Article

Privacy-Preserving Task-Matching and Multiple-Submissions Detection in Crowdsourcing

Jie Xu [1,2,*], Zhaowen Lin [2,3] and Jun Wu [2,3]

[1] The School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2] The National Engineering Laboratory for Mobile Network Security, Beijing 100876, China; linzw@bupt.edu.cn (Z.L.); wujun@bupt.edu.cn (J.W.)
[3] The School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China
* Correspondence: cheer1107@bupt.edu.cn

Abstract: Crowdsourcing enables requesters to publish tasks to a platform and workers are rewarded for performing tasks of interest. It provides an efficient and low-cost way to aggregate data and solve problems that are difficult for computers but simple for humans. However, the privacy risks and challenges are still widespread. In the real world, the task content may be sensitive and only workers who meet specific requirements or possess certain skills are allowed to acquire and perform it. When these distributed workers submit their task answers, their identity or attribute privacy may also be exposed. If workers are allowed to submit anonymously, they may have the chance to repeat their answers so as to get more rewards. To address these issues, we develop a privacy-preserving task-matching and multiple-submissions detection scheme based on inner-product cryptography and proof of knowledge (PoK) protocol in crowdsourcing. In such a construction, multi-authority inner-product encryption is introduced to protect task confidentiality and achieve fine-grained task-matching based on the attributes of workers. The PoK protocol helps to restrict multiple submissions. For one task, a suitable worker could only submit once without revealing his/her identity. Moreover, different tasks for one worker are unlinkable. Furthermore, the implementation analysis shows that the scheme is effective and feasible.

Keywords: task-matching; anonymous multi-submission detection; inner-product encryption; zero-knowledge proof

check for updates

Citation: Xu, J.; Lin, Z.; Wu, J. Privacy-Preserving Task-Matching and Multiple-Submissions Detection in Crowdsourcing. *Sensors* **2021**, *21*, 3036. https://doi.org/10.3390/s21093036

Academic Editor: Wenjuan Li

Received: 30 March 2021
Accepted: 23 April 2021
Published: 26 April 2021

1. Introduction

With the development of network technologies and the popularity of smartphones, crowdsourcing has become a popular distributed paradigm for problem-solving, which is applied to address problems that are too complex for computer programs or of high cost for an organization. An early typical example of crowdsourcing is captcha. ReCAPTCHA [1], a project initiated by Carnegie Mellon University, uses the wisdom of the masses to help the digitization of ancient books in the form of crowdsourcing. This project scans the text, which cannot be recognized by the optical character recognition technology accurately, and displays it in the captcha question, so that a human can recognize it when answering the captcha question.

In the era of big data, the amount of data is increasing, and the forms of data are more diversified, which leads to increasing demand for crowdsourcing and the increasing forms of tasks. Crowdsourcing platforms such as Amazon Mechanical Turk (AMT) [2], crowdflower and upwork of Amazon came into being. On these platforms, tens of millions of workers from more than 100 countries are involved in solving problems. It has inspired the collective imagination of researchers in numerous fields such as human–computer interaction, machine learning, artificial intelligence, information retrieval, database community, etc.

The openness and sharing of crowdsourcing make it more vulnerable to various attacks since it allows attackers to join crowdsourcing systems freely as requesters or workers. When task requesters have tasks to crowdsource, they need to set some parameters, including task pricing, answer time, task worker quality, etc. After that, they can publish tasks on the crowdsourcing platform, and then the tasks will be assigned to workers. When a task is answered by a worker, the requester can choose to accept or reject the answer. If the requester accepts the answer, he/she will pay the worker accordingly. In this process, combined with the task constraints, task content and worker authentication information, the attacker may infer the important private information of the participants, including identity, age, occupation, residence, and so on. If such kind of information cannot be properly kept, it will reduce the enthusiasm of users to participate in the task and further affect the completion of the task.

In the process of task release and matching, since different workers have their own specialties, unsuitable or malicious workers may randomly answer questions to get the reward, or deliberately submit wrong answers to distort the true value. To ensure the quality of answers, the requester should set up task constraints for different tasks so as to match appropriate workers. There are many ways of keyword matching. The flexibility of accurate matching is poor. The matching method that supports multiple policy expressions is more in line with diverse requirements, e.g., ((major = (art $\vee$ artificial intelligence)) $\wedge$ (age $\geq$ 30)), etc. Under the premise of privacy protection, how to achieve flexible task-matching has become a thorny issue.

In most previous mutual privacy-preserving task allocation research, the homomorphism [3] is adopted to realize multiple types of ciphertext policy matching without revealing task constraints and workers' private attributes, which cause the downgrade of efficiency. Moreover, content confidentiality is closely related to the privacy of participants. For the privacy of the task content, the proxy re-encryption algorithm or other technologies is needed. Then the computation and communication cost is further increased. However, based on inner-product encryption the relevant work [4] considered flexible matching of encrypted keywords and fine-grained access control of task content simultaneously. With the expansion of the network scale, it is difficult for a single authentication center to manage workers' keys efficiently. The multi-authority model [5] could better adapt to a large-scale distributed network. However, at this time, there are not only collusion problems of workers but also collusion or damage problems of some attribute authorities in the system.

After the task is assigned, the worker will perform the task and submit task data. At this phase, we should first ensure that it is the right workers who meet the requirements submit the answers. However, similarly, the workers may not wish to be tracked by the server. Since the platform is not completely credible, it may expose the worker's privacy because of interest-driven. Due to the flexible matching requirements set by the task requester, an attribute-based signature could be used. It allows signers to sign a message under policies that satisfying their attributes. In a crowdsourcing system, the worker obtains the attribute-based private key from the authority. When his/her attributes satisfy the constraint policy set by the requester, the signature can be verified to be valid. With anonymous attribute-based signature authentication, it is possible to prevent inappropriate workers from submitting, while avoiding the leakage of worker's privacy. However, dishonest qualified workers may submit multiple answers to a task for more rewards. Moreover, if a greedy participant submits similar or identical results with different pseudonyms many times, it will also reduce the diversity and credibility of the data, and further produce bias to the results that should have been perceived by numerous participants. Actually, a privacy-preserving submission detection scheme is needed, which ensures that only qualified workers can participate in answering and cannot submit repeatedly, and the worker's identity and history of participating in the task will not be disclosed.

In this paper, we first analyze the potential security threats to the privacy and quality assurance issues of crowdsourcing during the task allocation and task submission phase, and then propose a security and privacy protection model of the system. After that, a scheme based on multi-authority inner-product encryption (MIPE) and zero-knowledge proof protocol, called zk-MIPE, is designed. With MIPE, the scheme can realize secure sharing of task content and the flexible assignment of tasks based on encrypted task constraints and workers' attributes. With the repeated submission detection algorithm constructed by zero-knowledge proof protocol, it ensures that the requester and platform can only verify that a worker who has submitted an answer about a task meets the corresponding task constraints but cannot judge his/her specific identity or attribute information. Also, if the worker performs multiple tasks, no one can link them. At the same time, when workers submit repeatedly for the same task, they can be identified by association. Under the premise of protecting the participants' privacy, the scheme selects suitable workers to submit an answer honestly with more professional skills, thus further improve the quality of aggregated task data. In summary, the technical innovation of the proposed system is: we designed a novel MIPE scheme and a one-time anonymous inner-product authentication protocol based on zero-knowledge proof, and proved the confidentiality, one-time authentication, anonymity and unlinkability of the solution. In terms of application, we achieved the innovative features in function and security for crowdsourcing privacy protection: (1). it supports flexible task-matching based on inner-product with mutual privacy; (2). it supports anonymously inner-product-based authentication and duplicate submission detection without revealing identity and attributes privacy.

2. Related Work

Crowdsourcing Privacy

Presently, for a variety of data processing and analysis tasks, only relying on machine algorithms cannot achieve desired results. Fortunately, crowdsourcing provides an efficient and low-cost paradigm to solve this problem with the advantage of distributed mode. However, security and privacy issues are still thorny. In past research on privacy-preserving, some researchers analyzed the privacy threats of the whole crowdsourcing process to propose an overall security framework [6]. Meanwhile, blockchain is applied to deal with potential security issues (e.g., single point of failure, sensitive leakage) without a trusted third party, such as SecBCS [7], MCS-chain [8], CrowdBC [9]. Also, novel fog-based computing framework is proposed [10] for low latency vehicular crowdsensing networks.

Still, there are researchers in-depth discussing crowdsourcing security threats at each phase, and designing differentiated privacy protection schemes for specific security objectives using diversified technologies. Among them, location privacy is the first concern of researchers. The methods used to solve location privacy include k-anonymity [11], differential privacy [12,13], game theory [14], commitment [15], machine-learning-based obfuscation [16,17], encryption [18,19], etc. However, most of them focus on protecting the workers' privacy. To provide mutual privacy for both requesters and workers, Liu [3] proposed a privacy-preserving protocol based on homomorphic encryption with a dual-server setting. After that, Shu [20] constructs a task-matching scheme over the encrypted location with a single server by applying searchable encryption. Actually, in the scenes they mainly concern, the privacy requirements of task content are not high, which are usually public to all participants. However, the need for content privacy protection still exists. For some sensitive task content involving address, occupation and purpose, it can help attackers to further infer participants' privacy by combining other information. In the privacy-aware task assignment schemes proposed by Liu et al. [21] and Yuan et al. [22], attribute-based encryption is applied to protect content privacy and realize fine-grained access control. Extending to more complex multi-keyword crowdsourcing allocation scenario, our prior work [4] introduced inner-product encryption (IPE) to support flexible matching policies without disclosing task privacy and worker privacy. However, as the worker scale increases, centralized single authority mode has obvious disadvantages in efficiency and security.

Moreover, most of these schemes mainly discussed privacy protection in the task allocation phase. While in the data submission phase, the platform should verify the identity or attribute information of the participants to evaluate whether the appropriate workers have performed the task. At this time, if we do not provide effective privacy protection, the secure closed-loop still cannot be constructed. Based on signature and other technologies, Ni [23] and Shu [24] presented Sybil detection schemes respectively. Nevertheless, they are concerned about the deduplication of encrypted data content rather than the identity privacy of workers. Though Lu [25] proposed a blockchain-based private and anonymous repetition detection scheme for task submission, the introduction of zk-SNARK increases the computational overhead of the scheme. Compared with the previous scheme, we focus on the privacy protection of task releasing and task submission. In the task releasing stage, the scheme requires privacy of task content and constraint conditions, and should realize flexible ciphertext task-matching. In the task submission phase, workers could submit perceptual data anonymously and cannot submit it repeatedly.

Inner-Product Cryptosystems

In 1984, Shamir [26] proposed the concept of ID-based public key cryptography and constructed the first ID-based digital signature scheme based on the large integer decomposition problem. However, it was not until 2001 that Boneh and Franklin [27] presented the first secure and practical ID-based encryption scheme based on elliptic curve bilinear pairings. After that, Sahai and Waters [28] designed a fuzzy identity-based encryption scheme based on key sharing theory in 2005, and further proposed the concept of attribute-based encryption (ABE). Since then, research on ABE has covered privacy protection, richer access policy types, efficiency, security assumptions, attribute revocation, and other directions [29–31]. To implement policy hiding, Boneh and Waters [32] introduced a hidden vector encryption scheme supporting conjunctive, subset and range queries in 2007. Then Katz [33] raised the concept of IPE for the first time and proved its security under the standard model in 2008. The scheme allows conjunctive disjunction, polynomial and inner-product queries. However, the length of ciphertext increases linearly with the increase of vector dimension. Afterwards, Attrapadung and Libert [34] developed a scheme to reduce the length of ciphertext to a constant. Furthermore, Okamoto [35] realized a scheme with constant key length. On the other hand, to reduce the management cost of a single authentication server, Chase [5] presented an encryption scheme that enables the implementation of the AND access policy in a multi-authority environment. On this basis, to reduce the complexity of user decryption, Li [36] constructed a multi-authority outsourcing attribute encryption system based on linear secret-sharing schemes (LSSS). However, the IPE scheme in multi-authority environment still needs to be proposed. For anonymous authentication, Yuen [37] adopted k times attribute signature (k-ABS) to restrict access times. The data is still stored remotely in plaintext. Ning presented an outsourced σ-time attribute-based encryption (σ-ABE) scheme [38], in which users apply attributes as identity without using real names. Although the server cannot know a user's identity, it can associate a user's previous and subsequent access through the proxy key. Moreover, due to the lack of association between the attribute-related private key and the validation tags for times, there is a risk that the attacker will steal the other's validation tag, and then send his own attribute-related private key to access the data illegally. Inner-product cryptosystems enables the realization of flexible and diverse policies. Compared with cryptosystems supporting LSSS policy, it allows policy hiding. However, presently, neither the IPE encryption for multi-authority nor the k-time inner-product-based authentication scheme has been proposed. Therefore, in this article, we intend to solve this problem and apply the design scheme to crowdsourcing privacy protection.

3. Preliminary

3.1. System Assumption

As shown in Figure 1, the proposed crowdsourcing system contains the following entities: central authority CA, multiple attribute authorities AA, the crowdsourcing server CS, requesters and workers. As a trusted third party, CA initializes the system, generates global parameters and supervises each AA. Suppose there are m attribute authorities, denoted as AA_1,...,AA_m. They are responsible for managing disjoint attribute sets. The requester is an enterprise or individual who publishes the task on the system platform. The worker is a user who performs tasks and submits perception data. CS verifies whether workers meet the requirements and submit repeatedly. Let the sets of vectors $\vec{w} = (w_1, ..., w_m) \in Z_q^{mn}$ and $\vec{z} = (z_1, ..., z_m) \in Z_q^{mn}$ be the task constraint and the worker's attribute-based vector. Only if $< z_j, w_j >= 0$ holds for all $j \in [1, m]$, the worker could decrypt the corresponding task ciphertext.

For system security, we need at least one attribute authority is honest and secure in such a system. The requester is also considered to be honest. CS is considered to be honest-but-curious, i.e., it will honestly execute the protocol and screen out suitable workers, but it will also be curious about more information, such as task content and participant identity. The worker is considered to be honest but greedy, i.e., he will execute the protocol honestly but may submit data multiple times to get more rewards.

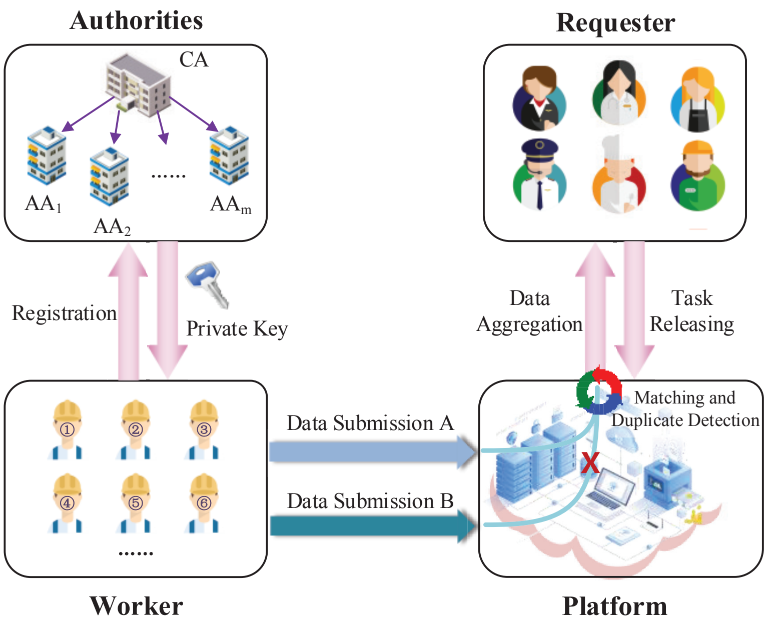


Figure 1. Framework of the zk-MIPE system.

The specific security objectives of the zk-MIPE scheme are as follows.

(1) *Content and constraints privacy.* Task content and constraints should be released in the form of ciphertext. Only suitable workers could learn the corresponding task plaintext.

(2) *One-time attribute-based authentication.* If the worker's attributes meet the task constraints, he/she can provide a valid proof to the CS. If not, he/she cannot forge a valid proof.

(3) *Identity and attribute privacy.* Although the CS enables the filtering out of suitable workers and the restriction of multiple submission, it cannot know the worker's identity and attributes, or even associate the previous and subsequent tasks that a worker participates in.

3.2. Inner-Product Access Structure

The inner product is a generalization of the concept of point multiplication. In a vector space, it is a method of multiplying vectors, and the product is a scalar. For a real vector space, let x_1, x_2, x_3 be vectors and r be a scalar, then the inner product $< \cdot, \cdot >$ satisfies the following properties.

(1) $< x_1 + x_2, x_3 > = < x_1, x_3 > + < x_2, x_3 >$;
(2) $< rx_1, x_2 > = r < x_1, x_2 > = < x_1, rx_2 >$;
(3) $< x_1, x_2 > = < x_2, x_1 >$;
(4) $< x_1, x_1 > \geq 0$, and only when $x_1 = 0$ the equal sign holds.

3.3. Bilinear Group

Definition 1. *Bilinear Map [27]: A group generator $\mathcal{G}$ takes a security parameter λ as input. It outputs a group $\vec{G} = (G_1, G_T, e, q)$ of prime order q, where G_1 is an additional group and G_T is a multiplication group. Let g be a generator of G_1. The bilinear map e has the following properties.*

(1) Bilinearity: For random $a, b \in Z_q$ and $x, y \in G_1$, we have $e(x^a, y^b) = e(x, y)^{ab}$;

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

(3) Computability: For random $g, h \in G_1$, there exists an efficient algorithm to compute $e(g, h)$.

Definition 2. *Computational Diffie-Hellman (CDH) Problem: A challenger runs $\mathcal{G}(\lambda)$ to generate $\vec{G} = (G_1, G_T, e, q)$. Then it chooses a random generator g and random $a, b \in Z_q$. Given a tuple (g, g^a, g^b) as input, we say that the CDH assumption holds if there is no polynomial-time algorithm can compute the element g^{ab}.*

Definition 3. *Decisional Diffie-Hellman (DDH) Problem: A challenger runs $\mathcal{G}(\lambda)$ to generate $\vec{G} = (G_1, G_T, e, q)$. Then it chooses a random generator g and random $a, b \in Z_q$. Given a tuple (g, g^a, g^b) as input, we say that the DDH assumption holds if there is no polynomial-time algorithm can distinguish g^{ab} from a random value with nonnegligible advantage in G_1.*

Definition 4. *q-Decisional Diffie-Hellman Inversion (DDHI) Problem: A challenger runs $\mathcal{G}(\lambda)$ to generate $\vec{G} = (G_1, G_T, e, q)$. Then it chooses a random generator g and a random $x \in Z_q$. Given a tuple $(g, g^x, g^{x^2}, ..., g^{x^q})$ as input, we say that the q-DDHI assumption holds if there is no polynomial-time algorithm can distinguish $g^{1/x}$ from a random value with nonnegligible advantage in G_1.*

3.4. Zero-Knowledge Proof Protocol

The zero-knowledge proof (ZKP) protocols have been applied to numerous fields, including both traditional secure multiparty computation and emerging privacy protection projects in distributed ledger and blockchain, such as Zcash [39], hawk [40], and so on.

A ZKP system is a protocol between a computationally bounded prover and a verifier. Let R be an NP relation. Set $R(x) = \{w : (x, w) \in R\}$ and the language $L = \{x : \exists w, \text{st}\,(x, w) \in R\}$. During the protocol, the verifier is convinced by the prover that x belongs to L, i.e., there exists a witness w such that $(x, w) \in R$ for x. However, in proof of knowledge (PoK), the prover cannot only prove the exists of some witness but also be convinced that he/she indeed know a specific witness w.

The main properties of ZKP for a relation R are as follows.

Completeness: Given a witness ω that satisfies $(x, \omega) \in R$, the prover could convince the verifier of his knowledge. i.e.,

$$\text{Verify(Prove}(x, \omega))\text{=accept.}$$

Soundness: Given a witness ω that does not satisfy $(x, \omega) \in R$, for any polynomial-time prover, the probability that the verification can be accepted is negligible. i.e.,

$$\text{Pr[Verify(Prove}(x, \omega))\text{=accept}\wedge (x, \omega) \notin R] \leq neg(\lambda).$$

Zero knowledge: The interaction between a prover and a verifier is called a view. The zero-knowledge property could be captured by the existence of a simulator E that could access to the verifier's input but not the prover's: with the assumption $x \in L$, if the simulated view, i.e., the transcript, is indistinguishable from the original view between the honest prover and the verifier, whether honest or cheating. We say the ZKP scheme has the property of zero knowledge. Moreover, in PoK, there exists a knowledge extractor, which has rewindable access to the prover, and could extract the witness with nonnegligible probability.

4. Model of zk-MIPE

Definition 5. *A privacy-preserving task-matching and multi-submission detection scheme zk-MIPE is defined by a tuple of the following algorithms:*

- **CA Setup**(λ, m). The algorithm is executed by the central authority CA. It takes a security parameter λ and several attribute authorities m as inputs. It then publishes a system public key PK and keeps a system master key SK secretly.
- **AA Setup**(λ, n). Run by the attribute authorities AA_j, the algorithm takes a security parameter λ and several intra-domain attributes n as inputs. It then outputs a public key PK_j and an attribute-related secret key SK_j for each AA_j.
- **Task Releasing**$(M, PK, \{PK_j\}_{j\in[1,m]}, \vec{w})$. Executed by the requester, the algorithm takes the public key, a message and a constraint as inputs. Then it outputs an inner-product ciphertext C.
- **Registration**$(SK, u, \{SK_j\}_{j\in[1,m]}, \vec{z})$. According to the identity u and attributes $\vec{z}$, the secret key K_u for the registrant is generated by CA and AA_j.
- **Decryption**(C, K_u). Executed by the worker u, the algorithm takes the ciphertext C and the private key K_u as inputs. It then outputs the message M.
- **Matching and Multi-Submission Verification**$(C, \{PK_j\}_{j\in[1,m]}, K_u)$. Executed by CS and workers, this algorithm takes as inputs the public parameter PK_j, the private key sk_j and the ciphertext C. It then runs a zero-knowledge proof to verify the compliance of attributes and submission times between CS and the worker. It then outputs accept or reject.

5. zk-MIPE Scheme

Based on the difficult problems of bilinear pairings and a specific zero-knowledge proof protocol, we propose a zk-MIPE algorithm to deliver task-matching and multiple data submissions detection services in crowdsourcing. The scheme is roughly described in Figure 2.

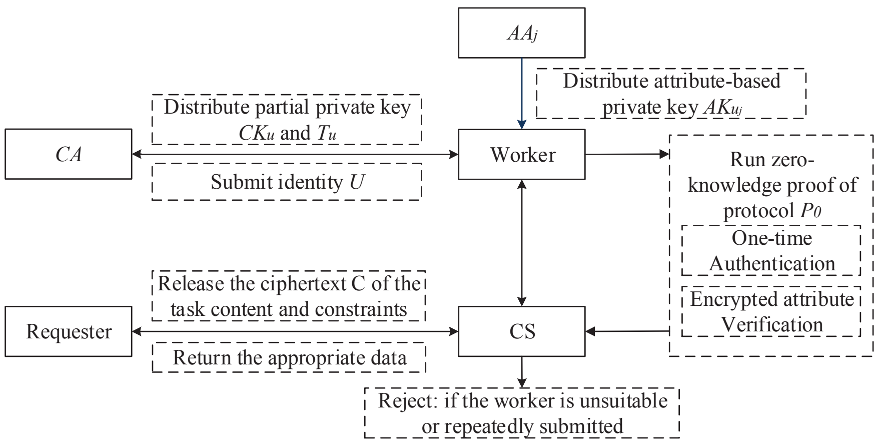

AAj
Distribute attribute-based private key AKuj
Distribute partial private key CKu and Tu
CA
Worker
Submit identity U
Run zero-knowledge proof of protocol P0
One-time Authentication
Encrypted attribute Verification
Release the ciphertext C of the task content and constraints
Requester
CS
Return the appropriate data
Reject: if the worker is unsuitable or repeatedly submitted

Figure 2. Overview of the zk-MIPE Scheme.

For instance, suppose the task content is to collect some physical indicators, and the task constraint is: male, 48 years old, and suffering from hypertension or arthritis. Let Z_1, Z_2, Z_3 be three attributes, which represent gender, age and disease. Let z_1, z_2, z_3 be the specific attribute values for workers. We quantify gender and disease in task constraints, e.g., for attribute Z_1, set male = 1 and female = 2, and for attribute Z_3, set hypertension = 1, arthritis = 2, gastritis = 3. Then the restriction is $\{Z_1 = 1 \wedge Z_2 = 45 \wedge (Z_3 = 1 \vee Z_3 = 2)\}$ which could be further denoted as $r_1(Z_1 - 1) + r_2(Z_2 - 45) + r_3(Z_3 - 1)(Z_3 - 2) = 0$ for $r_1, r_2, r_3 \in F_q$. The worker's attribute vector $\vec{z}$ is defined as $(1, z_1, z_2, z_3, z_3^2)$. To make the equation $r_1(z_1 - 1) + r_2(z_2 - 45) + r_3(z_3 - 1)(z_3 - 2) = 0$ hold if and only if the inner product $< \vec{w}, \vec{z} >= 0$ is zero, the vector $\vec{w}$ is defined as $(-45r_1 - r_2 + 2r_3, r_1, r_2, -3r_3, r_3)$.

Given a task ciphertext encrypted with restriction $\vec{w}$, if a worker's attribute is: male, 45 years old, with hypertension, he will be able to decrypt the task ciphertext and be eligible to participate in the task. In the task submission stage, he could generate a proof in the form of zero-knowledge and sends it to the CS together with his collected data. In the process of verification, the CS can verify whether the worker meets the constraints and whether the submission is repeated, but cannot get the explicit attribute information of the worker. Each worker could select a random number ϕ as his identity-based private key. For each task, he sends the calculated $S = e(g,g)^{\frac{1}{\phi + H(C_{task})}}$, where H is a one-way hash function, and the proof of the attribute private key to the CS. Through a zero-knowledge proof protocol he will prove to the CS that it is the first time to submit, and he is a suitable worker without disclosing $\phi, \vec{z}$, and the private key of $\vec{z}$. The value of S is unique for one task. If the CS detects the same S, it means duplicate submission. Moreover, if a worker chooses another random number ϕ' as his identity-based private key, since the attribute private key, generated by the authorities, is bound with the information of ϕ, he will not be able to pass the verification of matching attributes and constraints.

Furthermore, a crowdsourcing task usually involves multiple workers. IPE just solves the problem of one to many. A ciphertext can be decrypted by many users, which is suitable for multi-user scenarios. Once the crowdsourcing requester encrypts a task, it can be decrypted by any worker who meets the requirements. For the crowdsourcing server with mighty computing power, it is also feasible to handle the task requests issued by multiple requesters in parallel. The introduction of multiple authorities further increases the scalability of the scheme.

Specifically, the scheme is as follows.

- **CA Setup** (λ, m). Executed by CA, the algorithm takes a security parameter λ as input and runs $\mathcal{G}(\lambda)$ to output a symmetric group $\vec{G} = (G_1, G_T)$ of prime order q. It picks a random generator $g \in G_1$, a random $t \in Z_q$ and a one-way hash function $H_1 : \{0,1\}^* \to G_1$. Then it sets the public key as $PK = \{g, Y = g^t, H_1\}$ and the system master key as $SK = \{t\}$.
- **AA Setup** (λ, n). The attribute authority AA_j randomly picks $\alpha_j, \gamma_{j1}, ..., \gamma_{jn} \in Z_q$ and computes $h_{ji} = g_1^{\gamma_{ji}}$ as the public key for each attribute Att_{ji} belonging to AA_j. Then AA_j publishes $PK_j = \{e(g_1, g_1)^{\alpha_j}, h_{j1}, ...h_{jn}\}$ and sets $SK_j = \{\alpha_j, \gamma_{j1}, ...\gamma_{jn}\}$ as its secret key.
- **Task Releasing** $(M \in G_T, PK, \{PK_j\}_{j\in[1,m]}, \vec{w} = (w_1, ..., w_m) \in Z_q^{mn})$. The algorithm, executed by the requester, takes the public key PK, PK_j (for $j \in [1, m]$), description of constraints $\vec{w} = (w_1, ..., w_m) \in Z_q^{nm}$ in which $w_j = (w_{ji}) \in Z_q^n$ and the message $M \in G_T$ as input. It randomly chooses $s_1, s_2, \sigma_1, \sigma_2 \in Z_p$ and computes

$$C_0 = M \cdot e(g,g)^{\sum_{j=1}^{m} s_1 \alpha_j}, C_1 = g^{s_1}, C_2 = Y^{s_1}, C_1^{'} = g^{s_2}, C_2^{'} = Y^{s_2},$$
$$\{C_{ji} = h_{ji}^{s_1} \cdot g^{\sigma_1 w_{ji}}, C_{ji}^{'} = h_{ji}^{s_2} \cdot g^{\sigma_2 w_{ji}}\}_{i\in[1,n], j\in[1,m]}, C_T = e(g,g)^{\sum_{j=1}^{m} s_2 \alpha_j}.$$

Then it outputs the task ciphertext as

$$C = (C_0, C_1, C_2, C_1^{'}, C_2^{'}, C_{11}, C_{11}^{'}, ..., C_{mn}, C_{mn}^{'}, C_T).$$

- **Registration** $(SK, u, \{SK_j, z_j = (z_{j1}, ..., z_{j_n}) \in Z_q^n\}_{j\in[1,m]})$. Users can either register as requesters or workers. Both CA and AA_j are responsible for generating private keys for registered users by calling the following algorithms.

(1) If a user registers as a worker, he/she first selected a random $\phi \in Z_q$, computes $U = g^{\phi}$ as the public key, and sends U to CA. Then CA randomly picks $\beta_{u_1}, ..., \beta_{u_m} \in Z_q$, sets $\beta_u = \sum_{j=1}^{m} \beta_{u_j}$ and distributes β_{u_j} to AA_j secretly. In particular, β_u corresponds uniquely with the worker u. Then CA computes

$$CK_u = g^{\frac{\beta_u}{t}}, T_u = (Ug)^{\frac{t}{\beta_u+t}}.$$

After that, CA sends (CK_u, T_u) to the worker. For each registered requester, CA sends the system public key to the requester.

(2) After receiving β_{u_j} from CA, AA_j chooses a random $\tau_{u_j} \in Z_q$ and computes $Q_{u_j} = g^{\tau_{u_j}}$ for the worker u. Then it creates the secret key about the attribute-based vector z_j as

$$AK_{u_j} = g^{\alpha_j - \beta_{u_j}} \cdot Q_{u_j}^{\sum_{i=1}^{n} \gamma_{ji} z_{ji}}.$$

The algorithm outputs the worker secret key as $K_u = (CK_u, T_u, \{AK_{u_j}, Q_{u_j}\}_{j\in[1,m]})$.

- **Decryption** (C, K_u). The algorithm, executed by the worker, takes the ciphertext C and the secret key K_u as input. It first computes

$$E_1 = e(CK_u, C_2), E_2 = \prod_{j=1}^{m} \frac{e(AK_{u_j}, C_1)}{\prod_{i=1}^{n} e(Q_{u_j}^{z_{ji}}, C_{ji})}.$$

Then it could recover message M by computing $M = \frac{C_0}{E_1 \cdot E_2}$.
When $<\vec{w}, \vec{z}>= 0$, the computation is correct since

$$E_1 = e(CK_u, C_2) = e(g^{\frac{\beta_u}{t}}, g^{ts_1}) = e(g, g)^{\beta_u s_1},$$

$$\begin{aligned} E_2 &= \prod_{j=1}^{m} \frac{e(AK_{u_j}, C_1)}{\prod_{i=1}^{n} e(Q_{u_j}^{z_{ji}}, C_{ji})} = \prod_{j=1}^{m} \frac{e(g^{\alpha_j - \beta_{u_j}} \cdot Q_{u_j}^{\sum_{i=1}^{n} \gamma_{ji} z_{ji}}, g^{s_1})}{\prod_{i=1}^{n} e(Q_{u_j}^{z_{ji}}, h_{ji}^{s_1} g^{\sigma_1 w_{ji}})} \\ &= \prod_{j=1}^{m} \frac{e(g^{\alpha_j - \beta_{u_j}}, g^{s_1}) \cdot e(Q_{u_j}^{\sum_{i=1}^{n} \gamma_{ji} z_{ji}}, g^{s_1})}{\prod_{i=1}^{n} e(Q_{u_j}^{z_{ji}}, h_{ji}^{s_1}) e(Q_{u_j}^{z_{ji}}, g^{\sigma_1 w_{ji}})} \\ &= e(g, g)^{\sum_{j=1}^{m} (\alpha_j - \beta_{u_j}) s_1} \cdot \prod_{j=1}^{m} \frac{e(Q_{u_j}^{\sum_{i=1}^{n} \gamma_{ji} z_{ji}}, g^{s_1})}{e(Q_{u_j}, g)^{s_1 \sum_{i=1}^{n} \gamma_{ji} z_{ji}} e(Q_{u_j}, g)^{\sigma_1 <z_j, w_j>}} \\ &= e(g, g)^{\sum_{j=1}^{m} \alpha_j s_1} \cdot e(g, g)^{-\beta_u s_1}. \end{aligned}$$

Thus, $\frac{C_0}{E_1 \cdot E_2} = \frac{C_0}{e(g,g)^{\sum_{j=1}^{m} s_1 \alpha_j}} = M$.

- **Matching and Multi-Submission Verification**$(C, \{PK_j\}_{j\in[1,m]}, K_u)$. The algorithm tasks the system public, the worker secret key K_u and the task ciphertext C as inputs. In the interaction protocol between the worker and the platform, if $< w_j, z_j >= 0$ for $j = [1, m]$, the worker u first computes $S = e(g, g)^{\frac{1}{\phi + H(C_0)}}$ and sends S to CS. Then CS checks whether S has been used once. If used, CS rejects the request. If not, CS will allow u to run the following zero-knowledge proof of knowledge protocol P_0 with it to prove the knowledge of (ϕ, K_u):

$$
\begin{aligned}
&P_0\{(\phi, K_u = (CK_u, T_u, \{AK_{u_j}, Q_{u_j}\}_{j\in[1,m]})) :\\
&S = e(g,g)^{\frac{1}{\phi+H(C_0)}} \wedge\\
&e(T_u, g\cdot CK_u) = e(Ug, g)\wedge\\
&e(CK_u, C_2^{'})\cdot \prod_{j=1}^{m} \frac{e(AK_{u_j}, C_1^{'})}{\prod_{i=1}^{n} e(Q_{u_j}^{z_{ji}}, C_{ji}^{'})} = C_T\}.
\end{aligned}
$$

To implement the protocol P_0, u will calculate some auxiliary inputs and use some tricks to convert the protocol equivalently. Specifically, u interacts with CS as follows. (1) CS randomly picks two generators $\hat{g}, \hat{h} \in G$ and sends them to u, where the discrete logarithm of $\hat{h}$ with respect to $\hat{g}$ is unknown to u. Then u picks random $\kappa, \delta, \mu, \nu_1, ..., \nu_m, \varsigma_{11}, ..., \varsigma_{mn} \in Z_q$ and computes

$$
\begin{aligned}
&\pi_T = T_u\hat{h}^{\kappa}, \chi_T = \hat{h}^{\delta}\hat{g}^{\kappa}, \pi_C = CK_u\hat{h}^{\mu}, \{\pi_{A_j} = AK_{u_j}\hat{h}^{\nu_j}\}_{j\in[1,m]},\\
&\{\pi_{Q_{ji}} = Q_{u_j}^{z_{ji}}\hat{h}^{\varsigma_{ji}}\}_{j\in[1,m], i\in[1,n]}, \rho_1 = \kappa\mu, \rho_2 = \delta\mu.
\end{aligned}
$$

After that, u returns auxiliary values $(\pi_T, \chi_T, \pi_C, \{\pi_{A_j}, \pi_{Q_{ji}}\}_{j\in[1,m], i\in[1,n]})$ to CS. In this case, the protocol can be expressed as the following zero-knowledge proof of knowledge protocol P_1 to prove the knowledge of $(\phi, \kappa, \delta, \mu, \nu_1, ..., \nu_m, \varsigma_{11}, ..., \varsigma_{mn}, \rho_1, \rho_2)$:

$$
\begin{aligned}
&P_1\{(\phi, \kappa, \delta, \mu, \nu_1, ..., \nu_m, \varsigma_{11}, ..., \varsigma_{mn}, \rho_1, \rho_2) :\\
&\chi_T^{-\mu}\cdot \hat{h}^{\rho_2}\cdot \hat{g}^{\rho_1} = 1_G\wedge\\
&S = e(g,g)^{\frac{1}{\phi+H(C_0)}} \wedge\\
&e(\pi_T, g\pi_C) = e(\pi_T, \hat{h}^{\mu})\cdot e(\hat{h}^{\kappa}, g\pi_C)\cdot e(\hat{h},\hat{h})^{-\rho_1}\cdot e(Ug, g)\wedge\\
&e(\pi_C, C_2^{'})\cdot \prod_{j=1}^{m} \frac{e(\pi_{A_j}, C_1^{'})}{\prod_{i=1}^{n} e(\pi_{Q_{ji}}, C_{ji}^{'})} = C_T\cdot e(\hat{h}, C_2^{'})^{\mu}\cdot \prod_{j=1}^{m} \frac{e(\hat{h}, C_1^{'})^{\nu_j}}{\prod_{i=1}^{n} e(\hat{h}, C_{ji}^{'})^{\varsigma_{ji}}}\}.
\end{aligned}
$$

Assume that the auxiliary value calculated by u has been sent to CS. Next, we will describe the implementation details of the honest-verifier zero-knowledge protocol P_1 below.
(1) **Commitment**. u picks random $\xi_{\phi}, \xi_{\kappa}, \xi_{\delta}, \xi_{\mu}, \xi_{\nu_1}, ..., \xi_{\nu_m}, \xi_{\varsigma_{11}}, ..., \xi_{\varsigma_{mn}}, \xi_{\rho_1}, \xi_{\rho_2} \in Z_q$ and computes

$$
\begin{aligned}
&L_1 = \hat{h}^{\xi_{\delta}}\hat{g}^{\xi_{\kappa}}, L_2 = \chi_T^{-\xi_{\mu}}\cdot \hat{h}^{\xi_{\rho_2}}\hat{g}^{\xi_{\rho_1}}, L_3 = S^{\xi_{\phi}},\\
&L_4 = e(\pi_T, \hat{h})^{\xi_{\mu}}\cdot e(\hat{h}, g\pi_C)^{\xi_{\kappa}}\cdot e(\hat{h},\hat{h})^{-\xi_{\rho_1}}\cdot e(g,g)\cdot e(g,g)^{\xi_{\phi}},\\
&L_5 = C_T\cdot e(\hat{h}, C_2^{'})^{\xi_{\mu}}\cdot \prod_{j=1}^{m} \frac{e(\hat{h}, C_1^{'})^{\xi_{\nu_j}}}{\prod_{i=1}^{n} e(\hat{h}, C_{ji}^{'})^{\xi_{\varsigma_{ji}}}}.
\end{aligned}
$$

Then the worker sends these auxiliary values $L_1, ..., L_5$ to CS.
(2) **Challenge**. CS picks a random $\varepsilon \in Z_q$ and sends ε to the worker.
(3) **Response**. the worker computes the following auxiliary value at first.

$$\begin{aligned}
&z_\phi = \xi_\phi - \varepsilon\phi, z_\kappa = \xi_\kappa - \varepsilon\kappa,\\
&z_\delta = \xi_\delta - \varepsilon\delta, z_\mu = \xi_\mu - \varepsilon\mu,\\
&z_{v_1} = \xi_{\nu_1} - \varepsilon\nu_1, ..., z_{v_m} = \xi_{\nu_m} - \varepsilon\nu_m,\\
&z_{\varsigma_{11}} = \xi_{\varsigma_{11}} - \varepsilon\varsigma_{11}, ..., z_{\varsigma_{mn}} = \xi_{\varsigma_{mn}} - \varepsilon\varsigma_{mn},\\
&z_{\rho_1} = \xi_{\rho_1} - \varepsilon\rho_1, z_{\rho_2} = \xi_{\rho_2} - \varepsilon\rho_2.
\end{aligned}$$

Then u sends the sets of $(z_\phi, z_\kappa, z_\delta, z_\mu, z_{v_1}, ..., z_{v_m}, z_{\varsigma_{11}}, ..., z_{\varsigma_{mn}}, z_{\rho_1}, z_{\rho_2})$ to CS.
(4) **Verification**. CS checks whether the following equation holds:

$$\begin{aligned}
&L_1 = \hat{h}^{z_\delta}\chi_T^{\varepsilon}\hat{g}^{z_\kappa}, L_2 = \chi_T^{-z_\mu}\hat{h}^{z_{\rho_2}}\hat{g}^{z_{\rho_1}},\\
&L_3 = S^{z_\phi - \varepsilon H(C_0)} \cdot e(g,g)^{\varepsilon},\\
&L_4 = e(\pi_T, \hat{h})^{\xi_\mu} \cdot e(\hat{h}, g\pi_C)^{\xi_\kappa} \cdot e(\hat{h},\hat{h})^{-\xi_{\rho_1}} \cdot e(g,g) \cdot e(g,g)^{\xi_\phi},\\
&L_5 = C_T{}^{1-\varepsilon} \cdot [e(\pi_C, C_2') \cdot \textstyle\prod_{j=1}^{m} \dfrac{e(\hat{h}, C_1')}{\prod_{i=1}^{n} e(\pi_{Q_{ji}}, C_{ji}')}]^{\varepsilon} \cdot e(\hat{h}, C_2')^{z_\mu} \cdot \textstyle\prod_{j=1}^{m} \dfrac{e(\hat{h}, C_1')^{z_{v_j}}}{\prod_{i=1}^{n} e(\hat{h}, C_{ji}')^{z_{\varsigma_{ji}}}}.
\end{aligned}$$

Through the above interactive process, CS verifies whether the workers meet the constraints and submit repeatedly. If the verification is valid, CS returns the task answer submitted by the worker to the requester. As follows, we discuss the soundness of the protocol.
Soundness of P_0: P_1 is a 3-move protocol, where the prover sends the commitment, the verifier chooses a random challenge, and the prover response to the challenge based on elliptic curve discrete logs. It is straightforward to show that P_1 is of soundness, i.e., there exists an extractor E_1, which is given rewindable black-box access to the prover, could output some witness $(\phi, \kappa, \delta, \mu, \nu_1, ..., \nu_m, \varsigma_{11}, ..., \varsigma_{mn}, \rho_1, \rho_2)$ or a halting symbol $\perp$ to indicate "failure". By running P_1 and calling E_1, we can construct an extractor E_0. When E_1 outputs $\perp$, E_0 outputs $\perp$ and stops. If E_1 outputs the witness $(\phi, \kappa, \delta, \mu, \nu_1, ..., \nu_m, \varsigma_{11}, ..., \varsigma_{mn}, \rho_1, \rho_2)$, the extractor could further output some valid witness $(\phi, \pi_T, \pi_C, \pi_{A_1}, ..., \pi_{A_m}, \pi_{Q_{11}}, ..., \pi_{Q_{mn}})$ with the same probability. Based on the outputs of E_1, E_0 computes

$$\begin{aligned}
&\hat{\pi}_T = \pi_T\hat{h}^{-\kappa}, \hat{\pi}_C = \pi_C\hat{h}^{-\mu}, \{\hat{\pi}_{A_j} = \pi_{A_j}\hat{h}^{-\nu_j}\}_{j\in[1,m]},\\
&\{\hat{\pi}_{Q_{ji}} = \pi_{Q_{ji}}\hat{h}^{-\varsigma_{ji}}\}_{j\in[1,m], i\in[1,n]}.
\end{aligned}$$

We show how these values satisfy the equation relation of P_0 as follows.
Due to soundness of P_1,

$$e(\pi_T, g\cdot\pi_C) = e(\pi_T, \hat{h}^{\mu}) \cdot e(\hat{h}^{\kappa}, g\cdot\pi_C) \cdot e(\hat{h},\hat{h})^{-\rho_1} \cdot e(Ug, g).$$

Rearranging the terms, where $\rho_1 = \kappa\mu$:

$$e(\pi_T, g\pi_C) \cdot e(\hat{h}^{-\kappa}, g\pi_C) \cdot e(\pi_T, \hat{h}^{-\mu}) \cdot e(\hat{h},\hat{h})^{\rho_1} = e(Ug, g).$$

That is

$$\begin{aligned}
&e(\pi_T, g\pi_C) \cdot e(\hat{h}^{-\kappa}, g\pi_C) \cdot e(\pi_T, \hat{h}^{-\mu}) \cdot e(\hat{h},\hat{h})^{\rho_1}\\
&= e(\pi_T\hat{h}^{-\kappa}, g\pi_C) \cdot e(\pi_T\hat{h}^{-\kappa}, \hat{h}^{-\mu}) \cdot e(\hat{h}^{\kappa}, \hat{h}^{-\mu}) \cdot e(\hat{h},\hat{h})^{\kappa\mu}\\
&= e(\hat{\pi}_T, g\hat{\pi}_C) = e(Ug, g).
\end{aligned}$$

Due to soundness of P_1,

$$e(\pi_C, C_2') \cdot \textstyle\prod_{j=1}^{m} \frac{e(\pi_{A_j}, C_1')}{\prod_{i=1}^{n} e(\pi_{Q_{ji}}, C_{ji}')} = C_T \cdot e(\hat{h}, C_2')^{\mu} \cdot \prod_{j=1}^{m} \frac{e(\hat{h}, C_1')^{\nu_j}}{\prod_{i=1}^{n} e(\hat{h}, C_{ji}')^{\varsigma_{ji}}}.$$

Rearranging the terms:

$$e(\pi_C, C_2^{'}) \cdot \prod_{j=1}^{m} \frac{e(\pi_{A_j}, C_1^{'})}{\prod_{i=1}^{n} e(\pi_{Q_{ji}}, C_{ji}^{'})} \cdot e(\hat{h}, C_2^{'})^{-\mu} \cdot \prod_{j=1}^{m} \frac{e(\hat{h}, C_1^{'})^{-\nu_j}}{\prod_{i=1}^{n} e(\hat{h}, C_{ji}^{'})^{-\varsigma_{ji}}} = C_T.$$

That is

$$\begin{aligned} & e(\pi_C, C_2^{'}) \cdot \prod_{j=1}^{m} \frac{e(\pi_{A_j}, C_1^{'})}{\prod_{i=1}^{n} e(\pi_{Q_{ji}}, C_{ji}^{'})} \cdot e(\hat{h}, C_2^{'})^{-\mu} \cdot \prod_{j=1}^{m} \frac{e(\hat{h}, C_1^{'})^{-\nu_j}}{\prod_{i=1}^{n} e(\hat{h}, C_{ji}^{'})^{-\varsigma_{ji}}} \\ &= e(\pi_C \hat{h}^{-\mu}, C_2^{'}) \cdot \prod_{j=1}^{m} \frac{e(\pi_{A_j} \hat{h}^{-\nu_j}, C_1^{'})}{\prod_{i=1}^{n} e(\pi_{Q_{ji}} \hat{h}^{-\varsigma_{ji}}, C_{ji}^{'})} \\ &= e(\hat{\pi}_C, C_2^{'}) \cdot \prod_{j=1}^{m} \frac{e(\hat{\pi}_{A_j}, C_1^{'})}{\prod_{i=1}^{n} e(\hat{\pi}_{Q_{ji}}, C_{ji}^{'})} = C_T. \end{aligned}$$

Then E_0 could output $(\phi, \hat{\pi}_T, \hat{\pi}_C, \hat{\pi}_{A_1}, ..., \hat{\pi}_{A_m}, \hat{\pi}_{Q_{11}}, ..., \hat{\pi}_{Q_{mn}})$ as the witnesses satisfying P_0. Therefore P_0 is of soundness.

6. Security Proof

In this section, we analyze the security of our scheme and show that it has the properties of task confidentiality, one-time authentication and anonymity.

Assume there exists a PPT adversary $\mathcal{A}$ that wins the following games in our scheme, we can construct a PPT simulator $\mathcal{B}$ that solves the CDH problem, DDH problem or the q-DDHI problem with nonnegligible advantage.

Theorem 1. *Assume the DDH assumption holds, then the proposed zk-MIPE scheme is IND-CPA secure.*

Proof. Against an adversary who wants to learn task content, the security algorithms are designed as follows. □

Algorithm I

- **Init.** The challenger sets $\vec{G} = (G_1, G_T)$ and randomly chooses $(g, g^a, g^b, g^c) \in G_1$. It flips a coin $\bar{b}$ outside of $\mathcal{B}_1$'s view and sets T as follows:
  If $\bar{b} = 0$, it computes $T = e(g,g)^{abc}$; otherwise, it chooses a random $T \in G_2$. Then it sends (g, g^a, g^b, g^c, T) to $\mathcal{B}_1$. After that, $\mathcal{A}_1$ submits the challenge access structure $\vec{w}^* = (w_1^*, ..., w_m^*)$ to $\mathcal{B}_1$.
- **CA Setup.** Given the secure parameter λ, $\mathcal{B}_1$ randomly chooses $t \in Z_q$ and sets $Y = g^t$. Then it gives the public key $PK = \{Y, H_1\}$ to $\mathcal{A}_1$.
- **AA Setup.** $\mathcal{B}_1$ randomly chooses $\{x_j, \eta_{ji}\}_{j\in[1,m], i\in[1,n]}$ at first. Here, we suppose $AA_{\hat{j}}$ is one of the honest attribute authority.

(1) For $j \neq \hat{j}$, $\mathcal{B}_1$ sets $\alpha_j = x_j$, $\gamma_{ji} = \eta_{ji}$ and lets $SK_j = \{\alpha_j, \gamma_{ji}\}_{i\in[1,n]}$ for AA_j. Then it computes $h_{ji} = g^{\eta_{ji}}$ and sends the public key $PK_j = \{e(g,g)^{\alpha_j}, h_{j1}, ..., h_{jn}\}$ to $\mathcal{A}_1$.

(2) For $j = \hat{j}$, $\mathcal{B}_1$ sets $\alpha_{\hat{j}} = ab + x_{\hat{j}}$, $\gamma_{ji} = -w_{\hat{j}i}^* b + \eta_{\hat{j}i}$ and lets $SK_{\hat{j}} = \{\alpha_{\hat{j}}, \gamma_{\hat{j}i}\}_{i\in[1,n]}$ for $AA_{\hat{j}}$. Then it computes $h_{\hat{j}i} = g^{-w_{\hat{j}i}^* b + \eta_{\hat{j}i}}$ and sends $PK_{\hat{j}} = \{e(g,g)^{\alpha_{\hat{j}}}, h_{\hat{j}1}, ..., h_{\hat{j}n}\}$ to $\mathcal{A}_1$.

- **Registration Queries I.** $\mathcal{A}_1$ repeatedly makes registration queries with respect to attribute key value $\vec{z}$ such that $< w_{\hat{j}}^*, z_{\hat{j}} > \neq 0$. Notice that for any other honest AA_j, $\mathcal{B}_1$ will also respond the corresponding secret key even if $< w_j^*, z_j >= 0$.
  $\mathcal{A}_1$ chooses a user u and sets U as his/her public key. It sends U to $\mathcal{B}_1$. Then $\mathcal{B}_1$ chooses random $\beta_{u_1}, ..., \beta_{u_m} \in Z_q$, sets $\beta_u = \sum_{j=1}^{m} \beta_{u_j}$ and computes $CK_u = g^{\frac{\beta_u}{t}}$, $T_u = (Ug)^{\frac{t}{\beta_u + t}}$. After that $\mathcal{B}_1$ computes the attribute related secret key as follows.

(1) For $j \neq \hat{j}$, $\mathcal{B}_1$ chooses a random $\tau_{u_j} \in Z_q$ and computes

$$Q_{u_j} = g^{\tau_{u_j}}, AK_{u_j} = g^{\alpha_j - \beta_{u_j}} \cdot Q^{\sum_{i=1}^{n} \gamma_{ji} z_{ji}}.$$

(2) For $j = \hat{j}$, $\mathcal{B}_1$ randomly chooses a $\tau_{u_{\hat{j}}} \in Z_q$ and computes
$Q_{u_{\hat{j}}} = g^{\frac{a}{<w^*_{\hat{j}}, z_{\hat{j}}>} + \tau_{u_{\hat{j}}}}, AK_{u_{\hat{j}}} = g^{\alpha_{\hat{j}} - \beta_{u_{\hat{j}}}} \cdot Q_{u_{\hat{j}}}^{\sum_{i=1}^{n} \gamma_{\hat{j}i} z_{\hat{j}i}} = g^{x_{\hat{j}} - \beta_{u_{\hat{j}}} + \frac{a<\eta_{\hat{j}}, z_{\hat{j}}>}{<w^*_{\hat{j}}, z_{\hat{j}}>} - b\tau_{u_{\hat{j}}}<w^*_{\hat{j}}, z_{\hat{j}}> + \tau_{u_{\hat{j}}}<\eta_{\hat{j}}, z_{\hat{j}}>}.$

- **Challenge.** $\mathcal{A}_1$ submits two challenge messages $M_0, M_1 \in G_T$ to $\mathcal{B}_1$. $\mathcal{B}_1$ flips a coin $b \in \{0, 1\}$ and computes the ciphertext as follows.
  $\mathcal{B}_1$ chooses a random $(\varphi, s_2) \in Z_q$, sets $s_1 = c + \varphi$ and computes
  $C_0^* = M_b \cdot e(g, g)^{\sum_{j=1}^{m} s_1 \alpha_j} = M_b \cdot T \cdot e(g, g)^{\sum_{j=1}^{m} (c+\varphi) x_j + ab\varphi}$, $C_1^* = g^{s_1} = g^{c+\varphi}$, $C_2^* = Y^{s_1} = Y^{c+\varphi}$, $C_1'^* = g^{s_2}$, $C_2'^* = Y^{s_2}$, $C_{ji}'^* = h_{ji}^{s_2} g^{\sigma_2 w^*_{ji}} = g^{\eta_{ji} s_2 + \sigma_2 w^*_{ji}}$, $C_T^* = e(g, g)^{\sum_{j=1}^{m} s_2 \alpha_j}$.
  Then $\mathcal{B}_1$ computes C^*_{ji} as follows.

(1) For $j \neq \hat{j}$, $\mathcal{B}_1$ chooses a random $\theta \in Z_q$, sets $\sigma_1 = \theta$ and computes

$$C^*_{ji} = h_{ji}^{s_1} \cdot g^{\sigma_1 w^*_{ji}} = g^{\eta_{ji}(c+\varphi) + \theta w^*_{ji}}.$$

(2) For $j = \hat{j}$, $\mathcal{B}_1$ chooses a random $\theta \in Z_q$, sets $\sigma_1 = bc + \theta$ and computes
$C^*_{\hat{j}i} = h_{\hat{j}i}^{s_1} \cdot g^{\sigma_1 w^*_{\hat{j}i}} = g^{-w^*_{\hat{j}i} b\varphi + \eta_{\hat{j}i}(c+\varphi) + \theta w^*_{\hat{j}i}}$.

- **Registration Queries II.** $\mathcal{A}_1$ submits a polynomially bounded number of registration queries with respect to attribute sets $\vec{z}_1, ..., \vec{z}_q$. $\mathcal{B}_1$ responds as it did in **Registration Queries I**.
- **Guess.** $\mathcal{A}_1$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}_1$ will guess T is a DDH tuple, i.e., $\bar{b} = 0$; otherwise, it guesses T is a random tuple, i.e., $\bar{b} = 1$. It indicates that if the adversary wins this game with nonnegligible advantage, then the simulator will have obviously advantage in the DDH game.

Theorem 2. *Assume the CDH assumption holds, then the proposed zk-MIPE scheme is one-time authenticate.*

Proof. Against an adversary who wants to forge a valid proof for the attributes he/she does not possess, the security algorithms are designed as follows.

In our scheme, for each task, the value of a tag $S = e(g, g)^{\frac{1}{\phi + H(C_0)}}$ submitted by a user u is different and unique fixed. If submitting a tag twice will be forbidden. Thus, as follows, we show that it is difficult for unsuitable workers to forge a valid authentication message based on the CDH assumption. □

Algorithm II

- **Init.** The challenger sets $\vec{G} = (G_1, G_T)$ and randomly chooses $(g, g^a, g^b) \in G_1$. Then it sends g, g^a, g^b to $\mathcal{B}_2$. After that, $\mathcal{A}_2$ submits the challenge access structure and message $(\vec{w}^*, M^*)$.
- **CA Setup.** Running the CA setup algorithm, $\mathcal{B}_2$ does as in Algorithm I.
- **AA Setup.** Running the AA setup algorithm, $\mathcal{B}_2$ does as in Algorithm I.
- **Registration Queries I.** Running the registration algorithm, $\mathcal{B}_2$ does as in Algorithm I.
- **Verification Queries I.** $\mathcal{A}_2$ submits a series of queries about $(M_k, \vec{w}_k, \vec{z}_k)$ to $\mathcal{B}_2$. It requires that $\vec{w}_k \neq \vec{w}^*$, $< \vec{w}_k, \vec{z}_k >= 0$ and $< \vec{w}^*, \vec{z}_k > \neq 0$, and if not, it aborts. $\mathcal{B}_2$ runs matching and detection verification algorithm, interacts with $\mathcal{A}_2$, and generates proof transcript for $(M_k, \vec{w}_k, \vec{z}_k)$.
- **Forgery.** For the specified $(M^*, \vec{w}^*)$, $\mathcal{A}_2$ chooses a worker public key U^* and an attribute vector $\vec{z}^*$ such that $< \vec{w}^*, \vec{z}^* >= 0$. In this algorithm, we will not consider the privacy of $\vec{w}^*$. Based on $\vec{w}^*$, $\mathcal{B}_2$ computes ciphertext about message M^*. Then $\mathcal{A}_2$ interacts with $\mathcal{B}_2$ to generate a transaction of the protocol P_0, proving that it has the private key about a suitable vector. If $\mathcal{A}_2$ outputs a valid forged proof and the protocol is sound, $\mathcal{B}_2$ could then obtain g^{ab} from the forgery.

Theorem 3. *Suppose that the q-DDHI assumption holds and the protocol P_0 is zero-knowledge, then the proposed scheme is private and unlinkable.*

Proof. To prove the privacy of the scheme, we first summarize the zero-knowledge of P_0.

Zero-knowledgeness of P_0. For the implementation of P_0, we introduced some auxiliary inputs $(\pi_T, \chi_T, \pi_C, \{\pi_{A_j}, \pi_{Q_{ji}}\}_{j\in[1,m],i\in[1,n]})$ and protocol P_1. Based on the Logarithm assumption and the DDH assumption, the zero-knowledge property of P_1 is guaranteed for honest verifier, i.e., there exists a simulator $\mathcal{S}$ on imputing a random challenge ε, the simulator could output a transcript for $(L_1, ..., L_5, z_\phi, z_\kappa, z_\delta, z_\mu, z_{v_1}, ..., z_{v_m}, z_{\varsigma_{11}}, ..., z_{\varsigma_{mn}}, z_{\rho_1}, z_{\rho_2})$. For any adversary, the distribution of the output is indistinguishable. By invoking $\mathcal{S}$ the simulator of protocol P_1, protocol P_0 could further prove its zero-knowledge property. □

Then we define the game between an adversary $\mathcal{A}_3$ and a simulator $\mathcal{B}_3$ which is given a q-DDHI instance as follows.

Algorithm III

- **Init.** The challenger sets $\vec{G} = (G_1, G_T)$ and randomly chooses $g, g^x, g^{x^2}, ..., g^{x^q} \in G_1$. It flips a coin $\bar{b}$. If $\bar{b} = 0$, it computes $T = e(g,g)^{\frac{1}{x}}$; otherwise, it chooses a random $T \in G_T$. After that, $\mathcal{A}_3$ submits two challenge users u_0, u_1 with attribute vector $\vec{z_0}$, $\vec{z_1}$ to $\mathcal{B}_3$.
- **CA Setup.** Given the secure parameter λ, $\mathcal{B}_3$ chooses a random $t \in Z_p$ and sets $Y = g^t$. Then it gives the public key $PK = \{Y, H_1\}$ to $\mathcal{A}_3$.
- **AA Setup.** $\mathcal{B}_3$ randomly chooses $\{x_j, \eta_{ji}\}_{j\in[1,m],i\in[1,n]}$, sets $\alpha_j = x_j$, $\gamma_{ji} = \eta_{ji}$ and lets $SK_j = \{\alpha_j, \gamma_{ji}\}_{i\in[1,n]}$ for AA_j. Then it computes $h_{ji} = g^{\eta_{ji}}$ and sends the public key $PK_j = \{e(g,g)^{\alpha_j}, h_{j1}, ..., h_{jn}\}$ to $\mathcal{A}_3$.
- **Registration Queries I.** $\mathcal{B}_3$ sets $CK_{u^*} = g^x$ for a user u^* and receives the value T_{u^*}, which may equal to $g^{\frac{1}{x+1}}$ or a random element in G_1, from the challenger initially. $\mathcal{A}_3$ issues registration queries repeatedly. $\mathcal{B}_3$ generates the secret key honestly except for u^*. If $u_i = u^*$, it aborts. Moreover, it is required that $\mathcal{A}_3$ does not make secret key queries for both u_0 and u_1.
- **Challenge.** Without loss of generality, $\mathcal{B}_3$ assumes $u_0 = u^*$. It flips a coin $b \in \{0,1\}$ and runs registration queries to obtain the corresponding K_{u_b}. Then, $\mathcal{B}_3$ operates **Task Releasing** with an attribute vector $\vec{w}^*$ (with restrictions that $< \vec{w}^*, \vec{z_0} >= 0$ and $< \vec{w}^*, \vec{z_1} >= 0$) to acquire the ciphertext C^*. After receiving C^*, $\mathcal{A}_3$ issues **Verification** and receives a valid proof from $\mathcal{B}_3$ by applying the zero-knowledge protocol P_0.
- **Registration Queries II.** $\mathcal{A}_3$ submits a polynomially bounded number of registration queries repeatedly. $\mathcal{B}_3$ responds as it did in **Registration Queries I**.
- **Guess.** $\mathcal{A}_3$ outputs a guess b' of b. If $b' = b$, $\mathcal{B}_3$ will guess T is a q-DDHI tuple, i.e., $\bar{b} = 0$; otherwise, it guesses T is a random tuple, i.e., $\bar{b} = 1$. Observe that if H is a one-way pseudo-random hash function and the q-DDHI assumption holds, the adversary will know nothing about β_u. By the zero-knowledge property of protocol P_0, the information about the identity U, the policy $\vec{w}$ and the attribute $\vec{z}$ will not be leaked. Thus, the algorithm could protect identity privacy and submission unlinkability.

7. Performance Evaluation

In reality, we implement the ZK-MIPE scheme on a Linux desktop with 6-core Intel(R) Xeon(R) Platinum 8369HC CPU 3.40 GHz processor and 32 GB of RAM. We use the PBC library to simulate the group operations. The symmetric elliptic curve SS512 is chosen with embedding degree 2 and a 512-bit base field.

Tables 1 and 2 show the comparison between our scheme and other solutions in terms of functionality and security. Compared with [24], zk-MIPE supports more flexible matching poly and supports worker identity privacy. Compared with [23,25], zk-MIPE provides privacy for task constraints and worker attributes. As follows, we analyze the computational complexity of each participant in our scheme and test the running time to demonstrate scheme's effectiveness. The notations applied in the proposed scheme are summarized in Table 3. Ignoring the operations of equality comparison, hash and multiplication, the communication and computation comparison of the schemes is shown in Tables 4 and 5.

Table 1. Functional comparison.

Scheme	Authority	Matching Policy	Repetition Detection	Multi-Keyword
Fo-SDD	Single	Unlimited	$\times$	$\times$
SybSub	Single	Range	$\times$	$\times$
ZebraLancer	Distribute	Unlimited	$\times$	$\times$
zk-MIPE	Multiple	Inner-Product	$\times$	$\times$

Table 2. Security comparison.

Scheme	Task Content Privacy	Task Constraint Privacy	Identity Privacy	Attribute Privacy
Fo-SDD	$\surd$	$\times$	$\times$	$\times$
SybSub	$\times$	$\surd$	$\times$	$\surd$
ZebraLancer	$\times$	$\times$	$\surd$	$\times$
zk-MIPE	$\surd$	$\surd$	$\surd$	$\surd$

Table 3. Notations in Fo-SDD, SybSub, ZebraLancer and zk-MIPE.

Notations	Description
E_1, E_T	Exponentiation on G_1 and G_T respectively
E_2	Exponentiation in Paillier encryption
P	Pairing on $(G_1, G_1) \to G_T$
$\tilde{C}_s, \tilde{C}_a$	Ciphertext based on AES and public key encryption, respectively
C_{sc}	Coding a task into a smart contract
$\tilde{E}_s, \tilde{E}_a, \tilde{E}_P$	Symmetric encryption, asymmetric encryption and Paillier encryption
$\tilde{D}_s, \tilde{D}_a, \tilde{D}_P$	Symmetric decryption, asymmetric decryption and Paillier decryption
l_1, l_2, l_3	Bit length of task, attribute and smart contract, respectively
M	NP machine used to prove membership of an instance x in a given NP language L
t_M, s_M	Operations and computation space of M for the instance x
poly	Universal polynomial
λ	Security parameter
N	Product of two primes
m	Number of attribute authorities
n	Dimension of the attribute-based vector
l	Number of attributes managed by each authority
k	Number of suitable workers

Table 4. Communication cost.

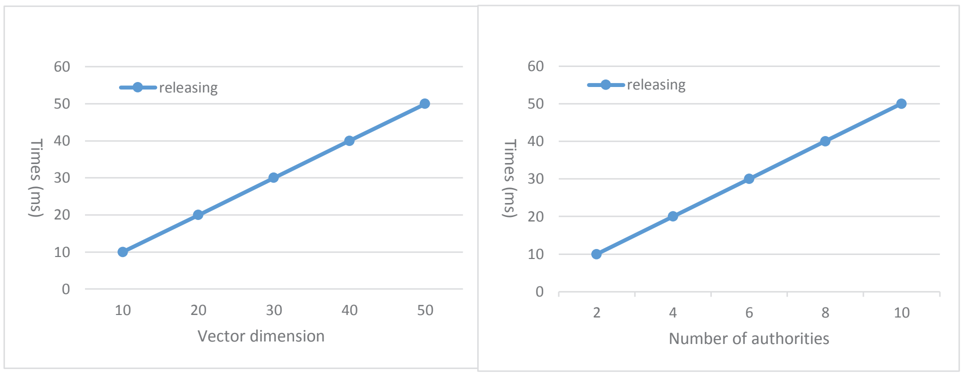
Scheme	Requester/Publisher	CS/Contract	Worker/Subscriber	Fog Node
Fo-SDD	$\|G_1\| + \|G_2\| + \|\tilde{C}_s\| + l_1$	$k(\|Z_q\| + \|G_2\| + \|\tilde{C}_s\| + l_1)$	$2\|Z_q\| + 4\|G_1\| + 2\|\tilde{C}_s\|$	$k(\|Z_q\| + 4\|G_1\| + \|\tilde{C}_s\| + l_1)$
SybSub	$l_2 + \|Z_{N^2}\|$	l_1	$2\|G_1\| + l_2 + \|Z_{N^2}\|$	-
ZebraLancer	l_3	$k\|\tilde{C}_a\|$	$\|\tilde{C}_a\| + s_M\text{poly}(\lambda)$	-
zk-MIPE	$(5 + 2nm)\|G_1\| + \|G_2\|$	$2k\|G_1\| + k\|Z_q\|$	$5\|G_T\| + (7 + m + nm)(\|Z_q\| + \|G_1\|)$	-

Table 5. Computation cost.

Scheme	Requester/Publisher	CS/Contract	Worker	Fog Node
Fo-SDD	$2E_1 + \tilde{E}_s$	$k(E_1 + E_2 + P + \tilde{D}_s)$	$6E_1 + 2E_T + 2P + \tilde{D}_s$	$E_T + \tilde{D}_s + k(3E_1 + \tilde{E}_s)$
SybSub	$lE_2 + 2l\tilde{E}_P$	$kl(E_1 + \tilde{D}_P) + l(k+1)P$	$3lE_1 + 2l\tilde{E}_P + lE_2$	-
ZebraLancer	$E_1 + C_{sc} + t_M\text{poly}(\lambda)$	$k\text{poly}(\lambda)$	$\tilde{E}_a + t_M\text{poly}(\lambda)$	-
zk-MIPE	$(4 + 4nm)E_1 + 2E_T$	$5kE_1 + (10 + m + nm)kE_T + (5 + 2m + 2nm)kP$	$(9 + m + nm)E_1 + (4 + n + m)(E_T + P) + 2E_T$	-

In our scheme, the main overhead on CA and AA_j are from system setup and user registration. In CA setup, the computation complexity of CA is E_1. In AA setup, the computation complexity of AA_j is $nE_1 + P$. In user registration, the computation complexity of CA and AA_j are $2kE_1$ and $3kE_1$, respectively. The total communication complexity of the authorities for distributing a key to a registered user is $m(Z_q + 3|G_1|)$.

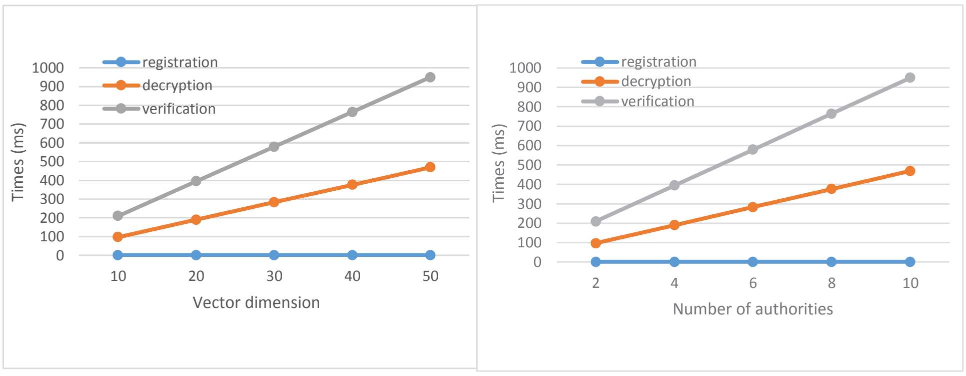
The main overhead on the requester is from task releasing. In this step, the requester expresses the task requirements with vector $\vec{w}$ and encrypt the task based on $\vec{w}$ such that only the suitable worker could decrypt the task content. Meanwhile, the requester is required to blind the vector $\vec{w}$ for the CS to perform matching verification in the matching and submission verification phase. The computation complexity of the requester is $(4 + 4nm)E_1 + 2E_T$. The total communication complexity of the requester for task releasing is $(5 + 2nm)|G_1| + |G_2|$. To test the time cost of the requester, we set the number of attribute authorities as $m = 5$, and vary the number of attributes n in Figure 3a. In Figure 3b, we set $n = 20$ and vary the number of attribute authorities m.



(**a**) (**b**)

Figure 3. Time cost on the requester (**a**) under different values of n where $m = 5$; (**b**) under different values of m where $n = 20$.

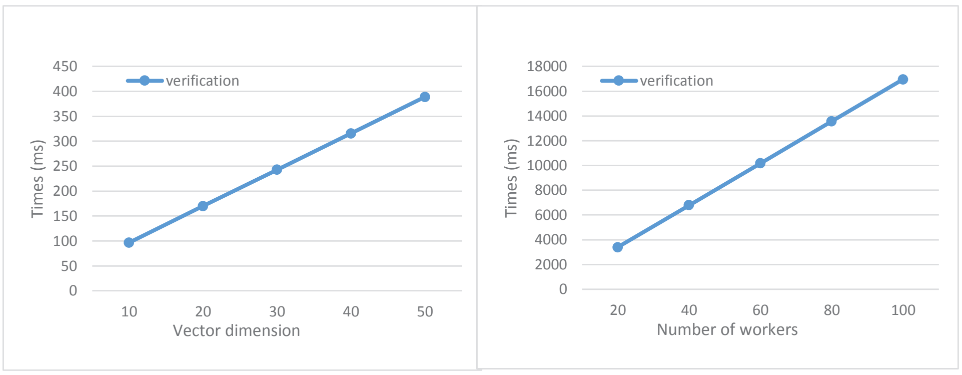
The main overhead on the worker is from registration, decryption and verification. As shown in Figure 4a, we set $m = 5$, and vary the number of attributes to test the time cost on the worker. In Figure 4b, we set $n = 20$ and vary the number of attribute authorities. In user registration and decryption, the computation complexity of the worker is $E_1 + nmE_1 + (m + n + 1)P$. Although in decryption algorithm, the computing cost for the worker increases linearly with the number of attributes, most of the computing overhead can be transferred to the CS by outsourcing computing. In this case, the worker only needs to carry out a small amount of calculation.



(**a**) (**b**)

Figure 4. Time cost on the worker (**a**) under different values of n where $m = 5$; (**b**) under different values of m where $n = 20$.

In the stage of submission and verification, the worker and CS achieve privacy-preserving matching and multi-submission verification through a zero-knowledge proof protocol. The interactive proof protocol consists of 3 rounds. The total computation and communication complexity of the worker are $(9 + m + nm)E_1 + (6 + m + n)E_T + (4 + n + m)P$ and $5|G_T| + (7 + m + nm)(|z_q| + |G_1|)$ respectively. In Figure 5, we take $m = 5$, and vary n as well as the number of workers k to test the time cost of verification for CS. The total computation and communication complexity of CS are $5E_1 + (10 + m + nm)E_T + (5 + 2m + 2nm)P$ and $2|G_1| + |z_q|$, respectively.



(**a**) (**b**)

Figure 5. Time cost on CS (**a**) under different values of n where $m = 5$ and $k = 1$; (**b**) under different values of k where $m = 5$ and $n = 20$.

8. Conclusions

In this paper, we present a novel multi-authorities inner-product encryption and one-time anonymous authentication scheme to realize privacy-preserving task-matching and multi-submission detection. In the system, both the user attributes and the number of submissions will be applied as authorization factors. By combining zero-knowledge proof technology and our anti-collusion multi-authorities inner-product encryption, the task confidentiality, worker attribute and unlinkability between different tasks participated by the same worker are guaranteed simultaneously. Moreover, the security of the scheme is proved based on bilinear difficulty assumptions and zero-knowledge of the protocol. For the sake of completeness, we finally analyze the function and efficiency of the scheme and show that it is practical for crowdsourcing environments. In addition to crowdsourcing privacy protection, our method could also play its role in the fields of searchable encryption, nearest neighbor search, fine-grained access control, electronic voting, electronic payment, and anonymous authentication.

In future work, we will continue to improve the algorithm itself and try to construct privacy protection schemes in a distributed crowdsourcing scenario without a trusted third party. Furthermore, we will study the integration of cryptography and other technologies, such as machine learning technology, to further improve the flexibility and efficiency of the solution.

Author Contributions: Conceptualization, J.X. and Z.L.; methodology, J.X.; validation, J.W.; formal analysis, J.X.; writing—original draft preparation, J.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (Grant No. 61802025), the Beijing Talents Foundation, China (Grant No. 2017000020124G062) and the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20170802).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Von Ahn, L.; Maurer, B.; McMillen, C.; Abraham, D.; Blum, M. Recaptcha: Human-based character recognition via Web security measures. *Science* **2008**, *321*, 1465–1468. [CrossRef] [PubMed]
2. McInnis, B.; Cosley, D.; Nam, C.; Leshed, G. Taking a HIT: Designing around rejection, mistrust, risk, and workers' experiences in Amazon Mechanical Turk. In Proceedings of the CHI '16: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, 7–12 May 2016; pp. 2271–2282.
3. Liu, A.; Wang, W.; Shang, S.; Li, Q.; Zhang, X. Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. *Geoinformatica* **2018**, *22*, 335–362. [CrossRef]
4. Xu, J.; Cui, B.; Shi, R.; Feng, Q. Outsourced privacy-aware task allocation with flexible expressions in crowdsourcing. *Future Gener. Comput. Syst.* **2020**, *112*, 383–393. [CrossRef]
5. Chase, M.; Chow, S.S.M. Improving privacy and security in multi-authority attribute-based encryption. In Proceedings of the CCS '09: 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009; pp. 121–130.
6. Xiong, J.; Ma, R.; Chen, L.; Tian, Y.; Li, Q.; Liu, X.; Yao, Z. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4231–4241. [CrossRef]
7. Lin, C.; He, D.; Zeadally, S.; Kumar, N.; Choo, K.R. SecBCS: A secure and privacy-preserving blockchain-based crowdsourcing system. *Sci. China Inf. Sci.* **2020**, *63*, 130102:1–130102:14. [CrossRef]
8. Feng, W.; Yan, Z. MCS-chain: decentralized and trustworthy mobile crowdsourcing based on blockchain. *Future Gener. Comput. Syst.* **2019**, *95*, 649–666. [CrossRef]
9. Li, M.; Weng, J.; Yang, A.; Lu, W.; Zhang, Y.; Hou, L.; Deng, R.H. CrowdBC: A blockchain-based decentralized framework for crowdsourcing. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 1251–1266. [CrossRef]
10. Wei, J.; Wang, X.; Li, N.; Yang, G.; Mu, Y. A Privacy-preserving fog computing framework for vehicular crowdsensing networks. *IEEE Access* **2018**, *6*, 43776–43784. [CrossRef]
11. Vu, K.; Zheng, R.; Gao, J. Efficient algorithms for k-anonymous location privacy in participatory sensing. In Proceedings of the 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 2399–2407.
12. Andrés, M.E.; Bordenabe, N.E.; Chatzikokolakis, K.; Palamidessi, C. Geoindistinguishability: differential privacy for location-based systems. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security CCS'13, Berlin, Germany, 4–8 November 2013; pp. 901–914.
13. Wei, J.; Lin, Y.; Yao, X.; Zhang, J. Differential privacy-based location protection in spatial crowdsourcing. *IEEE Trans. Services Comput.* **2019**, *16*, 934–949. [CrossRef]
14. Xiao, L.; Li, Y.; Han, G.; Dai, H.; Poor, H.V. A secure mobile crowdsensing game with deep reinforcement learning. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 35–47. [CrossRef]
15. Knirsch, F.; Unterweger, A.; Engle, D. Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions. *Comput. Sci. Res. Dev.* **2018**, *33*, 71–79. [CrossRef]
16. Yang, D.; Qu, B.; Cudré-Mauroux, P. Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE Trans. Konwl. Data Eng.* **2019**, *31*, 507–520. [CrossRef]
17. Rantos, K.; Drosatos, F.; Demertzis, K.; Ilioudis, D.; Papanikolaou, A. ADvoCATE: A consent management platform for personal data processing in the IoT using blockchain technology. In Proceedings of the SECITC 2018: Innovative Security Solutions for Information Technology and Communications, Bucharest, Romania, 8–9 November 2018; pp. 300–313.
18. Yi, X.; Paulet, R.; Bertino, E.; Varadharajan, V. Practical approximate k nearest neighbor queries with location and query privacy. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 1546–1559. [CrossRef]

19. Agir, B.; Papaioannou, T.G.; Narendula, R.; Jean-pierre, K.A. User-side adaptive protection of location privacy in participatory sensing. *Geoinformatica* **2014**, *18*, 165–191. [CrossRef]
20. Shu, J.; Jia, X.; Yang, K.; Wang, H. Privacy-preserving task recommendation services for crowdsourcing. *IEEE Trans. Services Comp.* **2021**, *14*, 235–247. [CrossRef]
21. Liu, S.S.; Liu, A.; Yan, Z.; Feng, W. Efficient LBS queries with mutual privacy preservation in IoV. *Veh. Commun.* **2019**, *16*, 62–71. [CrossRef]
22. Yuan, D.; Li, Q.; Li, G.; Wang, Q.; Ren, K. PriRadar: A privacy-preserving framework for spatial crowdsourcing. *IEEE Trans. Inform. Forensics Secur.* **2020**, *15*, 299–314. [CrossRef]
23. Ni, J.; Zhang, K.; Yu, Y.; Lin, X.; Shen, X.S. Providing task allocation and secure deduplication for mobile crowdsensing via fog computing. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 581–594. [CrossRef]
24. Shu, J.; Liu, X.; Yang, K.; Zhang, Y.; Jia, X.; Deng, R.H. SybSub: Privacy-preserving expressive task subscription with sybil detection in crowdsourcing. *IEEE Internet Things J.* **2019**, *6*, 3003–3013. [CrossRef]
25. Lu, Y.; Tang, Q.; Wang, G. ZebraLancer: private and anonymous crowdsourcing system atop open blockchain. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018; pp. 853–865.
26. Shamir, A. Identity-based cryptosystems and signature schemes. In *Cryptology Proceedings of CRYPTO 84*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 47–53.
27. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. In Proceedings of the CRYPTO 2001—21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 213–229.
28. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In Proceedings of the EUROCRYPT 2005—24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; PP. 457–473.
29. Han, J.; Susilo, W.; Mu, Y.; Yan, J. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 2150–2162. [CrossRef]
30. Jung, T.; Li, X.Y.; Wan, Z.; Wan, M. Privacy preserving cloud data access with multi-authorities. In proceedings of the 2013 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2625–2633.
31. Jung, T.; Li, X.Y.; Wan, Z.; Wan, M. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. *IEEE Trans. Inform. Forensics Secur.* **2015**, *10*, 190–199. [CrossRef]
32. Boneh, D.; Waters, B. Conjunctive, subset, and range queries on encrypted Data. In Proceedings of the 4th Theory of Cryptography Conference, TCC, Amsterdam, The Netherlands, 21–24 February 2007; pp. 535–554.
33. Katz, J.; Sahai, A.; Waters, B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Proceedings of the EUROCRYPT 2008—Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, 13–17 April 2008; pp. 146–162.
34. Attrapadung, N.; Libert, B. Functional encryption for inner product: achieving constant-size ciphertexts with adaptive security or support for negation. In Proceedings of the PKC 2010—13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, 26–28 May 2010; pp. 384–402.
35. Okamoto, T.; Takashima, K. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. *Des. Codes Cryptogr.* **2015**, *77*, 725–771. [CrossRef]
36. Li, J.; Chen, X.; Li, J.; Jia, C.; Ma, J.; Lou, W. Fine-grained access control system based on outsourced attribute-based encryption. In Proceedings of the ESORICS 2013—18th European Symposium on Research in Computer Security, Egham, UK, 9–13 September 2013; pp. 592–609.
37. Yuen, T.H.; Liu, J.K.; Au, M.H.; Huang, X.; Susilo, W.; Zhou, J. k-times attribute-based anonymous access control for cloud computing. *IEEE Trans. Comput.* **2015**, *64*, 2595–2607. [CrossRef]
38. Ning, J.; Cao, Z.; Dong, X.; Liang, K.; Ma, H.; Wei, L. Auditable σ-time outsourced attribute-based encryption for access control in cloud computing. *IEEE Trans. Inform. Forensics Secur.* **2018**, *13*, 94–105. [CrossRef]
39. Zcash. Available online: https://z.cash (accessed on 30 March 2021).
40. Kosba, A.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Proceedings of the IEEE Symposium on Secur. and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 839–858.