12-2018

SybMatch: Sybil detection for privacy-preserving task matching in crowdsourcing

Jiangang SHU

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

Kan YANG

Yinghui ZHANG

Xiaohua JIA

See next page for additional authors

Author

Jiangang SHU, Ximeng LIU, Kan YANG, Yinghui ZHANG, Xiaohua JIA, and Robert H. DENG

Published in 2018 IEEE Global Communications Conference, GLOBECOM 2018, Abu Dhabi, United Arab Emirates, December 9-13: Proceedings, Article no. 8647346
https://doi.org/10.1109/GLOCOM.2018.8647346

SybMatch: Sybil Detection for Privacy-Preserving Task Matching in Crowdsourcing

Jiangang Shu*, Ximeng Liu†, Kan Yang‡, Yinghui Zhang§, Xiaohua Jia*, and Robert H. Deng†
*Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China
†School of Information Systems, Singapore Management University, Singapore
‡Department of Computer Science, University of Memphis, Memphis, TN, USA
§National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications, Xi'an, China
Email: gshu2-c@my.cityu.edu.hk, snbnix@gmail.com, kan.yang@memphis.edu,
yhzhaang@163.com, csjia@cityu.edu.hk, robertdeng@smu.edu.sg

Abstract—**The past decade has witnessed the rise of crowdsourcing, and privacy in crowdsourcing has also gained rising concern in the meantime. In this paper, we focus on the privacy leaks and sybil attacks during the task matching, and propose a privacy-preserving task matching scheme, called SybMatch. The SybMatch scheme can simultaneously protect the privacy of publishers and subscribers against semi-honest crowdsourcing service provider, and meanwhile support the sybil detection against greedy subscribers and efficient user revocation. Detailed security analysis and thorough performance evaluation show that the SybMatch scheme is secure and efficient.**

Index Terms—**Sybil detection, crowdsourcing, task matching, privacy-preserving**

1. Introduction

Crowdsourcing [1] as a distributed computing paradigm has attracted a lot of attention over the past decade. Many crowdsourcing platforms (e.g., Amazon Mechanical MTurk[1]) have been established to provide all kinds of crowdsourcing services, such as data sensing, image categorization. Wit utilization of crowdsourcing, task publishers can outsource complex tasks to task subscribers through a crowdsourcing service provider (CSP) as the broker.

To achieve the efficient and accurate task recommendation, the CSP needs to match task requirements specified by task publishers with interests given by task subscribers [2]-[4]. However, considering the CSP cannot be fully trusted in the sense that it may obtain the sensitive information about task requirements and interests, such matching solutions over plaintexts will make publishers and subscribers in danger [5]-[8]. Therefore, it is important to protect the privacy of publishers and subscribers against the CSP during the task matching.

Searchable encryption (SE) [10] is a potential technique and it has been utilized to realize various matching functionalities, such as ranked search [11], personalized search [12], and blockchain based search [13]. However, most of SE schemes only allow the single user holding the secret key to query the encrypted data. Since there are multiple publishers and multiple subscribers in crowdsourcing, query accountability cannot be achieved and user revocation will incur the re-initialization of system if using single-user SE. Thus, these single-user SE schemes cannot be directly applied to the multi-user crowdsourcing environment. To make single-user SE applicable to the multi-user environment, Dong *et al.* [14] and Kiayias *et al.* [15] respectively utilized proxy re-encryption and key derivation method to realize the multi-user SE schemes where every user is allowed to search over the encrypted data published by all data owners.

Inspired by those works, we utilized proxy re-encryption to design two schemes [16], [17], which respectively achieves the single-keyword and multi-keyword task matching while protecting the privacy against the CSP. However, in real crowdsourcing, since the subscribers are from various backgrounds and their true identities are never being verified, they cannot be fully trusted either. In order to subscribe more tasks from the CSP, a greedy worker may launch the sybil attacks in the way that it frequently changes its pseudonym and repeatedly submits its subscription to the CSP. Therefore, efficient sybil detection should be considered in the designing of privacy-preserving task matching for defending against the sybil attacks.

In this paper, we analyze the potential privacy leaks and attacks during the task matching in crowdsourcing, and propose a privacy-preserving task matching scheme with sybil detection, called SybMatch. In SybMatch, we combine PEKS [18] with ID-based signature [19] to simultaneously protect the privacy against the CSP and achieve the sybil detection against the greedy subscribers. Moreover, the SybMatch scheme supports the efficient revocation. Through detailed security analysis, we prove that the SybMatch scheme is IND-CKA secure and existential unforgeable. Theoretical analysis and simulation evaluation demonstrate that the SybMatch scheme is efficient and feasible.

The rest of the paper is organized as follows. Section 2 presents the models, design goals and some preliminaries. The detailed construction and security analysis are described in Section 3. Then we analyze the the performance in Section 4 and finally conclude the paper in Section 5.

1. https://www.mturk.com/mturk/welcome

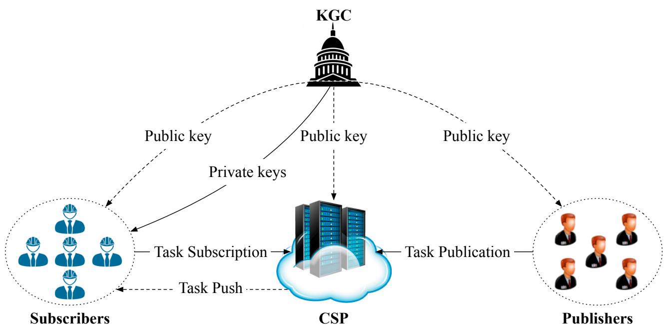


Figure 1. System model

2. Problem Formulation

2.1. System Model

As shown in Figure 1, there are four entities in the crowdsourcing system: a key generation centre (KGC), a crowdsourcing service provider (CSP), multiple subscribers and multiple publishers. Their roles are defined as follows:

The **KGC** is mainly responsible for system initialization and user registration. It outputs a public key to all the publishers for task publication, and assigns a distinct private key to each registered subscriber for task subscription.

The **subscribers** are the users who subscribe tasks from the CSP. In order to subscribe the tasks of its interests, a subscriber specifies a subscription (e.g., keywords, expressions) and encrypts it with its own private key. Then the subscriber sends the encrypted subscription to the CSP.

The **publishers** are the users who publish tasks on the CSP. When publishing a task, a publisher sets the task requirement (e.g., keywords, expressions) and encrypts it with the public key while encrypting the task content. Then the publisher submits the requirement ciphertext to the CSP, together with the encrypted task content.

The **CSP** is in charge of task matching. Upon receiving a task publication, the CSP performs the matching process between the requirement ciphertext and the subscriptions, and pushes the task to the matched subscribers.

Note that the encryption and decryption of task content are out of scope of this paper.

2.2. Threat Model

The KGC is fully *trusted* and the publishers are *honest*.

The authorized subscribers are considered as *honest-but-greedy* in the sense that they may launch sybil attacks to subscribe mores tasks from the CSP.

The CSP is considered as *honest-but-curious* in the sense that it will honestly execute the designed protocol but may be curious to know the sensitive information about the received task requirements and subscriptions. We assume that the CSP will not launch active attacks, such as collusion with the authorized subscribers or publishers. This assumption is reasonable, as a large and reputable CSP will not launch these reputation-damaging attacks.

2.3. Design Goals

Multi-publisher and multi-subscriber. The proposed scheme should support the task-subscriber matching between multiple publishers and multiple subscribers. Specifically, a constant-size requirement ciphertext output by any publisher can be tested with a constant-size encrypted subscription generated by any authorized subscriber.

User scalability. The proposed scheme shall allow the efficient user enrollment and user revocation in the dynamic crowdsourcing.

Security. 1) *Privacy-preserving.* The proposed scheme shall protect requirement privacy and subscription privacy from the CSP; 2) *Sybil detection.* The proposed scheme shall support the sybil detection that enables the CSP to defend against sybil attacks launched by greedy workers;

2.4. Preliminaries

Bilinear Map. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of a same prime order p with g as a generator of $\mathbb{G}$. A bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has the following properties:

- Bilinearity: $\forall a, b \in \mathbb{Z}_p^*$, $e(g^a, g^b) = e(g, g)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$.
- Computability: e is efficiently computed.

Computational Diffie-Hellman (CDH) Assumption. The CDH problem in $\mathbb{G}$ is stated as: given $(g, g^a, g^b) \in \mathbb{G}^3$ as input, where $a, b \in \mathbb{Z}_p^*$ are randomly chosen, compute $g^{ab} \in \mathbb{G}$. The CDH assumption holds if the advantage to solve the CDH problem is negligible for any PPT algorithm.

Bilinear Diffie-Hellman (BDH) Assumption. The BDH problem in $\mathbb{G}$ is stated as: given $(g, g^a, g^b, g^c) \in \mathbb{G}^4$ as input, where $a, b, c \in \mathbb{Z}_p^*$ are randomly chosen, output $e(g, g)^{abc} \in \mathbb{G}_T$. The BDH assumption holds if the advantage to solve the BDH problem is negligible for any PPT algorithm.

3. The SybMatch Scheme

For the sake of clarity, we consider the single keyword matching in SybMatch. More complex matching functionalities (e.g., range, boolean) can be easily achieved by integrating other SE schemes. As illustrated in Figure 2, the SybMatch system proceeds as follows:

- *Initialization and Registration.* The KGC initializes the system and assigns keys for registered users.
- *Subscription and Detection.* Each subscriber generates the encrypted subscription and sends it to the CSP which then performs the sybil detection for the received subscription.
- *Publication and Matching.* A publisher encrypts the task requirement and submits the requirement ciphertext to the CSP which then conducts the requirement-subscription matching process.
- *Revocation.* The KGC revokes the leaving subscribers in cooperation with the CSP.

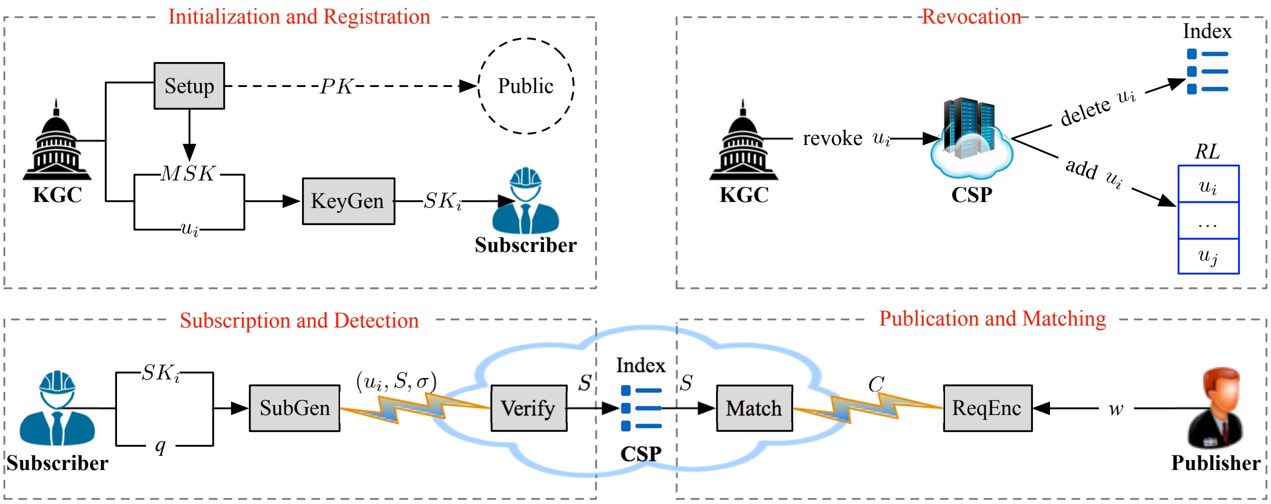


Figure 2. Framework of SybMatch

3.1. Construction

Initialization and Registration. The KGC initializes the system by calling the Setup algorithm, and outputs a public key PK to all the entities involved in the system including the CSP, publishers and subscribers. Then for each registered subscriber u_i, the KGC assigns it a distinct private key SK_i by executing the KeyGen algorithm.

$\mathsf{Setup}(1^\lambda) \to (PK, MSK)$. It generates two multiplicative cyclic groups $\mathbb{G}$, $\mathbb{G}_T$ of a same prime order p with a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$. Then it randomly selects $\alpha, x \in \mathbb{Z}_p^*$ and defines hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G}_T \to \{0,1\}^{\log p}$, $H_3 : \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_p^*$, $H_4 : \{0,1\}^* \to \mathbb{G}$ (Note that H_1 and H_4 are used for different purposes). The public key is output as

$$PK = (\mathbb{G}, \mathbb{G}_T, e, p, g, g^\alpha, g^x, H_1, H_2, H_3, H_4)$$

and the master secret key is set as $MSK = (\alpha, x)$.

$\mathsf{KeyGen}(MSK, u_i) \to SK_i$. Given a subscriber identity u_i, it outputs $SK_i = (ssk, K_i)$, where $ssk = \alpha$ and $K_i = H_4(u_i)^x$, as a private key of the subscriber u_i.

Subscription and Detection. To subscribe the tasks matching with some keyword q, a subscriber u_i generates the subscription and signature (S, σ) of q by calling the SubGen algorithm with its private key SK_i, and sends (S, σ) to the CSP together with the identity u_i. Upon receiving the subscription request from the subscriber u_i, the CSP performs the sybil attack detection to check whether the subscriber u_i has previously submitted the subscription. As illustrated in Algorithm 1, the sybil attack detection proceeds as follows:

1) It first verifies whether the request is indeed from the claimed identity u_i by calling the Verify algorithm. If $\mathsf{Verify}(PK, u_i, S, \sigma) = 0$ which means the request is not from the claimed identity u_i, it drops the subscription request.

Algorithm 1: Sybil attack detection

```
Input: Subscription Request (u_i, S, σ), Public Key
       PK, Revocation List RL, Index I
Output: Invalid Signature, Identity Revoked, Sybil
        Detected or Sybil Undetected
if Verify(PK, u_i, S, σ) = 0 then
  | Return "Invalid Signature"
else
  if u_i ∈ RL then
    | Return "Identity Revoked"
  else
    if u_i ∈ I then
      | Return "Sybil Detected"
    else
      Add (u_i, S) in I
      Return "Sybil Undetected"
```

2) Otherwise, it then checks whether the subscriber u_i is revoked by checking the existence of u_i in the revocation list RL, which stores the identities of revoked subscribers and is initially empty. If the subscriber u_i has been revoked, it drops the subscription request.
3) Otherwise, it continues to check whether there is an entry corresponding to u_i stored in the subscription index I. If no entry corresponding to u_i in the index, it stores the subscription S in the index. Otherwise, it drops the subscription request.

$\mathsf{SubGen}(SK_i, q) \to (S, \sigma)$. Given a keyword q, it first computes the subscription $S = H_1(q)^\alpha$. Then it randomly picks $r \in \mathbb{Z}_p^*$ and outputs a signature $\sigma = (U, V)$ of S as:

$$U = g^r, \quad V = H_4(u_i)^r \cdot K_i^h,$$

where $h = H_3(S, U)$.

$\mathsf{Verify}(PK, u_i, S, \sigma) \rightarrow 1/0$. Given a signature $\sigma = (U, V)$ of a subscription S for an identity u_i, it computes $h = H_3(S, U)$ and accepts the signature if $(g, H_4(u_i), U \cdot g^{x \cdot h}, V)$ is valid Diffie-Hellman Tuple. That is, it checks if

$$e(H_4(u_i), U \cdot g^{x \cdot h}) \stackrel{?}{=} e(g, V).$$

If the equality holds, it outputs 1; otherwise, it outputs 0.

Theorem 1 (Signature Correctness)**.** *The signature process is correct. That is, if σ is a signature of a subscription S for an identity u_i, we have* $\mathsf{Verify}(PK, u_i, S, \sigma) = 1$.

Proof. Suppose $\sigma = (U, V)$ where

$$\begin{aligned} U &= g^r, \\ V &= H_4(u_i)^r \cdot K_i^h = H_4(u_i)^{r+xh}, \\ h &= H_3(S, U), \end{aligned}$$

we have

$$\begin{aligned} e(H_4(u_i), U \cdot g^{x \cdot h}) &= e(H_4(u_i), g^{r+xh}), \\ e(g, V) &= e(g, H_4(u_i)^{r+xh}). \end{aligned}$$

Thus we have $e(H_4(u_i), U \cdot g^{x \cdot h}) = e(g, V)$. □

Remark. When receiving k subscription requests at the same time, the CSP can conduct an efficient batch verification as follows.

$\mathsf{BatchVerify}(\{(u_i, S_i, \sigma_i)\}_{1 \leq i \leq k}) \rightarrow 1/\{i\}$. Given k subscription requests $(u_1, S_1, \sigma_1), \cdots, (u_k, S_k, \sigma_k)$ where $\sigma_i = (U_i, V_i)$, it checks if

$$e(g, \prod_{i=0}^{k} V_i) \stackrel{?}{=} \prod_{i=0}^{k} e(H_4(u_i), U_i \cdot g^{x \cdot h_i}),$$

where $h_i = H_3(S_i, U_i)$. If the quality holds, it outputs 1; otherwise, it uses divide-and-conquer method to find the index $\{i\}$ of invalid signatures.

Publication and Matching. When publishing a task, a publisher encrypts a requirement keyword w with the ReqEnc algorithm and submits the ciphertext C to the CSP. Upon receiving the request of task publication, the CSP conducts the requirement-subscription matching process by running Match with each subscription stored in the index I. If $\mathsf{Match}(PK, C, S) = 1$ for some subscription S, the CSP pushes the task to the corresponding subscriber.

$\mathsf{ReqEnc}(PK, w) \rightarrow C$. It randomly chooses $t \in \mathbb{Z}_p^*$, and computes $A = e(H_1(w), (g^\alpha)^t)$. It outputs $C = (g^t, H_2(A))$ as a ciphertext of keyword w.

$\mathsf{Match}(PK, C, S) \rightarrow 1/0$. Given a ciphertext $C = (C_1, C_2)$ and a subscription S, it checks if

$$H_2(e(S, C_1)) \stackrel{?}{=} C_2.$$

If the equation holds, it outputs 1; otherwise, it outputs 0.

Theorem 2 (Matching Correctness)**.** *The task matching process is correct. That is, if both C and S are constructed based on a same keyword, we have* $\mathsf{Match}(PK, C, S) = 1$.

Proof. Given a ciphertext $C = (C_1, C_2)$, where $C_1 = g^t$ and $C_2 = H_2(e(H_1(w), (g^\alpha)^t))$ and a subscription $S = H_1(w)^\alpha$, we have

$$\begin{aligned} e(S, C_1) &= e(H_1(w)^\alpha, g^t) \\ \Leftrightarrow H_2(e(S, C_1)) &= H_2(e(H_1(w), g^{\alpha \cdot t})) \\ \Leftrightarrow H_2(e(S, C_1)) &= C_2. \end{aligned}$$

That completes the proof.

□

Revocation. When a subscriber u_i leaves the system, the KGC notifies the CSP to revoke u_i's subscription ability by enabling the CSP to run the Revoke algorithm.

$\mathsf{Revoke}(u_i)$. Given a subscriber identity u_i, it revokes the subscriber u_i's subscription ability by deleting the subscription of u_i in the index I, and at the same time adds the identity u_i into the revocation list RL, i.e., $RL = RL \cup u_i$.

3.2. Security Analysis

According to the security goals we set in Section 2.3, SybMatch should protect subscription privacy and requirement privacy, and meanwhile support the sybil detection. As for subscription privacy, same as most of PEKS schemes [18], we don't consider offline keyword guessing attacks and only ensure the one-wayness of subscription in SybMatch.

As for requirement privacy, we prove that the SybMatch scheme is Computationally Indistinguishable Secure against Adaptive Chosen Keyword Attacks (IND-CKA) through Theorem 3. Informally speaking, the adversary cannot distinguish the ciphertexts of two arbitrary keywords unless the corresponding subscriptions are available.

Theorem 3. *The SybMatch scheme is IND-CKA secure in the random oracle model under the BDH assumption. If a PPT adversary $\mathcal{A}$, making at most q_{H_2} hash function queries to H_2 and at most q_S subscription queries, breaks the IND-CKA security game with advantage ε, we can construct a PPT algorithm $\mathcal{B}$, which breaks the BDH problem with advantage at least $\varepsilon' = \varepsilon/(eq_{H_2} q_S)$.*

Proof. Given a BDH instance $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, $\mathcal{B}$ simulates the IND-CKA game with $\mathcal{A}$ as follows:

Setup. $\mathcal{B}$ randomly chooses $x \in \mathbb{Z}_p^*$, and gives $\mathcal{A}$ the public key $PK = (g, g^a, g^x)$.

H_1**-query.** $\mathcal{B}$ maintains a hash list called H_1-list, which stores the tuples $< w_i, c_i, a_i, b_i >$ and is initially empty. When $\mathcal{A}$ queries a keyword w_i, $\mathcal{B}$ proceeds as follows:

1) If w_i already exits in the H_1-list, $\mathcal{B}$ responds with the corresponding b_i.
2) Otherwise, $\mathcal{B}$ picks a random coin $c_i \in \{0, 1\}$ such that $\Pr[c_i = 0] = 1/(q_S + 1)$, and randomly chooses $a_i \in \mathbb{Z}_p^*$. Then it computes b_i as follows.

$$b_i = \begin{cases} g^b \cdot g^{a_i}, & if\ c_i = 0 \\ g^{a_i}, & if\ c_i = 1 \end{cases}$$

After that, $\mathcal{B}$ responds with b_i and adds the tuple $< w_i, c_i, a_i, b_i >$ to the H_1-list.

H_2**-query** When $\mathcal{A}$ queries $H_2(A)$, $\mathcal{B}$ responds with a random element $R \in \mathbb{G}_T$ and stores $< A, R >$ in H_2-list.

H_3**-query.** When $\mathcal{A}$ issues a query of $H_3(S_i, U_i)$, $\mathcal{B}$ chooses a random value $h_i \in \mathbb{Z}_p^*$ and responds with h_i.

H_4**-query.** When $\mathcal{A}$ issues a query of $H_4(u_i)$, $\mathcal{B}$ Chooses a random value $x_i \in \mathbb{Z}_p^*$ and responds with $H_4(u_i) = g^{x_i}$.

Subscription query. When $\mathcal{A}$ issues a subscription query of a keyword w_i for an identity u_i, $\mathcal{B}$ invokes the random oracle H_1 with w_i and then proceeds as follows:

1) If $c_i = 0$, it declares failure and exits.
2) Otherwise, it reports $b_i = g^{a_i}$. Then $\mathcal{B}$ randomly chooses $r_i \in \mathbb{Z}_p^*$ and responds $\mathcal{A}$ with $S_i = (g^a)^{a_i}$, $\sigma_i = (U_i = g^{r_i - x \cdot h_i}, V_i = g^{x_i \cdot r_i})$. Note that S_i is the correct subscription of w_i under the public key (g, g^a), and since $(g, H_4(u_i), U_i \cdot g^{x \cdot h_i}, V_i)$ is a valid DDH tuple, σ_i is a valid signature.

Challenge. $\mathcal{A}$ provides two keywords w_0, w_1 on which it wishes to be challenged. $\mathcal{B}$ invokes the random oracle H_1 and obtains $H_1(w_0) = b_0$, $H_1(w_1) = b_1$. If $c_0 = c_1 = 1$, $\mathcal{B}$ reports failure and exists. Otherwise, $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$ such that $c_b = 0$, and responds with $\mathcal{C} = (g^c, J)$ where $J = H_2(H_1(w_b), (g^a)^c) = H_2(e(g, g)^{ac(b+a_i)})$.

Output. $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$. $\mathcal{B}$ picks a random pair (A, R) from the H_2-list and outputs $A/e(g^a, g^c)^{a_b}$ as its guess for $e(g, g)^{abc}$.

To calculate the success probability of $\mathcal{B}$, we define the following events:

- ξ_1: $\mathcal{B}$ doesn't abort during the subscription phase.
- ξ_2: $\mathcal{B}$ doesn't abort during the challenge phase.

During the subscription phase, if $c_i = 0$, then $\mathcal{B}$ aborts. Thus, we have Pr$[\xi_1] = (1 - 1/(q_S + 1))^{q_S} \geq 1/e$. During the challenge phase, if $c_0 = c_1 = 1$, then $\mathcal{B}$ aborts. Thus, we have Pr$[\xi_2] = 1 - (1 - 1/(q_S + 1))^2 \geq 1/q_S$. Since ξ_1 and ξ_2 are independent, we have Pr$[\xi_1 \wedge \xi_2] =$ Pr$[\xi_1] \cdot$ Pr$[\xi_2] \geq 1/(e \cdot q_S)$. Considering that $\mathcal{B}$ chooses the correct pair with probability at least $1/q_{H_2}$, $\mathcal{B}$'s success advantange is at least $\varepsilon/(eq_{H_2}q_S)$. □

As for sybil detection, we need to prove that the subscription request (u_i, S, σ) cannot be forged. Obviously, the unforgeability of (u_i, S, σ) depends on the unforgeability of ID-based signature scheme [19]. According to Theorem 3 in [19], this ID-based signature scheme achieves the existential unforgeability under adaptively chosen message and ID attacks in the random oracle model under the CDH assumption.

4. Performance Evaluation

In this section, we evaluate the performance of SybMatch through theoretical analysis and simulation study, in comparison with two state-of-the-art related schemes: a proxy based multi-user SE scheme called MSDE [14] and a proxy-free multi-user SE scheme called SEMEKS [15].

TABLE 1. COMPUTATION COMPLEXITY

Algorithm	MSDE	SEMEKS	SybMatch
Setup	neg	8E	2E
KeyGen	neg	16E	$\mathrm{E} + \mathrm{H_4}$
SubGen	$6\mathrm{E} + \mathrm{f_s}$	12E	$4\mathrm{E} + \sum_{\mathrm{i}=1,3,4} \mathrm{H_i}$
Verify	-	-	$\mathrm{E} + 2\mathrm{P} + \sum_{\mathrm{i}=3,4} \mathrm{H_i}$
ReqEnc	$3\mathrm{E} + \mathrm{f_s} + \mathrm{H_5}$	9E	$2\mathrm{E} + \mathrm{P} + \sum_{\mathrm{i}=1,2} \mathrm{H_i}$
Match	$2\mathrm{E} + \mathrm{H_5}$	5P	$\mathrm{P} + \mathrm{H_2}$
Revoke	neg	-	neg

E: exponentiation on group $\mathbb{G}$; P: pairing operation on group $\mathbb{G}$. Hash operations $\mathrm{H_1}$, $\mathrm{H_2}$, $\mathrm{H_3}$, $\mathrm{H_4}$, $\mathrm{H_5}$: $\mathbb{G} \rightarrow \mathbb{Z}_\mathrm{p}^*$; $\mathrm{f_s}$: key-based hash operation $\{0, 1\}^* \rightarrow \mathbb{Z}_p^*$.
"neg": negligible compared with the above operations.

TABLE 2. COMMUNICATION COST

	MSDE	SEMEKS	SybMatch
Secret key	$\lvert\mathbb{Z}_p^*\rvert + \lvert s\rvert$	$12\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_p^*\rvert + \lvert\mathbb{G}\rvert$
Re-key	$2 \cdot \lvert\mathbb{Z}_p^*\rvert$	-	-
Subscription	$\lvert\mathbb{Z}_p^*\rvert + 2\lvert\mathbb{G}\rvert$	$5\lvert\mathbb{G}\rvert$	$\lvert\mathbb{G}\rvert$
Signature	-	-	$\lvert\mathbb{Z}_p^*\rvert + 2\lvert\mathbb{G}\rvert$
Ciphertext	$2\lvert\mathbb{Z}_p^*\rvert + 2\lvert\mathbb{G}\rvert$	$5\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_p^*\rvert + \lvert\mathbb{G}\rvert$

$\lvert\mathbb{G}\rvert$: element size in $\mathbb{G}$; $\lvert\mathbb{Z}_p^*\rvert$: element size in $\mathbb{Z}_p^*$.

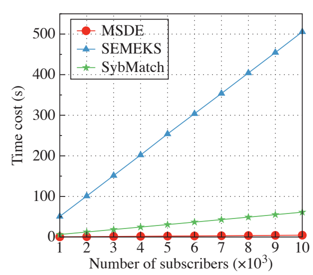


Figure 3. Subscriber registration

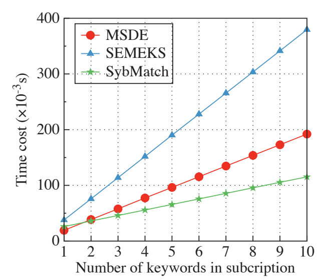


Figure 4. Subscription generation

We first theoretically analyse the computation complexity and communication cost for all the schemes, as shown in TABLE 1, TABLE 2, respectively. We can see SybMatch far outperforms SEMEKS in all aspects of computation and communication costs, and outperforms MSDE in some aspects, e.g., communication costs of subscription and ciphertext.

To evaluate the practical performance, we implement all the schemes in python on a Ubuntu 12.04 virtual machine with a single core at 3.20GHz and 1GB RAM, relying on the PBC library [20] and the Charm framework [21]. In the simulation, we choose a 160-bit prime p and a SS512 curve where the base field size is 512-bit and the embedding degree is 2.

Figure 3 shows the time cost of subscriber registration under different number of subscribers. It shows that SybMatch slightly underperforms MSDE but far outperforms SEMEKS. Figure 4 indicates that the subscription generation of SybMatch will be more efficient than those in both MSDE and SEMEKS when the number of keywords in the subscription increases. Figure 5 shows that signature

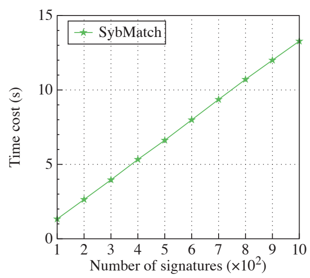


Figure 5. Signature verification

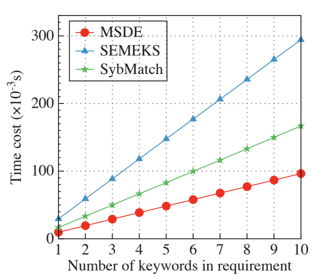
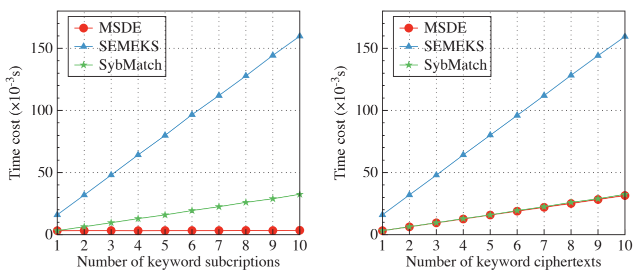


Figure 6. Requirement encryption



(a) (b)

Figure 7. Match. (a) when the number of keyword ciphertexts is 1; (b) when the number of keyword subscriptions is 1.

verification on the CSP is quite efficient. For example, it only takes about 1.32s to conduct a batch verification for 100 signatures. Figure 6 shows that requirement encryption in SybMatch is less efficient than that in MSDE, but more efficient than that in SEMEKS. We also evaluate the performance of matching under different number of keyword subscriptions and keyword ciphertexts, as shown in Figure 7, and observe that the matching operation in SybMatch is much more efficient than that in SEMEKS.

5. Conclusion

In this paper, we formulated the problem of privacy-preserving task matching for multi-publisher and multi-subscriber crowdsourcing and identified sybil attacks launched by greedy subscribers. To address the privacy leaks and defend against the sybil attacks, we designed the SybMatch scheme which can realize the privacy-preserving task matching while supporting the efficient sybil detection. We also comprehensively analyzed the security and performance of SybMatch. Through detailed theoretical analysis and simulation study in comparison with related works, we validated that the SybMatch scheme is efficient and feasible.

Acknowledgment

This work was supported by grants from Research Grants Council of Hong Kong [GRF CityU 11208917, CRF CityU C1008-16G] and NSF China Grant [No. 61732022 and No. 61702105].

References

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, 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. of Human Computation*, 2011, pp. 1-4.

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

[4] 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 *Proc. of WWW 2013*, 2013, pp. 367-374.

[5] H. Kajino, H. Arai, and H. Kashima, "Preserving worker privacy in crowdsourcing," *Data Mining and Knowledge Discovery*, vol. 28, no. 5-6, pp. 1314-1335, 2014.

[6] 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.

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

[8] J. Shu, X. Liu, X. Jia, K. Yang, and R. H. Deng, "Anonymous Privacy-Preserving Task Matching in Crowdsourcing," *IEEE Internet of Things Journal*, 2018, doi: 10.1109/JIOT.2018.2830784.

[9] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130-2145, 2018.

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

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

[12] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546-2559, 2016.

[13] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "TKSE: Trustworthy Keyword Search over Encrypted Data with Two-side Verifiability via Blockchain," *IEEE Access*, vol. 6, no. 1, pp. 31077-31087, 2018.

[14] 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.

[15] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient Encrypted Keyword Search for Multi-user Data Sharing," in *Proc. of ESORICS 2016*, 2016, pp. 173-195.

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

[17] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-Preserving Task Recommendation Services for Crowdsourcing," *IEEE Transactions on Services Computing*, 2018, doi: 10.1109/TSC.2018.2791601.

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

[19] J. H. Cheon, Y. Kim, and H. Yoon, "A New ID-based Signature with Batch Verification," *IACR Cryptology EPrint Archive*, pp. 131, 2004.

[20] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proc. of 2011 IEEE Symposium on Computers and communications*, 2011, pp. 850-855.

[21] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111-128, 2013.