(\gamma+x)\prod_{j=1}^{r}(\gamma+x_j)}$. Indeed

$$\hat{y}^{\gamma+x} = g_1^{1/X\prod_{j=1}^{r}(\gamma+x_j)} = \hat{g}_1.$$

Hence, $(x, \hat{y})$ is a valid private signing key with respect to the current public key $(\hat{g}_1, \hat{g}_2, h, u, v, \hat{w})$. That means the worker revocation mechanism is correct. ■

E. Security Analysis

As defined by the privacy requirements in Section IV-B, the proposed scheme achieves *Confidentiality*, *Anonymity*, and *Traceability* as follows.

1) *Confidentiality:* The proposed scheme is IND-CKA secure in the random oracle model under the BDH assumption. Informally speaking, the adversary cannot distinguish the ciphertexts of two arbitrary keywords unless the corresponding trapdoors are available. Thus, ciphertexts and trapdoors are well protected.

TABLE I
NOTATIONS IN MSDE, MUED, SEMEKS, AND APTM

Notation	Description
E, $\mathrm{E}_{T'}$	group exponentiation on $\mathbb{G}$, $\mathbb{G}_{T'}$, respectively
E_1, E_2, E_T	group exponentiation on $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively
$\mathrm{P}, \mathrm{P}_{12}$	pairing on $<\mathbb{G}, \mathbb{G}>$, $<\mathbb{G}_1, \mathbb{G}_2>$, respectively
$\mathrm{f_s}$	key-based hash operation $\{0,1\}^* \rightarrow \mathbb{Z}_{p'}^*$
$\mathrm{h_s}$	key-based hash operation $\{0,1\}^* \rightarrow \mathbb{G}$
H	hash operation $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$
H_1	hash operation $\{0,1\}^* \rightarrow \mathbb{G}$
$\mathrm{H_g}$	hash operation $\mathbb{G} \rightarrow \{0,1\}^*$
$\mathrm{H}_{\mathrm{gt}'}$	hash operation $\mathbb{G}_{T'} \rightarrow \{0,1\}^*$
$<\mathrm{S_E}, \mathrm{S_D}>$	symmetric encryption and decryption operations
$\mathcal{M}$	plaintext size under $<\mathrm{S_E}, \mathrm{S_D}>$
$\mathcal{C}$	ciphertext size under $<\mathrm{S_E}, \mathrm{S_D}>$
n	number of workers
k	number of keywords in task requirement
k'	number of keywords in query
r	number of to-be-revoked or newly revoked workers

2) *Anonymity:* The proposed scheme achieves CPA-full anonymity in the random oracle model assuming the security of linear encryption. The definition of CPA-full anonymity also captures query unlinkability. Informally speaking, the adversary cannot distinguish the group signatures from two arbitrary workers unless it has permission to trace the signatures. Thus, worker identity privacy is well protected.
3) *Traceability:* The proposed scheme achieves insider traceablility in the random oracle model under the q-SDH assumption. Informally speaking, the adversary (except the KM) cannot create valid group signatures that are untraceable or traced back to uncorrupted workers. The definition of insider traceability also captures query unforgeability and revocability.

The detailed security definitions and proofs are given in the supplementary materials available online.[4]

F. Extension

Compared with the previous multiuser solutions [8], [9], [24], [25], [28] which only support a single type of matching functionality, e.g., the single-keyword or multikeyword matching, the proposed scheme has a good extensibility in the sense that it can be adapted to support various matching functions by combining any existing SE scheme, e.g., fuzzy keyword matching [11], Boolean matching [14], and pattern matching [15]. This extensibility is important for task matching in crowdsourcing, as the single matching functionality cannot meet the workers' growingly diversified query needs.

VI. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of the proposed scheme (denoted by APTM here) through both theoretical analysis and experimental implementation, in comparison with the state-of-the-art schemes (MSDE [24], MuED [25], and

[4]Supplemental material for the reader can be downloaded online at http://ieeexplore.ieee.org/

TABLE II
COMPARISON IN UTILITY AND SECURITY

Scheme	Anonymity	Revocation	Extensibility	Traceability	No Re-key	Anti-collusion[1]	Non-interaction
MSDE	×	√	×	√	×	×	√
MuED	×	√	×	√	×	√	×
SEMEKS	-	×	×	×	√	√	√
APTM	√	√	√	√	√	√	√

[1] It denotes that given the secret key hold by the user and the re-key hold by the server, the adversary can recover the master secret key.

TABLE III
COMPUTATION COMPLEXITY

Operation	MSDE	MuED	SEMEKS	APTM
GKGen & SKGen	neg	$n\cdot \mathrm{E}$	$8\mathrm{E}+n\cdot 16\mathrm{E}$	$\mathrm{E}+(n+2)\cdot \mathrm{E}_1+\mathrm{E}_2$
Enc	$k\cdot(4\mathrm{E}+\mathrm{f_s}+\mathrm{H_g})$	$k\cdot(\mathrm{E}+\mathrm{E}_{T'}+\mathrm{P}+\mathrm{h_s}+\mathrm{H}_{\mathrm{gt}'}+\mathrm{S_E})$	$k\cdot 9\mathrm{E}$	$k\cdot(2\mathrm{E}+\mathrm{P}+\mathrm{H}_1+\mathrm{H}_{\mathrm{gt}'})$
Trap & Sign	$k'\cdot(5\mathrm{E}+\mathrm{f_s})$	$k'\cdot(\mathrm{E}+\mathrm{h_s})$	$k'\cdot 12\mathrm{E}$	$k'\cdot(\mathrm{E}+\mathrm{H}_1)+(9\mathrm{E}_1+3\mathrm{E_T}+3\mathrm{P}_{12}+\mathrm{H})$
Match & Verify	$2k'\cdot\mathrm{E}+k\cdot k'\cdot\mathrm{H_g}$	$k'\cdot(\mathrm{P}+\mathrm{H}_{\mathrm{gt}'})+k\cdot k'\cdot\mathrm{S_D}$	$k\cdot k'\cdot 5\mathrm{P}$	$k\cdot k'\cdot(\mathrm{P}+\mathrm{H}_{\mathrm{gt}'})+(8\mathrm{E}_1+5\mathrm{E_T}+5\mathrm{P}_{12}+\mathrm{H})$
Trace	-	-	-	$2\mathrm{E}_1$
Revoke	neg	neg	-	$\mathrm{E}_1+(r+2)\cdot\mathrm{E}_2$
Update	-	-	-	$2r\cdot\mathrm{E}_1$

"neg" denotes that the computation cost is negligible compared with several time-consuming operations, i.e., exponentiation, pairing, hash functions, and symmetric encryption and decryption.

TABLE IV
COMMUNICATION COST

	MSDE	MuED	SEMEKS	APTM
Secret keys	$n\cdot(\lvert\mathbb{Z}^*_{p'}\rvert+\lvert s\rvert)$	$n\cdot(\lvert\mathbb{Z}^*_{p'}\rvert+\lvert s\rvert)$	$n\cdot 12\lvert\mathbb{G}\rvert$	$n\cdot(\lvert\mathbb{Z}^*_{p'}\rvert+\lvert\mathbb{Z}^*_p\rvert+\lvert\mathbb{G}_1\rvert)$
Re-keys	$n\cdot(2\cdot\lvert\mathbb{Z}^*_{p'}\rvert)$	$n\cdot(\lvert\mathbb{Z}^*_{p'}\rvert+\lvert\mathbb{G}\rvert)$	-	-
Ciphertext	$\lvert\mathbb{Z}^*_{p'}\rvert+k\cdot(2\cdot\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}^*_{p'}\rvert)$	$\lvert\mathbb{Z}^*_{p'}\rvert+k\cdot(\lvert\mathbb{G}\rvert+\lvert\mathbb{G}_{T'}\rvert+\lvert\mathcal{M}\rvert+\lvert\mathcal{C}\rvert)$	$k\cdot 5\lvert\mathbb{G}\rvert$	$k\cdot(\lvert\mathbb{Z}^*_{p'}\rvert+\lvert\mathbb{G}\rvert)$
Query[1]	$\lvert\mathbb{Z}^*_{p'}\rvert+2k'\cdot\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}^*_{p'}\rvert+k'\cdot\lvert\mathbb{G}\rvert$	$k'\cdot 5\lvert\mathbb{G}\rvert$	$k'\cdot\lvert\mathbb{G}\rvert+6\lvert\mathbb{Z}^*_p\rvert+3\lvert\mathbb{G}_1\rvert$
Signature	-	-	-	$6\lvert\mathbb{Z}^*_p\rvert+3\lvert\mathbb{G}_1\rvert$

[1] It denotes the trapdoor for MSDE, MuED and SEMEKS, and while it contains the trapdoor and the signature for APTM.

SEMEKS[5] [28]). All the notations in the following evaluation can be referred in Table I.

A. Experimental Setting

In order to evaluate the practical performance, we implement all the algorithms in the four schemes using python 2.7 based on the pairing-based cryptography library [36] with version 0.5.14 and the Charm [37] framework for rapidly prototyping cryptosystems with version 0.42. Our experiments are run on a Ubuntu 12.04 virtual machine with a single core at 3.2 GHZ and 8GB RAM. In our experiments, we realize MSDE, MuED, SEMEKS and the part of SE in APTM based on the elliptic curve SS512 ($|\mathbb{G}| = 512$ bits, $|\mathbb{G}_{T'}| = 1024$ bits), which is a symmetric elliptic curve with base field 512-bit and embedding degree 2. And we implement the short group signature in APTM based on the asymmetric elliptic curves: MNT159 ($|\mathbb{G}_1| = 159$ bits, $|\mathbb{G}_2| = 477$ bits, $|\mathbb{G}_T| = 954$ bits), MNT201 ($|\mathbb{G}_1| = 201$ bits, $|\mathbb{G}_2| = 603$ bits, $|\mathbb{G}_T| = 1206$ bits) and MNT224 ($|\mathbb{G}_1| = 224$ bits, $|\mathbb{G}_2| = 672$ bits, $|\mathbb{G}_T| = 1344$ bits) with embedding degree 2, and a 159-bit, 201-bit, and 224-bit base field, respectively. Each curve above has a 160-bit group order, and thus we have $|\mathbb{Z}_p| = |\mathbb{Z}_{p'}| = 160$ bits.

[5]We omit the part of data sharing and only extract the parameters and algorithms for multiuser keyword searching.

B. Evaluation Results

Table II lists the major differences of APTM from the other three schemes in terms of utility and security. Compared with the proxy-based solutions (MSDE and MuED), APTM supports the anonymous matching without leaking the identities of workers while eliminating the storage cost of rekeys on the crowd-server. Compared with the broadcast-based solution (SEMEKS), APTM achieves the efficient revocation and traceability. Compared with both of two types of solutions, APTM can be easily adapted to realize various matching functions which include but are not limited to the single-keyword matching. Moreover, APTM resists the server-user collusion attack in MSDE and avoids the server-owner interaction during the keyword encryption in MuED. Meanwhile, we analyze each algorithm for all the schemes in terms of computation complexity and communication cost in Tables III and IV, respectively. Then, we carry out a detailed performance evaluation for each entity involved including the KM, each requester, each worker and the crowd-server as follows.

KM: The overhead on the KM in APTM is mainly from three parts: 1) system initialization; 2) worker revocation; and 3) signature tracing.

In the system initialization, the KM needs to generate the public keys and secret keys, and transmit the secret keys to all the workers. Its computation complexity is $\mathrm{E}+(n+2)\cdot\mathrm{E}_1+\mathrm{E}_2$

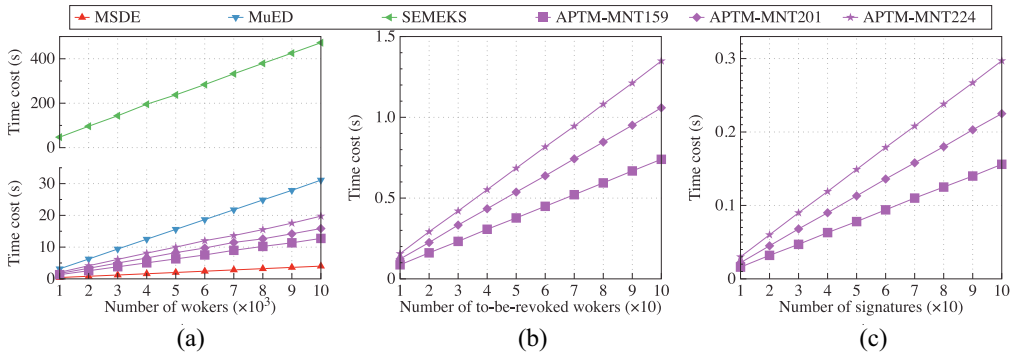


Fig. 3. Time cost on the KM. (a) System initialization, (b) worker revocation, and (c) signature tracing.

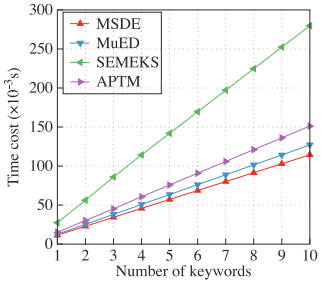


Fig. 4. Time cost of requirement encryption.

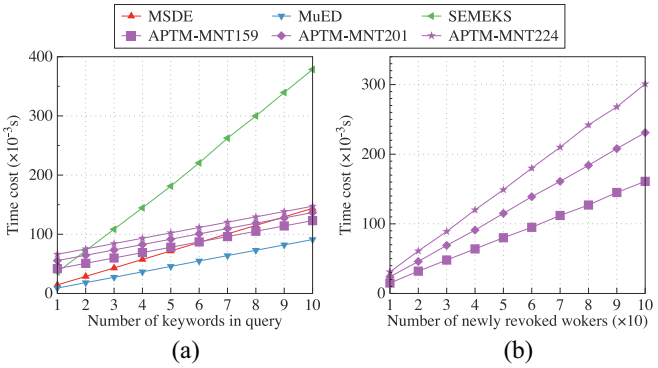


Fig. 5. Time cost on the worker. (a) Trapdoor and signature generation and (b) group private signing key update.

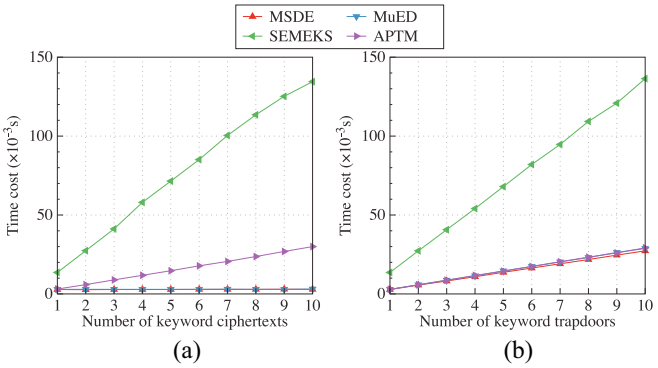


Fig. 6. Time cost of matching over a single requirement ciphertext. Under different number of keyword (a) ciphertexts when $k' = 1$ and (b) trapdoors when $k = 1$.

and the total communication cost of secret keys is $n \cdot (|\mathbb{Z}^*_{p'}| + |\mathbb{Z}^*_p| + |\mathbb{G}_1|)$. The transmission cost of secret keys in APTM is less than that in SEMEKS but more than that in MSDE and MuED. However, APTM saves the transmission cost of rekeys to the crowd-server compared with MSDE and MuED. In Fig. 3(a), we vary the value of n from 1000 to 10 000 to test the time cost of system initialization, and notice that time cost in all the schemes increases linearly with the number of workers (n). APTM slightly outperforms MuED and far outperforms SEMEKS. We can also observe that in the APTMs under different curves, the higher the MNT curve is, the more the corresponding time cost is. It takes about 20 s for the most expensive APTM-MNT224 to setup the system with 10 000 workers, which is a one-time cost and acceptable for the KM.

In the worker revocation, the KM needs to update the group public key and the revocation list. In Fig. 3(b), we vary the value of r to test the time cost of the **Revoke** algorithm, and observe that the time in all the APTMs is linear with the number of to-be-revoked workers (r), which validates its computation complexity $\mathrm{E}_1 + (r+2) \cdot \mathrm{E}_2$. When revoking one worker, it takes the time varying from 7.4 to 13 ms in all the APTMs, and when revoking 100 workers at once under the most expensive MNT224, it takes about 1.35 s, which will not place a burden on the KM.

We also evaluate the efficiency of signature tracing for the APTMs under different curves in Fig. 3(c). To trace a signature, the time cost under each curve is 1.6 ms, 2.2 ms, and 3 ms, respectively. The tracing operation will only be executed when the secret key leakage happens.

Requester: To evaluate the efficiency of requirement encryption, we vary the value of k to measure the computation cost of the **Enc** algorithm, as shown in Fig. 4. We can observe that the computation time in all the schemes is linear with the number of keywords in the task requirement (k), as analyzed in the computation complexities. Although the time cost in APTM is a little more than that in MSDE and MuED, it is far less than that in SEMEKS. For example, when k is set as 10, APTM takes 150 ms to encrypt the task requirement, and MSDE, MuED, and SEMEKS needs 114 ms, 127 ms, and 280 ms, respectively. Moreover, the communication cost of ciphertext in APTM ($k \cdot (|\mathbb{Z}^*_{p'}| + |\mathbb{G}|)$) is the minimum among all the schemes.

Worker: The main job on the worker in APTM includes the generation of trapdoor and signature, and the update of group private signing key.

In order to save the cost in the trapdoor and signature generation on multiple keywords, we concatenate all the generated trapdoors as the whole message and only generate one signature. Therefore, its total communication cost of trapdoor and signature is $k' \cdot |\mathbb{G}| + 6|\mathbb{Z}_p^*| + 3|\mathbb{G}_1|$, which is the sum of $k' \cdot |\mathbb{G}|$ for k' trapdoors and $6|\mathbb{Z}_p^*| + 3|\mathbb{G}_1|$ for one signature, which is less than that in MSDE and SEMEKS. Fig. 5(a) shows the total time cost of the **Trap** and **Sign** algorithms with respect to the number of keywords under different curves, in comparison with MSDE, MuED, and SEMEKS. The time cost in all the APTMs is linear with the number of keywords in the query (k') and the APTMs far outperform SEMEKS. We notice that the APTMs will become more efficient than MSDE with the increasing number of keywords. For example, when $k' = 10$, MSDE takes 144 ms to generate the trapdoor while the APTMs need 123 ms, 136 ms, and 147 ms under MNT159, MNT201, and MNT224, respectively, in which the signature generation takes 32.8 ms, 46.4 ms, and 57.1 ms, respectively.

Fig. 5(b) shows that the time cost to update the group private signing key is linear with the number of newly revoked workers (r) in the revocation list RL, as verified by its computation complexity $2r \cdot \mathrm{E}_1$. This updating time cost on a small number of newly revoked workers is quite efficient. For example, the time to update the group private signing key for one newly revoked worker is 1.5 ms, 2.4 ms, and 3 ms under MNT159, MNT201, and MNT 224, respectively. Thus, the singing key update on the worker side, incurred by worker revocation, has slight impact on the query efficiency.

Crowd-Server: The cost on the crowd-server includes the signature verification cost and the trapdoor matching cost. In the signature verification, its computation complexity is $8E_1 + 5E_T + 5P_{12} + H$. The time to verify a signature is 41.5 ms, 61.7 ms, and 76.9 ms under MNT159, MNT201, and MNT 224, respectively. In the matching over a single requirement ciphertext, its computation complexity is $k \cdot k' \cdot (\mathrm{P} + \mathrm{H}_{\mathrm{gt}'})$. Fig. 6(a) shows that given the trapdoor of a single query keyword ($k' = 1$), the matching time in SEMEKS and APTM increases linearly with the number of keyword ciphertexts (k), while MSDE and MuED are both insensitive. Fig. 6(b) shows that given the requirement ciphertext of a single keyword ($k = 1$), the matching time in all the schemes increases with the increasing number of keyword trapdoors (k'). Although the APTM is more time-consuming than MSDE and MuED, it is much more efficient than SEMEKS. Moreover, the matching time on a more powerful cloud-based crowdsourcing platform would be largely reduced for practical use.

VII. Conclusion

In this paper, we systematically studied the privacy issues in the task matching for crowdsourcing and defined a set of privacy requirements against the crowd-server, dishonest workers, and revoked workers. Then we designed a single-keyword task matching scheme in the multirequester/multiworker environment. Compared with the existing proxy-based and broadcast-based solutions, the proposed scheme achieves identity anonymity and efficient revocation, meanwhile can be adapted to realize various matching functions. Finally, we analyzed the performance of the proposed scheme from both theoretical and experimental aspects. The detailed performance evaluation shows that the proposed scheme is feasible for practical use.

References

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

[2] V. Ambati, S. Vogel, and J. G. Carbonell, "Towards task recommendation in micro-task markets," in *Proc. Human Comput.*, 2011, pp. 1–4.

[3] M.-C. Yuen, I. King, and K.-S. Leung, "Task recommendation in crowdsourcing systems," in *Proc. 1st Int. Workshop Crowdsourcing Data Min.*, 2012, pp. 22–26.

[4] D. E. Difallah, G. Demartini, and P. Cudré-Maduro, "Pick-a-crowd: Tell me what you like, and i'll tell you what to do," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 367–377.

[5] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *Proc. VLDB Endowment*, vol. 7, no. 10, pp. 919–930, 2014.

[6] Y. Shen *et al.*, "Towards preserving worker location privacy in spatial crowdsourcing," in *Proc. IEEE GLOBECOM*, San Diego, CA, USA, 2015, pp. 1–6.

[7] Y. Gong, L. Wei, Y. Guo, C. Zhang, and Y. Fang, "Optimal task recommendation for mobile crowdsourcing with privacy control," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 745–756, Oct. 2016.

[8] J. Shu and X. Jia, "Secure task recommendation in crowdsourcing," in *Proc. IEEE GLOBECOM*, Washington, DC, USA, 2016, pp. 1–6.

[9] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2018.2791601.

[10] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Signal Process.*, Berkeley, CA, USA, 2000, pp. 588–593.

[11] J. Li *et al.*, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2010, pp. 1–5.

[12] C. Wang, N. Cao, J. Li, K. Ren, and W. J. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE ICDCS*, 2010, pp. 253–262.

[13] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[14] T. Moataz and A. Shikfa, "Boolean symmetric searchable encryption," in *Proc. 8th ACM SIGSAC Symp. Inf. Comput. Commun. Security*, 2013, pp. 265–276.

[15] D. Wang *et al.*, "Generalized pattern matching string search on encrypted data in cloud systems," in *Proc. IEEE INFOCOM*, 2015, pp. 2101–2109.

[16] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Adv. Cryptol. Eurocrypt*, 2004, pp. 506–522.

[17] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptography Conf.*, 2007, pp. 535–554.

[18] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proc. IEEE Signal Process.*, 2007, pp. 350–364.

[19] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Security*, vol. 19, no. 5, pp. 895–934, 2011.

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

[21] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 875–888.

[22] S. Faber *et al.*, "Rich queries on encrypted data: Beyond exact matches," in *Proc. Eur. Symp. Res. Comput. Security*, 2015, pp. 123–145.

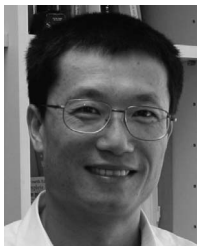
[23] S.-F. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for boolean queries," in *Proc. Eur. Symp. Res. Comput. Security*, 2016, pp. 154–172.
[24] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Security*, vol. 19, no. 3, pp. 367–397, 2011.
[25] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Security Pract. Exp.*, 2008, pp. 71–85.
[26] 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 Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016.
[27] Y. Miao *et al.*, "m2-ABKS: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting," *J. Med. Syst.*, vol. 40, no. 11, p. 246, 2016.
[28] 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. Security*, 2016, pp. 173–195.
[29] Y. Zhang *et al.*, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Inf. Sci.*, vol. 379, pp. 42–61, Feb. 2017.
[30] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: Efficient policy-hiding attribute-based access control," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2825289.
[31] D. Chaum and E. Van Heyst, "Group signatures," in *Proc. Adv. Cryptol. EUROCRYPT*, 1991, pp. 257–265.
[32] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. 11th ACM Conf. Comput. Commun. Security*, 2004, pp. 168–177.
[33] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," *Advances in Cryptology—CRYPTO 2004*, vol. 3152. Springer: Heidelberg, Germany, 2004, pp. 41–55.
[34] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2004, pp. 56–73.
[35] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.
[36] B. Lynn. *The Pairing-Based Cryptography (PBC) Library*. Accessed: Jan. 10, 2017. [Online]. Available: https://crypto.stanford.edu/pbc
[37] J. A. Akinyele *et al.*, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

Jiangang Shu (GS'16) received the B.E. and M.S. degrees from the Nanjing University of Information Science and Technology, Nanjing, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer science at the Department of Computer Science, City University of Hong Kong, Hong Kong.

He is currently a Visiting Postgraduate Research Student with the School of Information System, Singapore Management University, Singapore. His current research interests include security and privacy in crowdsourcing, cloud computing security, and steganography.

Ximeng Liu (S'13–M'16) received the B.Sc. degree in electronic engineering and Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2010 and 2015, respectively.

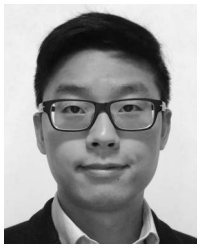
He is currently a Research Fellow with the School of Information System, Singapore Management University, Singapore, and a Qishan Scholar with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His current research interests include cloud security, applied cryptography, and big data security.



Xiaohua Jia (F'13) received the B.Sc. and M.Eng. degrees from the University of Science and Technology of China, Hefei, China, in 1984 and 1987, respectively, and the D.Sc. degree in information science from the University of Tokyo, Tokyo, Japan, in 1991.

He is currently the Chair Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include cloud computing and distributed systems, computer networks, and mobile computing.

Prof. Jia is an Editor of the IEEE INTERNET OF THINGS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS from 2006 to 2009, *Wireless Networks*, the *Journal of World Wide Web*, and the *Journal of Combinatorial Optimization*. He was the General Chair of ACM MobiHoc in 2008, a TPC Co Chair of IEEE GlobeCom in 2010 and Ad Hoc and Sensor Networking Symposium, and the Area-Chair of IEEE INFOCOM in 2010 and 2015.



Kan Yang (M'13) 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, in 2013.

He is currently a Tenure-Track Assistant Professor with the Department of Computer Science, University of Memphis, Memphis, TN, USA. His current 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.

Robert H. Deng (F'16) is an AXA Chair Professor of cybersecurity and the Director of Secure Mobile Centre, School of Information Systems, Singapore Management University, Singapore. His current research interests include data security and privacy, cloud security, and Internet of Things security.

Mr. Deng's professional contributions include an extensive list of positions in several industry and public service Advisory Boards, Editorial Boards, and conference committees. These include the Editorial Boards of *IEEE Security & Privacy Magazine*, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and the Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security.