Hindawi
Security and Communication Networks
Volume 2021, Article ID 6682456, 13 pages
https://doi.org/10.1155/2021/6682456

Research Article

Fine-Grained Task Access Control System for Mobile Crowdsensing

Jingwei Wang[1], **Xinchun Yin**[1,2] **and Jianting Ning**[3]

[1]*School of Information Engineering, Yangzhou University, Yangzhou 225127, China*
[2]*Guangling College, Yangzhou University, Yangzhou 225128, China*
[3]*Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350007, China*

Correspondence should be addressed to Xinchun Yin; xcyin@yzu.edu.cn

Received 19 November 2020; Revised 7 January 2021; Accepted 13 January 2021; Published 4 February 2021

Academic Editor: Athanasios V. Vasilakos

Mobile crowdsensing enables people to collect and process a massive amount of information by using social resources without any cost on sensor deployment or model training. Many schemes focusing on the problems of task assignment and privacy preservation have been proposed so far. However, the privacy-preserving of requesters and task access control, which are vital to mobile crowdsensing, is barely considered in the literature. To address the aforementioned issues, a fine-grained task access control system for mobile crowdsensing is proposed. In particular, the requester can decide the group of task performers who can access the task by utilizing attribute-based encryption technology. T he untrusted crowdsensing platform cannot obtain any sensitive information concerning the requester or the task, while the qualified task performers are capable of retrieving tasks within 0.85 ms. Security analysis and experimental results are presented to show the feasibility and efficiency of the proposed system.

1. Introduction

The evolution of mobile device promotes a new sensing paradigm called mobile crowdsensing. Integrated with a set of powerful sensors (e.g., camera, recorder, and GPS), the mobile device is capable of collecting different kinds of information for specific purposes [1]. Formally, mobile crowdsensing refers to a group of mobile users being coordinated to perform large-scale sensing tasks over urban environments by using their mobile devices [2]. Figure 1 shows a typical workflow of mobile crowdsensing. As we can see, there are three kinds of participants: the requester, the crowdsensing platform, and the task performer. The process of mobile crowdsensing consists of eight stages: design task, release task, scan task, choose task, resolve task, submit result, accept/refuse, and integration. If a requester wants to launch a task, he needs to submit his requirement about the task to the crowdsensing platform. The task performers in the crowdsensing platform scan the tasks and choose appropriate ones. After accomplishing the task, the task performers submit the results back to the platform, and the requester can decide whether the results are qualified. If yes, all the results are integrated, and the task is finished [3].

Due to the convenience of deployment and communication, mobile crowdsensing has been applied in a large number of scenarios such as smart transportation, environmental monitoring, and data labelling [4–7]. Although mobile crowdsensing provides an unprecedented solution for data collecting and processing, it brings many new challenges as well. Privacy preservation is one of the main problems that need to be considered when designing a mobile crowdsensing scheme. Most of the previous studies focus on protecting task performers [8, 9]. That is, they mainly consider how to preserve the privacy of task performers in the stage of submit result. For some specific tasks, the privacy leakage may also occur in the stage of release task

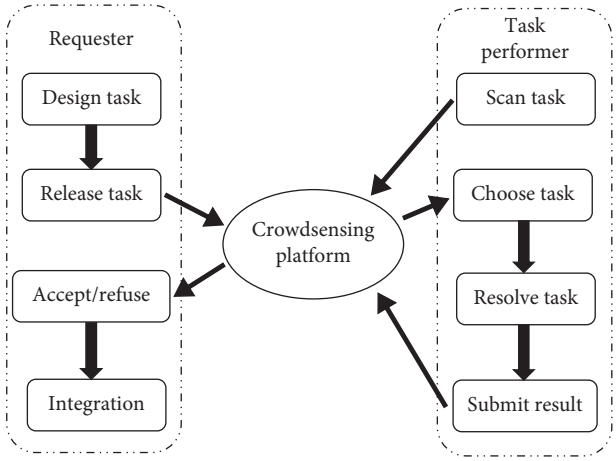


Figure 1: Workflow of mobile crowdsensing.

since the platform is not fully trusted. The platform may deduce the privacy of requesters according to their tasks. Besides, task access control is another problem that needs to be addressed. In mobile crowdsensing, the tasks are stored on the platform. Anyone in the system can access and accept the task. For the tasks that require users with specific conditions, it may lead to poor quality of task results if task performers are not qualified. Moreover, task access control can efficiently prevent the task information from been obtained by irrelevant entities (e.g., the crowdsensing platform). Thus, task access control is vital to mobile crowdsensing.

Attribute-based encryption (ABE) [10–12] is a promising data-sharing tool that provides fine-grained access control over the encrypted data. There are mainly two branches: ciphertext policy ABE (CP-ABE) [11] and key policy ABE (KP-ABE) [10]. In CP-ABE, the user secret key is associated with an attribute set, and the ciphertext is associated with an access policy. Only when the attribute set satisfies the access policy can the ciphertext be decrypted [13]. In KP-ABE, the situation is reversed. The user secret key and ciphertext are associated with an attribute set and access policy, respectively. Apparently, data owners in CP-ABE can decide the access policy, which makes CP-ABE more suitable for mobile crowdsensing. However, there still exist some problems if it is directly used in mobile crowdsensing [14].

A Motivating Story. Consider a researcher who would like to collect the heart rate record of people in region A for research. To achieve the purpose, he decides to issue his requirement to a crowdsensing platform. To encourage task performers in the system, everyone who contributes to this task will get some allowance. The researcher needs to encrypt the task with ABE technology, and the corresponding access policy is {"region A" AND "male"}. Thus, only male users in region A can accept this task. As the platform is not fully trusted, the access policy should not be included in the ciphertext. Otherwise, the platform may deduce the privacy of the researcher or task performers who accepted this task.

The crowdsensing platform is public, which contains a variety of task performers. Generally speaking, the more the task performers take part in the task, the better the result is. However, the requester may get a contrary result without a selecting mechanism for task performers. To address this problem, many task allocation algorithms have been proposed [2, 3, 15, 16]. Most of them let the crowdsensing platform perform the task allocation. Since the crowdsensing platform is not fully trusted, it may collect the attributes of the task and thus deduce the privacy of the requester or task performers who are qualified to accept this task. Therefore, we need to design a better solution that achieves fine-grained task access control without loss of confidentiality by using ABE technology. However, the following two issues arise.

By employing ABE technology, the content of the task is unknowable to the crowdsensing platform. However, the encrypted information is also difficult for task performers to browse. A task performer may need to download all the ciphertexts and try to decrypt them one by one, which is inefficient especially in the scenario of mobile crowdsensing where the devices are resource-limited [17]. How can we let the task performers get the appropriate tasks efficiently when the tasks are encrypted? Besides, in ABE, the access policy embedded in the ciphertext is sent to the crowdsensing platform. Since it is directly connected to the task, the crowdsensing platform may deduce the privacy of requesters and task performers from it. To protect the content of the ciphertext, ABE schemes supporting hidden access policy are needed.

1.1. Contributions. In this paper, we propose a fine-grained task access control system for mobile crowdsensing called FGTAC. The contributions are summarized as follows:

(1) *Fine-Grained Task Access Control*. Considering the situation that the requester needs to recruit a group of users with specific attributes for his task, a fine-grained task access control mechanism is needed. By employing ABE technology, the tasks are encrypted under the access policy defined by requesters. Only users with appropriate attributes can retrieve the encrypted task. In FGTAC, an AND-gate access policy with a wildcard is employed. Every attribute in the access policy has three kinds of possible values: positive, negative, and wildcard. As a result, the proposed system can achieve both fine-grained access control and flexibility.

(2) *Privacy Preservation*. One of the main purposes of our system is to preserve the privacy of requesters and task performers. To the best of our knowledge, barely of the existing literature pay attention to preserve the privacy of requesters. However, the crowdsensing platform is not fully trusted. If tasks are sent to the crowdsensing platform without any protection, the crowdsensing platform may deduce the information about requesters and thus threaten their privacy. To address this problem, we employ ABE technology in our system. Nevertheless, the access policy in ABE is sent to the crowdsensing platform together with the ciphertext, which may help the crowdsensing platform deduce the content

of the task. Thus, we employ Vie te's formulas [18] to hide access policy in the ciphertext.

(3) *Task Search*. As the tasks are encrypted and the access policies are hidden, the task performers cannot search the tasks as usual [19]. In our FGTAC system, an efficient search function is provided. The keyword is encrypted and stored in the ciphertext. Before the search algorithm is performed, the user keyword is transferred into the trapdoor. As a result, the crowdsensing platform cannot obtain any useful information from it. Moreover, the search algorithm only costs two bilinear pairing operations, which can be performed within 9 ms. The predecryption algorithm is performed once search algorithm finishes.

(4) *Ciphertext Predecryption*. In the scenario of mobile crowdsensing, devices cannot afford the high computational overhead of bilinear pairing operations. To reduce their computational overhead, we design the function of ciphertext predecryption in FGTAC. On the user side, the decryption algorithm only costs two exponentiation operations in $\mathbb{G}_T$, which can be performed efficiently by mobile devices.

1.2. Related Works. Task assignment and privacy-preserving are the two main problems in mobile crowdsensing [20, 21]. In task assignment, the crowdsensing platform needs to determine who is suitable to perform the task. To find the appropriate performers, many solutions are proposed in the literature [15, 22, 23]. Wu et al. [15] proposed two approximate task assignment algorithms for sweep coverage. Different from the schemes valuing the point of interest, they designed a model that weighs the quality of coverage from the perspective of the area. In Ref. [22], Karaliopoulos et al. employed opportunistic networking technology in choosing performers for crowdsensing tasks. In Ref. [23], the authors designed a crowdsensing system that mainly considers the location and profits of performers. However, the crowdsensing platform is hard to trust in reality. It may threaten the security of task performers and requesters if the crowdsensing platform obtains private information. Wang et al. [24] enabled the performers to select the personal privacy level so that they do not need to upload their true location to the crowdsensing platform. Ni et al. [16] introduced a strong privacy-preserving scheme for mobile crowdsensing called SPOON. Specifically, proxy re-encryption and BBS + signature are used to protect the tasks and results. Sei et al. [9] proposed a data collecting scheme, which is anonymized, by improving the traditional randomized response scheme. Unfortunately, none of the aforementioned schemes is designed to protect the privacy of requesters. In the workflow of mobile crowdsensing, both the privacy of performers and requests can be obtained by the crowdsensing platform through the task information and thus threaten their security.

To handle the problems above, ABE technology, which was firstly proposed by Sahai and Water [25], is a promising tool. As a well-known data-sharing tool, many ABE schemes aiming at settling the challenges of confidentiality and flexibility in data access control have been proposed [26–30]. To enable the users to search on the encrypted files, Sun et al. [26] proposed a fine-grained keyword search scheme. With the technologies of lazy re-encryption and proxy re-encryption, the expensive computation cost of ABE is delegated to other entities. As the access policy may leak the privacy, a partially hidden access policy ABE scheme is proposed in Ref. [27]. The access policy in their scheme only consists of the attribute names, and the corresponding values are not provided. Further in Ref. [28], a fully hidden access policy ABE scheme is introduced. Compared with Ref. [27], the scheme in Ref. [28] is more secure. To handle the expensive computation in ABE, Ning et al. set a secret value for users to protect ciphertexts in Ref. [29]. In the stage of decryption, the cloud server performs the decryption algorithm to generate the transformed ciphertext. The cloud server cannot get any useful information from the transformed ciphertext, while users can retrieve the encrypted data efficiently.

1.3. Organization. In Section 2, we state the preliminaries, including notations, bilinear map, complexity assumption, Vie te's formulas, access structure, and the definition of CP-ABE. The system model, definition, and security model are outlined in Section 3. Section 4 gives detailed construction and the security analysis of FGTAC. Section 5 evaluates the proposed system in terms of feature comparison, theoretical analysis, and experimental analysis. A brief conclusion is presented in Section 6.

2. Preliminaries

2.1. Notations. Some frequently used notations are given in Table 1.

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

(1) *Bilinearity*: $\forall x, y \in \mathbb{G}$ and $\alpha, \beta \in \mathbb{Z}_p$, the equation $e(x^\alpha, y^\beta) = e(x, y)^{\alpha\beta} = e(x^\beta, y^\alpha)$ holds.

(2) *Nondegeneracy*: $e(g, g) \neq 1$.

(3) *Computability*: $\forall x, y \in \mathbb{G}$, the computation of bilinear pairing $(e(x, y))$ can be performed efficiently.

2.3. Complexity Assumption. We use decisional q-bilinear Diffie-Hellman inversion (q-BDHI) problem in our scheme. A challenger selects two groups $\mathbb{G}$ and $\mathbb{G}_T$. Let g be a generator of $\mathbb{G}$ and $x \in \mathbb{Z}_p$. Given the tuple $y = (g, g^x, \ldots, g^{x^q})$ as input, it must be difficult for adversaries to distinguish $e(g, g)^{1/x}$ from $R \in \mathbb{G}_T$. An algorithm $\mathcal{B}$ has advantage ε in breaking it if $|\Pr[\mathcal{B}(y, e(g, g)^{1/x}) = 0] - \Pr[\mathcal{B}(y, e(g, g)^{1/x}) = 0]| \geq \varepsilon$.

Definition 1. (**q**-*BDHI)*. The decisional q-BDHI assumption holds if all probabilistic polynomial-time (PPT) algorithms have at most a negligible advantage in breaking it.

Table 1: Some notations used in this paper.

Notation	Description
$\mathbb{G}, \mathbb{G}_T$	Two multiplicative cyclic groups.
$\mathbb{Z}_p$	Group of integers modulo p.
e	The bilinear map: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$.
W	The AND-gate access policy $W = S_1 \Lambda S_2 \Lambda \cdots \Lambda S_L$.
S	The attribute set $S = \{S_1, S_2, \ldots, S_L\}$.
$S \vDash W$	The attribute set S satisfies the access policy W.
$[a]$	The number list $[1, 2, \ldots, a]$.
N_1, N_2, N_3	The maximum number of wildcards, positive attributes, and negative attributes, respectively.

2.4. Viete's Formulas. Consider two vectors $\overrightarrow{v} = (v_1, \ldots, v_L)$ and $\overrightarrow{z} = (z_1, \ldots, z_L)$. Vector $\overrightarrow{v}$ contains both alphabets and wildcards, and vector $\overrightarrow{z}$ only contains alphabets. Let $J = \{j_1, j_2, \ldots, j_n\} \subset \{1, 2, \ldots, L\}$ denotes the positions of wildcards in vector $\overrightarrow{v}$. Let $\prod_{j \in J}(i - j) = \sum_{k=0}^{n} \lambda_k i^k$, where λ_k are the coefficients, then we have $\sum_{i=1, i \notin J}^{L} v_i \prod_{j \in J}(i - j) = \sum_{k=0}^{n} \lambda_k \sum_{i=1}^{L} z_i i^k$ if $v_i = z_i v_i = *$ for $i \in [L]$. To hide the computation, we can choose a random group element H_i and put v_i, z_i as the exponents of H_i. Then, the above equation becomes $\prod_{i=1, i \notin J}^{L} H_i^{v_i \prod_{j \in J}(i-j)} = \prod_{k=0}^{n} (\prod_{i=0}^{L} H_i^{z_i i^k})$. By using Viète's formulas, we can construct the coefficient λ_k in aforementioned equation by $\lambda_{n-k} = (-1)^k \sum_{1 \le i_1 \le i_2 \le \cdots \le i_k \le n} j_{i_1}, j_{i_1}, \ldots, j_{i_1}, k \in [0, n]$, where $n = |J|$.

2.5. Access Structure. Let $W = S_1 \wedge S_2 \wedge \cdots \wedge S_L$ be an AND-gate access policy. Every attribute S_i has three kinds of states. '+' stands for positive, '−' stands for negative, and '∗' stands for wildcard. If a user wants to join the system. An attribute set S will be assigned to him, where $S = \{S_1, S_2, \ldots, S_L\}$. Similar to the access policy, every attribute has two kinds of states. '+' stands for positive and '−' stands for negative. The notation $S \vDash W$ means that S satisfies W.

2.6. CP-ABE definition. There are four algorithms in a basic ciphertext policy attribute-based encryption scheme: Setup, KeyGen, Encrypt, and Decrypt.

(1) *Setup* $(\lambda) \longrightarrow (\mathrm{PK}, \mathrm{MSK})$: It takes the secure parameter λ as input and outputs the public key PK and the master secret key MSK.

(2) *KeyGen* $(\mathrm{PK}, \mathrm{MSK}, S) \longrightarrow \mathrm{SK}$: It takes PKand MSK and an attribute set S as input and outputs the user secret key SK.

(3) *Encrypt* $(\mathrm{PK}, W, \mathrm{msg}) \longrightarrow \mathrm{CT}$: It takes PK, an access policy W and a message msg as input and outputs the ciphertext CT.

(4) *Decrypt* $(\mathrm{PK}, \mathrm{SK}, \mathrm{CT}) \longrightarrow \mathrm{msg}$: It takes PK, SK, and CT as input and outputs msg if $S \vDash W$. Otherwise, the algorithm outputs $\perp$.

3. System Model and Security

In this section, we give out the system model, definition, and security model of FGTAC.

3.1. System Model. In Figure 2, we show the system model of FGTAC. It consists of four entities: the key generation center (KGC), the crowdsensing platform, the requester, and task performers.

(1) *KGC* needs to generate the system parameters and distribute secret keys for task performers in FGTAC. As the core management entity, KGC is trusted by other entities.

(2) *Crowdsensing Platform* is responsible for storing the tasks and receiving the sensing data. When a task performer queries for a task, the crowdsensing platform will perform the predecryption algorithm to relieve his computational overhead. It is semi-trusted in the sense that the crowdsensing platform may collect private information for its own interest.

(3) *Requester* is the entity that issues crowdsensing tasks via the crowdsensing platform to collect sensing data. Since the requester is able to issue tasks at will, we assume the requester has sufficient computation ability. In addition, the requester may have some special requirements on task performers and do not hope the crowdsensing platform knows the content of his task. Thus, the requester needs to encrypt his tasks with ABE technology.

(4) *Task Performers* are workers in FGTAC. They would like to accomplish the requesters' tasks to get the allowance. The tasks are encrypted by ABE technology. All the task performers in the system are assigned a set of attributes. Only task performers whose attribute set satisfies the access policy can retrieve the task in the crowdsensing platform. The task performers are untrusted for they may collude to get the tasks that they are not qualified to obtain.

3.2. Definition. According to the system model, the definition of the proposed system is designed as follows:

(1) *Setup* $(\lambda) \longrightarrow (\mathrm{PK}, \mathrm{MSK})$: The setup algorithm is run by KGC. It takes a secure parameter λ as input and outputs the public key PK and the master secret key MSK. KGC keeps MSK private and publishes PK.

(2) *UserReg* $(\mathrm{PK}) \longrightarrow (\mathrm{UK}, \mathrm{USK})$: The user registration algorithm is run by task performers. It takes PK as input and outputs the user public key UK and the user master secret key USK. The task performers keep USK private and publish UK.

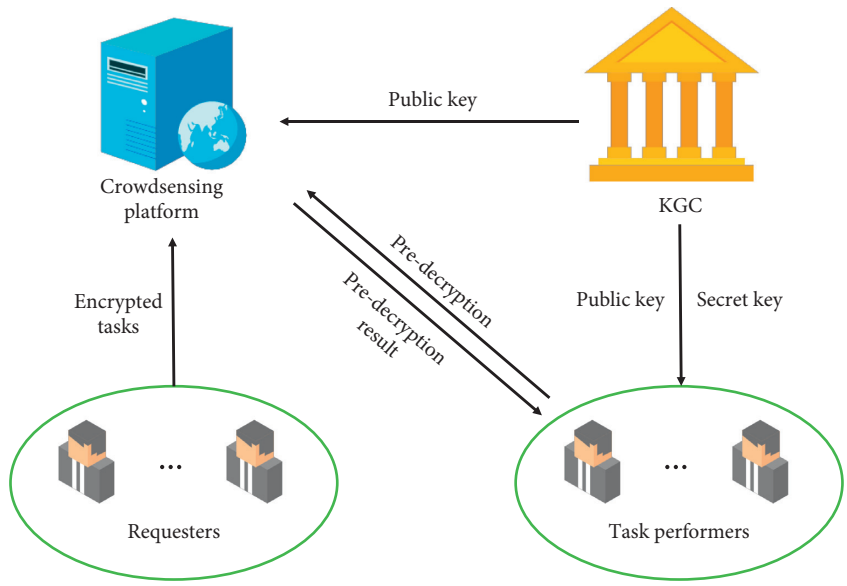


Figure 2: System model of FGTAC.

(3) *KeyGen* (PK, MSK, UK, S) ⟶ SK: The key generation algorithm is run by KGC. It takes PK, MSK, and UK, and an attribute set S as input, outputs the user secret key SK. KGC distributes SK to corresponding task performers in the system.

(4) *Encrypt* (PK, W, msg, KW) ⟶ CT: The encryption algorithm is run by the requester. It takes PK, an access policy W, a message msg, and the keyword KW defined by the requester as input and outputs the ciphertext CT. Note that all the ciphertexts are transferred to the crowdsensing platform for storage.

(5) *Trapdoor* (PK, KW_u) ⟶ td: The trapdoor generation algorithm is run by task performers. It takes PK and the user keyword KW_u as input and outputs the trapdoor td.

(6) *Search* (PK, CT, td) ⟶ 1 or 0: The search algorithm is run by the crowdsensing platform. It takes PK, CT, and td as input and outputs 1 if td matches CT. Otherwise, the algorithm outputs 0.

(7) *PreDecrypt* (PK, SK, CT) ⟶ preCT or ⊥: The predecryption algorithm is run by the crowdsensing platform. It takes PK, SK and CT as input and outputs the predecrypted ciphertext preCT if the attribute set S in SK satisfies W in CT. Otherwise, the algorithm outputs ⊥.

(8) *Decrypt* (PK, preCT, USK) ⟶ msg: The decryption algorithm is run by task performers. It takes PK, preCT, and USK as input and outputs the message msg.

3.3. Security Model. According to the system model in Section 3.1, we say that KGC is fully trusted, the crowdsensing platform is semitrusted, and requesters and task performers are untrusted. We define the security of FGTAC by the security games as follows:

3.3.1. Chosen-Plaintext Attack Security. The security game of chosen-plaintext attack (CPA) is designed between a challenger C and an active adversary (malicious user) A. The interaction is as follows:

Initialization: $\mathcal{A}$ declares the challenge access policy W^*, challenge user public key UK^*, and sends them to $\mathcal{C}$.

Setup: C first runs the Setup algorithm to get the public key PK and the master secret key MSK. After that, PK is sent to A.

Query Phase 1: Let T be an empty set and $c = 0$ be an integer counter. A launches queries as follows:

Generate.SK(S): C sets $c = c + 1$. It runs the *UserReg* algorithm to get the tuple of user public key and user secret key (UK, USK), and runs the *KeyGen* algorithm with attribute set S and UK to get the user secret key SK. C stores $(c, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$ to T. Besides, the situation that $S \models W^*$ is not allowed.
Corrupt.SK (i): C checks whether T contains $(i, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$. If it is, C returns (UK, USK) to A. Otherwise, C returns ⊥.

Challenge: A declares two equal-length plaintexts m_0, m_1 and sends them to C. C randomly selects $b \in \{0, 1\}$ and runs the *Encrypt* algorithm to get challenge ciphertext CT_b^*. It sends CT_b^* to A.

Query Phase 2: A adaptively queries SK as in *Query Phase 1*.

Guess: A outputs a guess $b' \in \{0, 1\}$ for b.

The probability for A to win the security game is $|\Pr[b = b'] - 1/2|$.

Definition 2. (CPA security). The proposed system is CPA secure if all PPT adversaries have at most a negligible probability in breaking CPA security game.

3.3.2. Chosen Keyword Attack Security. The security game of chosen keyword attack (CKA) is designed between a challenger C and an active adversary (malicious user) A. The interaction is as follows:

Setup: C first runs the *Setup* algorithm to get the public key PK and the master secret key MSK. Then, C returns PK to A.

Query Phase 1: Let T_1, T_2 be two empty sets, $c = 0$ be an integer counter. A launches queries as follows:

Generate.Trapdoor(KW): C sets $c = c + 1$. It runs the *Trapdoor* algorithm with keyword KW to get the trapdoor td. C stores $(c, \mathrm{KW}, \mathrm{td})$ to T_1.
Corrupt.Trapdoor(i): C checks whether T_1 contains $(i, \mathrm{KW}, \mathrm{td})$. If it is, C returns td to A and sets $T_2 = T_2 \cup \mathrm{KW}$. Otherwise, C returns $\perp$.

Challenge: A declares two equal-length keywords kw_0, kw_1 and sends them to C. Note that $kw_1, kw_2 \notin T_2$. C randomly selects $b \in \{0, 1\}$ and runs the *Encrypt* algorithm to get challenge ciphertext CT_b^*. It returns CT_b^* to A.

Guess: A outputs a guess $b' \in \{0, 1\}$ for b.

The probability for A to win the security game is $|\Pr[b = b'] - 1/2|$.

Definition 3. (CKA security). The proposed system is CKA secure if all PPT adversaries have at most a negligible probability in breaking CKA security game.

3.3.3. Secret Key Exposure Attack Security. The security game of secret key exposure attack is designed between a challenger C and an active adversary (semitrusted crowdsensing platform) A. The interaction is as follows:

Initialization: A declares the challenge access policy W^* and sends it to C.

Setup: C first runs the *Setup* algorithm to get the public key PK and the master secret key MSK. C returns PK to A.

Query Phase 1: Let T_1, T_2 be two empty sets, $c_1 = 0, c_2 = 0$ be two integer counters. A launches queries as follows:

Generate.SK (S): C sets $c_1 = c_1 + 1$. It runs the *UserReg* algorithm to get $(\mathrm{UK}, \mathrm{USK})$, and runs the *KeyGen* algorithm with attribute set S and UK to get the secret key SK. It stores $(c_1, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$ to T_1. Besides, the situation that $S \models W^*$ is not allowed.
Generate.CT(W, KW): C sets $c_2 = c_2 + 1$. It runs the *Encrypt* algorithm to get the ciphertext CT. It stores $(c_2, W, \mathrm{KW}, \mathrm{CT})$ to T_2. Besides, the situation that $W = W^*$ is not allowed.
Corrupt (i, j): C checks whether T_1 contains $(i, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$ and T_2 contains $(j, W, \mathrm{KW}, \mathrm{CT})$. If it is, C returns the tuple $(\mathrm{SK}, \mathrm{CT})$ to A. Otherwise, C returns $\perp$.

PreDecrypt$(\mathrm{SK}, \mathrm{CT})$: A runs the *PreDecrypt* algorithm to get the predecrypted ciphertext preCT.

Challenge: A declares two equal-length plaintexts m_0, m_1 to C. C randomly selects $b \in \{0, 1\}$ and runs the *Encrypt* algorithm to get ciphertext CT_b^*. C runs the *UserReg* algorithm and *KeyGen* algorithm to get SK. It sends $\mathrm{SK}, \mathrm{CT}_b^*$ to A, and A performs *PreDecrypt* algorithm to get the challenge predecrypted ciphertext preCT_b.

Query Phase 2: A adaptively queries SK and CT as in *Query Phase 1*.

Guess: A outputs a guess $b' \in \{0, 1\}$ for b.

The probability for A to win the security game is $|\Pr[b = b'] - 1/2|$.

Definition 4. (Secret key exposure attack security). The proposed system is secure against secret key exposure attack if all PPT adversaries have at most a negligible probability in breaking the secret key exposure attack security game.

3.3.4. Collude Attack Security. The security game of collude attack is designed between a challenger C and an active adversary (malicious user) A. The interaction is as follows:

Initialization: A declares the challenge access policy W^* and sends it to C.

Setup: C first runs the *Setup* algorithm to get the public key PK and the master secret key MSK. C returns PK to A.

Query Phase: Let T be an empty set, $c = 0$ be an integer counter. A launches queries as follows:

Generate.SK(S): C sets $c = c + 1$. It runs the *UserReg* algorithm to get the tuple of user public key and user secret key $(\mathrm{UK}, \mathrm{USK})$, and runs the *KeyGen* algorithm with attribute set S and UK to get the user secret key SK. C stores $(c, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$ to T. Besides, the situation that $S \models W^*$ is not allowed.
Corrupt.SK(i): C checks whether T contains $(i, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$. If it is, C returns $(\mathrm{UK}, \mathrm{USK})$ to A. Otherwise, C returns $\perp$.

Forgery: A output a forged secret key for the challenge access policy W^*.

Definition 5. (Collude attack security). The proposed system is collude attack secure if all PPT adversaries have at most a negligible probability in breaking the collude attack security game.

4. The Proposed System

4.1. Construction. The attribute universe contains L attributes. For each of the attributes, there are three kinds of states. '+' stands for positive, '−' stands for negative, and '∗' stands for wildcard. Let N_1, N_2, and N_3 be the three upper bounds defined as follows: N_1 is the maximum

number of the wildcard ($N_1 \le L$); N_2 is the maximum number of positive attributes ($N_2 \le L$); N_3 is the maximum number of negative attributes ($N_3 \le L$). The detailed construction of the proposed system consists of the following eight algorithms:

Setup $(\lambda) \longrightarrow (\mathrm{PK}, \mathrm{MSK})$. The setup algorithm is run by *KGC*. It takes secure parameter λ as input, generates a bilinear pairing group $(\mathbb{G}, \mathbb{G}_T, p, e)$ and sets $n = N_1 + 3$. It randomly selects $g, h, v, w \in \mathbb{G}, \alpha \in \mathbb{Z}_p$. For $\forall i \in [n]$, the algorithm selects $r_i, u_i, t_i \in \mathbb{Z}_p$ randomly and calculates $R_i = g^{r_i}, U_i = g^{u_i}, T_i = v^{t_i}, T_i' = g^{t_i}$ and $Y = e(g,g)^{\alpha}$. H: $\{0,1\}^* \longrightarrow \mathbb{Z}_p$ is a hash function. The public key PK and the master secret key MSK are set as

$$\begin{aligned} \mathrm{PK} &= \langle \mathbb{G}, \mathbb{G}_T, p, e, g, h, v, w, \{R_i, U_i, T_i, T_i'\}_{i\in[n]}, Y, H\rangle, \\ \mathrm{MSK} &= \langle \alpha, \{r_i, u_i, t_i\}_{i\in[n]}\rangle. \end{aligned} \tag{1}$$

UserReg $(\mathrm{PK}) \longrightarrow (\mathrm{UK}, \mathrm{USK})$. The user registration algorithm is run by task performers in system. A task performer randomly selects $\beta \in \mathbb{Z}_p$ and calculates g^{β}. The user public key UK and the user master secret key USK are set as

$$\begin{aligned} \mathrm{UK} &= g^{\beta}, \\ \mathrm{USK} &= \beta. \end{aligned} \tag{2}$$

KeyGen $(\mathrm{PK}, \mathrm{MSK}, \mathrm{UK}, S) \longrightarrow \mathrm{SK}$: The key generation algorithm is run by KGC. A task performer needs to submit his attribute set S to KGC to get his user secret key. Let the attribute set contains s_2 positive attributes and s_3 negative attributes. $V = \{v_1, v_2, \ldots, v_{s_2}\}$ and $Z = \{z_1, z_2, \ldots, z_{s_3}\}$ represent the positions of positive attributes and negative attributes, respectively. According to Vie te's formulas, for the positive attribute position set V and negative attribute position Z, it sets

$$\begin{aligned} v_k &= -\sum_{i\in V} i^k, \quad k \in [0, N_1], \\ z_k &= +\sum_{i\in V} i^k, \quad k \in [0, N_1]. \end{aligned} \tag{3}$$

The algorithm creates vectors $\overrightarrow{x_V}$ and $\overrightarrow{x_Z}$ as

$$\begin{aligned} \overrightarrow{x_V} &= \left(v_0, v_1, \ldots, v_{N_1}, 1, 0, \frac{1}{2}\right), \\ \overrightarrow{x_Z} &= \left(z_0, z_1, \ldots, z_{N_1}, 0, 1, \frac{1}{2}\right). \end{aligned} \tag{4}$$

The key generation algorithm randomly selects $r \in \mathbb{Z}_p$, and computes $K_0 = \mathrm{UK}^{\alpha} w^{1/r}$. For $\forall i \in [n]$, it sets $K_{i,1} = g^{\overrightarrow{x_{V_i}}/r}$, $K_{i,1}' = g^{\overrightarrow{x_{Z_i}}/r}$, $K_{i,2} = R_i^{-t_i}$, and $K_{i,3} = (U_i h)^{r_i} v^{-(\overrightarrow{x_{V_i}} + t\overrightarrow{x_{Z_i}})/r}$. The user secret key is set as follows:

$$\mathrm{SK} = \langle K_0, \{K_{i,1}, K_{i,1}', K_{i,2}, K_{i,3}\}_{i\in[n]}\rangle. \tag{5}$$

Encrypt $(\mathrm{PK}, W, \mathrm{msg}, \mathrm{KW}) \longrightarrow \mathrm{CT}$: The encryption algorithm is run by the requester. Let the access policy W contains w_1 wildcards, w_2 positive attributes and w_3 negative attributes. $J' = \{j_1', j_2', \ldots, j_{w_1}'\}$, $V' = \{v_1', v_2', \ldots, v_{w_2}'\}$, and $Z' = \{z_1', z_2', \ldots, z_{w_3}'\}$ represent the positions where '$*$', '$+$', and '$-$' attributes are, respectively. According to Vie te's formulas, the algorithm computes the coefficients $\{a_k\}_{k\in[0,n_1]}$ as

$$\begin{aligned} a_{n_1} &= 1, \\ a_{n_1-1} &= -\left(j_1' + j_2' + \cdots + j_{n_1}'\right), \\ a_{n_1-2} &= \left(j_1' j_2' + j_1' j_3' + \cdots + j_{n_1-1}' j_{n_1}'\right), \\ &\cdots \\ a_0 &= -\left(j_1', j_2', \ldots, j_{n_1}'\right). \end{aligned} \tag{6}$$

It computes as follows:

$$\begin{aligned} \Pi_{V'} &= +\sum_{i\in V'} \prod_{j_k\in J'} (i - j_k), \\ \Pi_{Z'} &= +\sum_{i\in Z'} \prod_{j_k\in J'} (i - j_k). \end{aligned} \tag{7}$$

The vector $\overrightarrow{v}$ is set as

$$\overrightarrow{v} = \left(a_0, a_1, \ldots, a_{n_1}, 0_{n_1+1}, \ldots, 0_{N_1}, \Pi_{V'}, \Pi_{Z'}, 1\right). \tag{8}$$

The encryption algorithm randomly selects $s \in \mathbb{Z}_p$ and computes $C = \mathrm{msg} Y^{1/s}, C_0 = g^{1/s}$. For $\forall i \in [n]$, it computes $C_{i,1} = w^{\overrightarrow{v}} T_i$, $C_{i,2} = U_i h$, and $C_{3,i} = T_i'$. According to the keyword KW, the algorithm calculates $C_4 = h^{H(\mathrm{KW})/s}$ The ciphertext CT is set as follows:

$$\mathrm{CT} = \langle C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i\in[n]}, C_4\rangle. \tag{9}$$

Trapdoor $(\mathrm{PK}, \mathrm{KW}_u) \longrightarrow \mathrm{td}$: The trapdoor generation algorithm is run by task performers. The trapdoor is used to search the tasks about the keyword KW_u specified by task performers. The algorithm randomly selects $t \in \mathbb{Z}_p$ and computes $\mathrm{TD}_1 = g^t$, and $\mathrm{TD}_2 = h^{H(\mathrm{KW}_u)t}$. The trapdoor is set as follows:

$$\mathrm{td} = \langle \mathrm{TD}_1, \mathrm{TD}_2\rangle. \tag{10}$$

Search $(\mathrm{PK}, \mathrm{CT}, \mathrm{td}) \longrightarrow 1$ or 0: The search algorithm is run by the crowdsensing platform. On receiving td, the crowdsensing platform checks if $e(C_0, \mathrm{TD}_2) = e(C_4, \mathrm{TD}_1)$ holds. If yes, the crowdsensing platform performs the *PreDecrypt* algorithm.

PreDecrypt $(\mathrm{PK}, \mathrm{SK}, \mathrm{CT}) \longrightarrow \mathrm{preCT}$ or $\perp$: The predecryption algorithm is run by the crowdsensing platform. If S embedded in SK satisfies the W inCT, the algorithm returns the predecrypted ciphertext preCT. Otherwise, it returns $\perp$. The computing process is as follows:

$$P = \prod_{i=1}^{n} \left(e\left(C_{i,1}, K_{i,1}\right) e\left(C_{i,1}, K_{i,1}'\right) e\left(C_{i,2}, K_{i,2}\right) e\left(C_{i,3}, K_{i,3}\right)\right),$$

$$\text{preCT} = \frac{e\left(C_0, K_0\right)}{P}. \tag{11}$$

Decrypt(PK, preCT, USK) $\longrightarrow$ msg: The decryption algorithm is run by task performers. To retrieve the predecrypted ciphertext preCT, the algorithm performs the following operations:

$$\text{msg} = \frac{C}{\text{preCT}^{1/\beta}}. \tag{12}$$

Note that the proposed system supports multikeyword search if KW in *Encrypt* and KW_u in *Trapdoor* consist of a set of keywords.

4.2. Security Analysis

4.2.1. Chosen-Plaintext Attack Security

Theorem 1. *The proposed system is chosen-plaintext attack secure for PPT adversaries (malicious users).*

Proof. Let A be a PPT adversary who attacks the FGTAC system. The security game is played between A and a simulator $\mathcal{B}$ as follows:

Initialization: A declares the challenge access policy W^*, challenge user public key $\text{UK}^* = g^{\beta_1}$, and sends them to $\mathcal{B}$.

Setup: $\mathcal{B}$ sets $n = N_1 + 3$ and randomly selects $g, h, v, w \in \mathbb{G}$, $\alpha \in \mathbb{Z}_p$. Then, it selects $r_i, u_i, t_i \in \mathbb{Z}_p$ and calculates $R_i = g^{r_i}$, $U_i = g^{u_i}$, $T_i = v^{t_i}$, and $T_i' = g^{t_i}$ where $i \in [n]$. H: $\{0,1\}^* \longrightarrow \mathbb{Z}_p$ is a hash function. The public key is $\text{PK} = \langle g, h, v, w, \{R_i, U_i, T_i, T_i'\}_{i\in[n]}, Y, H\rangle$ and the master secret key is $\text{MSK} = \langle \alpha, \{r_i, u_i, t_i\}_{i\in[n]}\rangle$. PK is sent to A.

Query Phase 1: Let T be an empty set, $c = 0$ be an integer counter. A launches queries as follows:

Generate.SK(S): $\mathcal{B}$ sets $c = c + 1$. It runs the *UserReg* algorithm to generate $\text{UK} = g^{\beta}$ and $\text{USK} = \beta$. Then, $\mathcal{B}$ runs the *KeyGen* algorithm to get the corresponding user secret key $\text{SK} = \langle K_0 = \text{UK}^{\alpha} w^{1/r}, \{K_{i,1} = g^{\overrightarrow{xV_i}/r}, K_{i,1}' = g^{\overrightarrow{xZ_i}/r}, K_{i,2} = R_i^{-t_i}, K_{i,3} = (U_i h)^{r_i} v^{-(\overrightarrow{xV_i} + t\overrightarrow{xZ_i})/r}\}_{i\in[n]}\rangle$ according to the attribute set S and UK. It stores $(c, \text{UK}, \text{USK}, S, \text{SK})$ to T. Besides, the situation that $S \models W^*$ is not allowed.

Corrupt.SK(i): $\mathcal{B}$ checks whether T contains $(i, \text{UK}, \text{USK}, S, \text{SK})$. If it is, $\mathcal{B}$ sends (UK, USK) to A. Otherwise, $\mathcal{B}$ returns $\perp$.

Challenge: A declares two equal-length plaintexts m_0, m_1 and sends them to $\mathcal{B}$. $\mathcal{B}$ randomly selects $b \in \{0,1\}$ and runs the *Encrypt* algorithm to get challenge ciphertext $\text{CT}_b^* = \langle C = m_b Y^{1/s}, C_0 = g^{1/s}, \{C_{i,1} = w^{\overrightarrow{v}} T_i, C_{i,2} = U_i h, C_{3,i} = T_i'\}_{i\in[n]}\rangle$. It sends CT_b^* to A.

Query Phase 2: A adaptively queries SK as in *Query Phase 1*.

Guess: A outputs a guess $b' \in \{0,1\}$ for b.

A wins the game if b' is the same as b. Since the distribution of PK, UK, SK, and CT are identical to that in the real world, the advantage of the adversary to win this game is ignorable. □

4.2.2. Chosen Keyword Attack Security

Theorem 2. *The proposed system is chosen keyword attack secure for PPT adversaries (malicious users).*

Proof. Let A be a PPT adversary who attacks FGTAC system. The security game is played between A and a simulator $\mathcal{B}$ as follows:

Setup: $\mathcal{B}$ sets $n = N_1 + 3$ and randomly selects $g, h, v, w \in \mathbb{G}$, $\alpha \in \mathbb{Z}_p$. Then, it selects $r_i, u_i, t_i \in \mathbb{Z}_p$ and calculates $R_i = g^{r_i}$, $U_i = g^{u_i}$, $T_i = v^{t_i}$, and $T_i' = g^{t_i}$ where $i \in [n]$. H: $\{0,1\}^* \longrightarrow \mathbb{Z}_p$ is a hash function. The public key is $\text{PK} = \langle g, h, v, w, \{R_i, U_i, T_i, T_i'\}_{i\in[n]}, Y, H\rangle$ and the master secret key is $\text{MSK} = \langle \alpha, \{r_i, u_i, t_i\}_{i\in[n]}\rangle$. Then, $\mathcal{B}$ returns PK to A.

Query Phase 1: Let T_1, T_2 be two empty sets, $c = 0$ be an integer counter. A launches queries as follows:

Generate.Trapdoor(KW): $\mathcal{B}$ sets $c = c + 1$. It randomly selects $t \in \mathbb{Z}_p$ and computes $TD_1 = g^t$ and $TD_2 = h^{H(KW_u)t}$.td is set as $\langle \text{TD}_1, \text{TD}_2\rangle$. $\mathcal{B}$ stores (c, KW, td) to T_1.

Corrupt.Trapdoor(i): $\mathcal{B}$ checks whether T_1 contains $(i, \text{KW}, \text{td})$. If it is, $\mathcal{B}$ returns td to A and sets $T_2 = T_2 \cup \text{KW}$. Otherwise, $\mathcal{B}$ returns $\perp$.

Query Phase 2: A adaptively queries td as in *Query Phase 1*.

Challenge: A declares two equal-length keywords kw_0, kw_1 and sends them to $\mathcal{B}$. Note that $kw_0, kw_1 \notin T_2$. $\mathcal{B}$ randomly selects $b \in \{0,1\}$ and runs the *Encrypt* algorithm to select $s \in \mathbb{Z}_p$ and get the challenge ciphertext $\text{CT}_b^* = \{C = Y^{1/s}, C_4 = h^{H(KW_b)/s}\}$ to A.

Guess: A outputs a guess $b' \in \{0,1\}$ for b.

A wins the game if b' is the same as b. Since the distribution of PK, td, and CT are identical to that in the real world, the advantage of adversary to win this game is ignorable. □

4.2.3. Secret Key Exposure Attack Security

Theorem 3. *The proposed system is secret key exposure attack secure for a PPT adversary (semitrusted crowdsensing platform).*

Proof. Let A be a PPT adversary who attacks the FGTAC system. The security game is played between A and a simulator $\mathcal{B}$ as follows:

Initialization: The adversary A declares the challenge access policy W^* and sends it to $\mathcal{B}$.

Setup: $\mathcal{B}$ sets $n = N_1 + 3$ and randomly selects $g, h, v, w \in \mathbb{G}$, $\alpha \in \mathbb{Z}_p$. Then it selects $r_i, u_i, t_i \in \mathbb{Z}_p$ and calculates $R_i = g^{r_i}, U_i = g^{u_i}, T_i = v^{t_i}, T_i' = g^{t_i}$ where $i \in [n]$. H: $\{0,1\}^* \longrightarrow \mathbb{Z}_p$ is a hash function. The public key is $\mathrm{PK} = \langle g, h, v, w, \{R_i, U_i, T_i, T_i'\}_{i\in[n]}, Y, H\rangle$ and the master secret key is $\mathrm{MSK} = \langle \alpha, \{r_i, u_i, t_i\}_{i\in[n]}\rangle$. Then, $\mathcal{B}$ sends PK to A.

Query Phase 1: Let T_1 and T_2 be two empty sets, $c_1 = 0$ and $c_2 = 0$ be two integer counters. A launches queries as follows:

Generate.SK(S): $\mathcal{B}$ sets $c_1 = c_1 + 1$. It runs the *UserReg* algorithm to get the user public key $\mathrm{UK} = g^{\beta}$ and the user secret key $\mathrm{USK} = \beta$. Then, $\mathcal{B}$ runs the *KeyGen* algorithm to get the corresponding user secret key $\mathrm{SK} = \langle K_0 = \mathrm{UK}^{\alpha} w^{1/r}, \{K_{i,1} = g^{\overrightarrow{xv_i}}/r, K_{i,1}' = g^{\overrightarrow{xz_i}/r}, K_{i,2} = R_i^{-t_i}, K_{i,3} = (U_i h)^{r_i} v^{-(\overrightarrow{xv_i} + t\overrightarrow{xz_i})/r}\}_{i\in[n]}\rangle$ according to the attribute set S and UK. It stores $(c_1, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$ to T_1. Besides, the situation that $S \vDash W^*$ is not allowed.

Generate.CT(W, KW): $\mathcal{B}$ sets $c_2 = c_2 + 1$. It runs the *Encrypt* algorithm to get the ciphertext $\mathrm{CT} = \langle C = \mathrm{msg}Y^{1/s}, C_0 = g^{1/s}, \{C_{i,1} = w^{\overrightarrow{v}} T_i, C_{i,2} = U_i h, C_{i,3} = T_i'\}_{i\in[n]}, C_4 = h^{H(\mathrm{KW})/s}\rangle$. It stores $(c_2, W, \mathrm{KW}, \mathrm{CT})$ to T_2. Besides, the situation that the access policy $W = W^*$ is not allowed.

Corrupt (i, j): $\mathcal{B}$ checks whether T_1 contains $(i, \mathrm{UK}, \mathrm{USK}, S, \mathrm{SK})$ and T_2 contains $(j, W, \mathrm{KW}, \mathrm{CT})$. If it is, $\mathcal{B}$ returns the tuple $(\mathrm{SK}, \mathrm{CT})$ to A. Otherwise, $\mathcal{B}$ returns $\perp$.

PreDecrypt $(\mathrm{SK}, \mathrm{CT})$: A runs the *PreDecrypt* algorithm to compute $P = \prod_{i=1}^{n} (e(C_{i,1}, K_{i,1}) e(C_{i,1}, K_{i,1}') e(C_{i,2}, K_{i,2}) e(C_{i,3}, K_{i,3}))$ and set the predecrypted ciphertext $\mathrm{preCT} = e(C_0, K_0)/P$.

Challenge: A declares two equal-length plaintexts m_0, m_1 and sends them to $\mathcal{B}$. $\mathcal{B}$ randomly selects $b \in \{0, 1\}$ and runs the *Encrypt* algorithm to get $\mathrm{CT}_b^* = \langle C = m_b Y^{1/s}, C_0 = g^{1/s}, \{C_{i,1} = w^{\overrightarrow{v}} T_i, C_{i,2} = U_i h, C_{i,3} = T_i'\}_{i\in[n]}\rangle$. $\mathcal{B}$ runs the *UserReg* algorithm and *KeyGen* algorithm to get $\mathrm{SK} = \langle K_0 = \mathrm{UK}^{\alpha} w^{1/r}, \{K_{i,1} = g^{\overrightarrow{xv_i}/r}, K_{i,1}' = g^{\overrightarrow{xz_i}/r}, K_{i,2} = R_i^{-t_i}, K_{i,3} = (U_i h)^{r_i} v^{-(\overrightarrow{xv_i} + t\overrightarrow{xz_i})/r}\}_{i\in[n]}\rangle$. $\mathcal{B}$ sends $\mathrm{SK}, \mathrm{CT}_b^*$ to A, and A performs *PreDecrypt* algorithm to get the challenge predecrypted ciphertext $\mathrm{preCT} = e(C_0, K_0)/P$.

Query Phase 2: A adaptively queries SK and CT as in phase 1.

Guess: A outputs a guess $b' \in \{0, 1\}$ for b.

A wins the game if b' is the same as b. Since the distribution of PK, SK, UK, and CT are identical to that in the real world, the ciphertext is protected by the USK and the adversary A cannot obtain any sensitive information. □

4.2.4. Collude Attack Security

Theorem 4. *The proposed system is colluding attack secure for PPT adversaries (malicious users).*

Proof. Let A be a PPT adversary who attacks the FGTAC system. The security game is played between A and a simulator $\mathcal{B}$ as follows:

Initialization: A declares the challenge access policy W^* and challenge user public key $\mathrm{UK}^* = g^{\beta_1}$ and sends them to $\mathcal{B}$.

Setup: $\mathcal{B}$ sets $n = N_1 + 3$ and randomly selects $g, h, v, w \in \mathbb{G}$, $\alpha \in \mathbb{Z}_p$. Then, it selects $r_i, u_i, t_i \in \mathbb{Z}_p$ and calculates $R_i = g^{r_i}, U_i = g^{u_i}, T_i = v^{t_i}, T_i' = g^{t_i}$ where $i \in [n]$. H: $\{0,1\}^* \longrightarrow \mathbb{Z}_p$ is a hash function. The public key is $PK = \langle g, h, v, w, \{R_i, U_i, T_i, T_i'\}_{i\in[n]}, Y, H\rangle$, and the master secret key is $\mathrm{MSK} = \langle \alpha, \{r_i, u_i, t_i\}_{i\in[n]}\rangle$. After thatPK is sent to A.

Query Phase: Let T be an empty set, $c = 0$ be an integer counter. A launches queries as follows:

Generate.SK(S): C sets $c = c + 1$. It runs the *UserReg* algorithm to generate and . Then, runs the *KeyGen* algorithm to get the corresponding user secret key $\mathrm{SK} = \langle K_0 = \mathrm{UK}^{\alpha} w^{1/r}, \{K_{i,1} = g^{\overrightarrow{xv_i}/r}, K_{i,1}' = g^{\overrightarrow{xz_i}/r}, K_{i,2} = R_i^{-t_i}, K_{i,3} = (U_i h)^{r_i} v^{-(\overrightarrow{xv_i} + t\overrightarrow{xz_i})/r}\}_{i\in[n]}\rangle$ according to the attribute set and . It stores to . Besides, the situation that is not allowed.

Corrupt.SK (i): checks whether contains . If it is, sends to . Otherwise, returns .

Forgery: output a forged secret key for the challenge access policy .

wins the game if the forged secret key is valid. Since all the secret keys in are randomized by the random number . The advantage for the adversary to forge a valid secret key is ignorable. □

5. Performance Evaluation

5.1. Feature Comparison. The performance evaluation is performed among Ref. [26, 28, 30–32] and our system. We mainly consider the following features in Table 2: hidden access policy, ciphertext search, predecryption, and access structure. Although ABE technology, which provides fine-grained access control, is employed in all the aforementioned schemes, only Ref. [28, 30, 32] and our system support hiding the access policy. Malicious users in Ref. [26, 31] may deduce sensitive information about the ciphertext from the access policy. Compared with Ref. [28, 30, 31], [26, 32], and our system enables users to

Table 2: Feature comparison.

	[26]	[28]	[30]	[31]	[32]	Our system
Hidden access policy	✗	✓	✓	✗	✓	✓
Ciphertext search	✓	✗	✗	✗	✓	✓
Pre-decryption	✗	✗	✓	✗	✗	✓
Access structure	+, −, ∗	+, −, ∗	Tree	And, Or	And	+, −, ∗

✓ represents the scheme owns the feature. ✗ represents the scheme does not own the feature.

Table 3: Comparison in storage.

	PK	MSK	SK	CT	Trapdoor
[26]	$(9n+1)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$(9n+1)\lvert\mathbb{Z}_p\rvert$	$(6n+1)\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$	—	$(6n+1)\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_p\rvert$
[28]	$(8n+30)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}\rvert + (8n+28)\lvert\mathbb{Z}_p\rvert$	$(4n+14)\lvert\mathbb{G}\rvert$	$(4n+14)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	—
Our system	$(4n+20)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$(3n+13)\lvert\mathbb{Z}_p\rvert$	$(4n+17)\lvert\mathbb{G}\rvert$	$(3n+14)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$2\lvert\mathbb{G}\rvert$

$\lvert\mathbb{G}\rvert, \lvert\mathbb{G}_T\rvert, \lvert\mathbb{Z}_p\rvert$ stand for the size of an element in $\mathbb{G}, \mathbb{G}_T, \mathbb{Z}_p$. There are n positive attributes, n negative attributes, and n wildcards in our system, and [28], $3n$ attributes in Ref. [26].

Table 4: Comparison in computation.

	KeyGen	Encrypt	Search	Predecrypt	Decrypt
[26]	$(6n+2)G + (9n+1)Z_p$	$(3n+1)G + G_T + 3nZ_p$	$(3n+3)G_T + (3n+1)C_e$	—	—
[28]	$(34n+103)G + (2n+6)Z_p$	$(24n+74)G + 2G_T$	—	—	$(4n+14)(G+G_T)$
Our system	$(7n+31)G + (6n+25)Z_p$	$(3n+14)G + 2G_T + 3Z_p$	$2C_e$	$(4n+17)(G_T+C_e)$	$2G_T + Z_p$

G, G_T stand for the exponentiation operation in $\mathbb{G}$ and $\mathbb{G}_T$. Z_p stands for the operation in $\mathbb{Z}_p$. C_e stands for the billnear pairing operation. There are n positive attributes, n negative attributes, and n wildcards in our system, and [28], $3n$ attributes in Ref. [26].

search specific ciphertexts according to the keyword they provide (ciphertext search). To achieve lightweight computation, predecryption is provided in Ref. [30] and our system. As a result, task performers in both of the two systems can decrypt the ciphertext with their mobile devices efficiently. Moreover, the access structure in Ref. [26, 28] and our system are the same, while the other schemes use "And-Gate" or the "Tree" structure, which explains why we only compare our system with Ref. [26, 28] in Tables 3 and 4.

5.2. Theoretical Analysis. As mentioned above, Table 3 gives the comparison between Ref. [26, 28] and our system. Five objects are selected to be compared. Specifically, the size of the public key in our system is the smallest among all three schemes when . When , Ref. [26] outperforms our system. When, Ref. [28] outperforms Ref. [26]. The size of the master secret key in our system is , which is superior to that in Ref. [26, 28] when . The size of the secret key in Ref. [28] and our system are almost the same. It is less than that in Ref. [26]. For the size of ciphertext , our system is superior to Ref. [28]. We do not list in Ref. [26] for the reason that Ref. [26] only provides an algorithm to generate an encrypted index for a file waiting to be outsourced. The size of the trapdoor in Ref. [26] grows linearly with the number of attributes , while in our system, is constant . To sum up, the storage overhead in our system is significantly reduced compared to Ref. [26, 28].

Table 4 compares the computation overhead of, and. Note that Ref. [26] only generates a secure index in the encryption algorithm, the function of decryption is not provided. The overhead of in our system is similar to that in Ref. [26], and Ref. [28] contains and which is expensive compare to our system. Both the overhead of in our system and Ref. [26] are less than that in Ref. [28]. In Ref. [26], the computational overhead of grows every time the attribute increases, while it is in our system. We allow the crowdsensing platform to perform the Predecrypt algorithm such that the computational overhead on the user side in our system is only. However, in Ref. [28], the overhead is $(4n+4)(G+G_T)$.

5.3. Experimental Analysis. We implement the scheme in Ref. [26, 28] and our system with java pairing-based cryptography 2.0.0 (JPBC) [33] to evaluate their performance. Type A curves predefined in JPBC is used. The experimental environment we used is a MacBook Prolaptop with Intel Core i7 2.7 GHz processor and 16 GB RAM. To provide a fair condition for comparison, the access policies of ciphertexts are defined in the form of , where represent the position of wildcards, positive attributes, and negative attributes, respectively. Each kind of attributes in the access policy increases from 3 to 27, which means the total attributes increase from 9 to 81 in the experiment. To make the

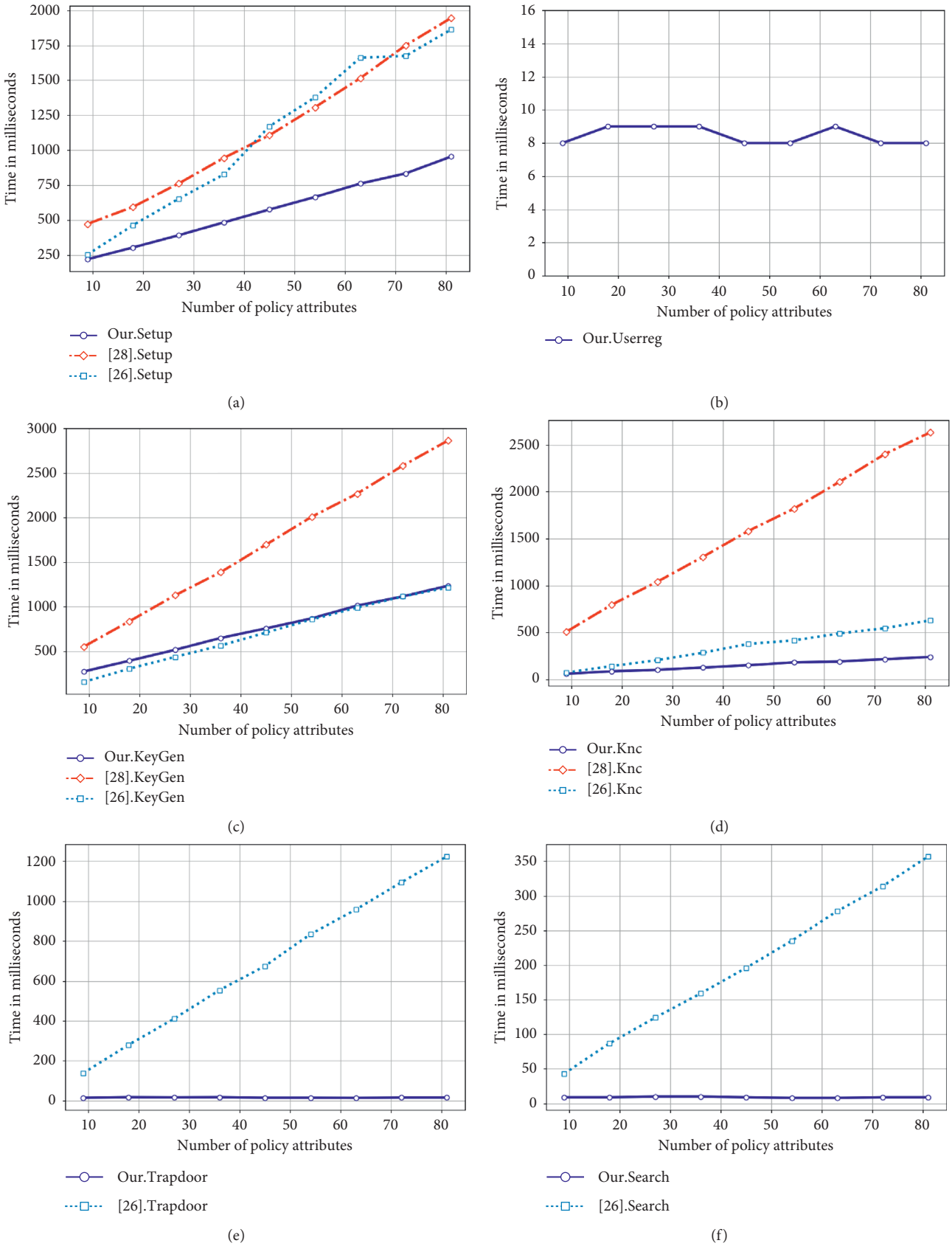
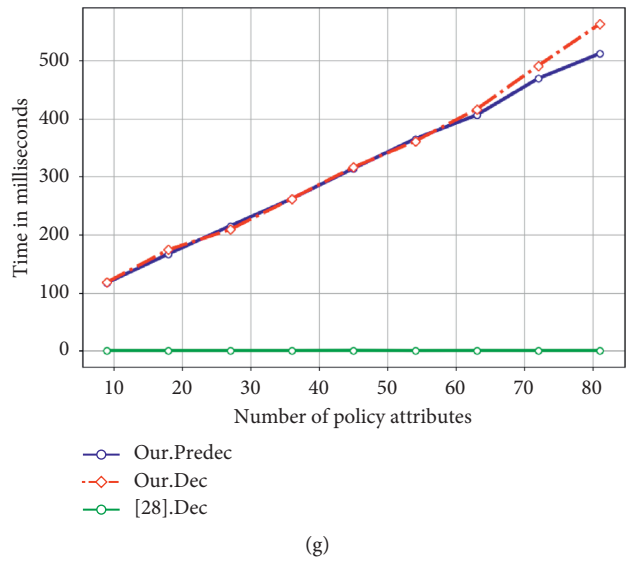


Figure 3: Continued.



(g)

Figure 3: Experimental results on Setup, UserReg, KeyGen, Encrypt, Trapdoor, Search, and Decrypt. (a) Setup Time. (b) UserReg Time. (c) KeyGen Time. (d) Encrypt Time. (e) Trapdoor Time. (f) Search Time. (g) Decrypt Time.

experiment results convincing, we repeat all the algorithms 50 times and take the average.

Figure 3 presents the time cost of algorithms in Ref. [26, 28] and our system. As we can see, the time cost of Setup, KeyGen, and Encrypt, in Figures 3(a), 3(c), and 3(d), of all the three schemes, grow linearly with the number of policy attributes. Although Ref. [26] is 100 ms faster in the beginning, the time cost of KeyGen tends to the same afterward. In Figure 3(b), the time cost of UserReg in FGTAC is 8.5 ms on average, which is independent of the number of attributes. Similarly, the time cost of Trapdoor and Search in Figures 3(e) and 3(f) are both constant (about 16 ms and 9 ms, respectively). However, in Ref. [28], the time cost becomes unacceptable when the number of attributes is large. We give out the time cost of Decrypt in Figure 3(g). The time cost of Predecrypt in our system is the same as the Decrypt in Ref. [28]. However, the time cost for Decrypt on the user side in our system is about 0.56 ms on average, which can be performed efficiently by mobile devices. According to the experimental results, we demonstrate that our system achieves ciphertext search and predecryption without increasing computational overhead comparing to other schemes.

6. Conclusions

We have designed a fine-grained access control system called FGTAC in the scenario of mobile crowdsensing with an awareness of protecting the privacy of requesters. Specifically, the content of the task is encrypted by CP-ABE technology and only can be obtained by task performers who have appropriate attributes. In other words, the requester can decide the group of users who can access his task independently. To prevent the semitrusted crowdsensing platform from deducing the privacy of the requester and task performers, the access policy of the encrypted task is fully hidden in our system. For task performers, an efficient task search function, which can be conducted within approximately 9 ms, is provided. Moreover, considering the devices of task performers may be resource-limited, we also developed the function of ciphertext predecryption. According to the analysis in Section 4.2, the proposed system is CPA secure, CKA secure, secret key exposure attack secure, and collude attack security. The experiment results demonstrate that the computational overhead of our system is efficient. In our future work, we will design the mobile crowdsensing system supporting audit and user tracking without yielding expensive computational overhead such that a feedback mechanism for the task performers can be achieved.

Data Availability

All data included in this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61472343 and 61972094) and the Young Talent Promotion Project of Fujian Science and Technology Association.

References

[1] Z. Xiao, H. Fang, and X. Wang, "Anomalous iot sensor data detection: an efficient approach enabled by nonlinear frequencydomain graph analysis," *IEEE Internet Things Journals*, vol. 23, no. 1–1, 2020.

[2] M. Xiao, G. Gao, J. Wu, S. Zhang, and L. Huang, "Privacy-preserving user recruitment protocol for mobile crowdsensing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 519–532, 2020.

[3] S. Song, Z. Liu, Z. Li, T. Xing, and D. Fang, "Coverage-oriented task assignment for mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7407–7418, 2020.

[4] Z. Zhang, S. He, J. Chen, and J. Zhang, "REAP: an efficient incentive mechanism for reconciling aggregation accuracy and individual privacy in crowdsensing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 12, pp. 2995–3007, 2018.

[5] Y. Zheng, H. Duan, and C. Wang, "Learning the truth privately and confidently: encrypted confidence-aware truth discovery in mobile crowdsensing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2475–2489, 2018.

[6] I. J. Vergara-Laurens, L. G. Jaimes, and M. A. Labrador, "Privacy-preserving mechanisms for crowdsensing: survey and research challenges," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 855–869, 2017.

[7] H. Fang, X. Wang, and L. Hanzo, "Learning-aided physical layer authentication as an intelligent process," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2260–2273, 2019.

[8] H. Fan, K. Xing, L. Tan et al., "Privacy preserved self-awareness on the community via crowd sensing," *Secured Communication Networks*, vol. 2017, Article ID 8026787, 8 pages, 2017.

[9] Y. Sei and A. Ohsuga, "Differential private data collection and analysis based on randomized multiple dummies for untrusted mobile crowdsensing," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 926–939, 2017.

[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pp. 89–98, Alexandria, VA, USA, November 2006.

[11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P 2007)*, pp. 321–334, Oakland, CL, USA, May 2007.

[12] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Transactions on Dependable and Secure Computing*, vol. 32, p. 1, 2020.

[13] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, and K. R. Choo, "Cryptcloud+: secure and expressive data access control for cloud storage," *IEEE Transactions on Services Computing*, vol. 28, p. 1, 2018.

[14] S. Xu, Y. Li, R. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, vol. 34, p. 1, 2019.

[15] L. Wu, Y. Xiong, M. Wu, Y. He, and J. She, "A task assignment method for sweep coverage optimization based on crowdsensing," *IEEE Internet of Things Jounals*, vol. 6, no. 6, 2019.

[16] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1317–1331, 2020.

[17] J. Ning, G. S. Poh, X. Huang, R. Deng, S. Cao, and E.-C. Chang, "Update recovery attacks on encrypted database within two updates using range queries leakage," *IEEE Transactions on Dependable and Secure Computing*, vol. 42, p. 1, 2020.

[18] S. Sedghi, P. van Liesdonk, S. Nikova, P. H. Hartel, and W. Jonker, "Searching keywords with wildcards on encrypted data," in *Proceedings of the Security and Cryptography for Networks, 7th International Conference, SCN 2010*, pp. 138–153, Amalfi, Italy, September 2010.

[19] J. Ning, J. Xu, K. Liang, F. Zhang, and E.-C. Chang, "Passive attacks against searchable encryption," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 789–802, 2019.

[20] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor, "A secure mobile crowdsensing game with deep reinforcement learning," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 35–47, 2018.

[21] F. Khan, A. Ur Rehman, J. Zheng, M. A. Jan, and M. Alam, "Mobile crowdsensing: a survey on privacy-preservation, task management, assignment models, and incentives mechanisms," *Future Generation Computer Systems*, vol. 100, pp. 456–472, 2019.

[22] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks,," in *Proceedings of the 2015 IEEE Conference on Computer Communications, INFOCOM 2015*, pp. 2254–2262, Kowloon, Hong Kong, May 2015.

[23] Z. Wang, R. Tan, J. Hu et al., "Heterogeneous incentive mechanism for time-sensitive and location-dependent crowdsensing networks with random arrivals," *Computer Networks*, vol. 131, pp. 96–109, 2018.

[24] Z. Wang, J. Hu, R. Lv et al., "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2019.

[25] A. Sahai and B. Waters, "Fuzzy identity based encryption," *Cryptology ePrint Archive*, vol. 9216, p. 86, 2004.

[26] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1187–1198, 2016.

[27] L. Liu, J. Lai, R. H. Deng, and Y. Li, "Ciphertext-policy attribute-based encryption with partially hidden access structure and its application to privacy-preserving electronic medical record system in cloud environment," *Security and Communication Networks*, vol. 9, no. 18, pp. 4897–4913, 2016.

[28] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 35–45, 2016.

[29] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable $\sigma $ -time outsourced attribute-based encryption for access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 94–105, 2018.

[30] N. Helil and K. Rahman, "CP-ABE access control scheme for sensitive data set constraint with hidden access policy and constraint policy," *Secured Communication Networks*, vol. 2713, no. 595, 2017.

[31] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13*, pp. 463–474, Berlin, Germany, November 2013.

[32] S. Qiu, J. Liu, Y. Shi, and R. Zhang, "Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack," *Science China Information Sciences*, vol. 60, no. 5, 2017.

[33] A. D. Caro and V. Iovino, "jpbc: java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pp. 850–855, Kerkyra, Greece, July 2011.