Privacy-Preserving and Verifiable Multi-task Data Aggregation for IoT-based Healthcare

Xinzhe Zhang[a], Lei Wu[a,*], Lijuan Xu[b], Zhien Liu[a], Ye Su[a], Hao Wang[a] and Weizhi Meng[c]

[a]School of Information Science and Engineering, Shandong Normal University, Jinan, 250358, China
[b]Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, 250014, China
[c]School of Computing and Communications, Lancaster University, Lancaster, LA1 4YW, United Kingdom

ARTICLE INFO

Keywords:
Privacy preservation
Multi-task data aggregation
Homomorphic cryptosystem
Verifiability
Commitment

ABSTRACT

The combination of mobile crowdsensing (MCS) and IoT-based healthcare introduces innovative solutions for collecting health data. The considerable accumulation of health data through MCS expedites advancements in medical research and disease prediction, giving rise to privacy considerations. Data aggregation emerges as a salient solution that facilitates the provision of aggregated statistics while obfuscating raw personal data. However, prevailing aggregation schemes primarily pivot around single-task or multi-dimensional data aggregation, rarely contemplating the multi-task aggregation scenarios. Furthermore, in some schemes that implement multi-tasking, protection of task contents and verifiability of aggregation results are not achieved. Therefore, we propose a specialized data aggregation scheme for multi-task scenarios on fog computing. Initially, we employ a symmetric cryptographic algorithm to encrypt task contents and distribute the corresponding symmetric keys through a key management scheme based on the Chinese Remainder Theorem (CRT). Subsequently, we utilize blinding techniques to encrypt the raw data of users, ensuring efficient data aggregation. To enhance resilience against adversarial tampering with aggregated data, we employ the Pedersen commitment scheme to achieve the verifiability of task aggregation results. Finally, theoretical analyses and experimental evaluations collectively demonstrate the security and effectiveness of our proposed scheme.

1. Introduction

Mobile Crowdsensing (MCS) is a novel paradigm that harnesses the capabilities of myriad mobile devices to sense and collect data for broad social engagement and is currently used in the healthcare industry [1], e.g., CrowdMed [2] and ABCrowdMed [3]. In IoT-based healthcare, MCS assumes a crucial role by providing substantial support to government authorities and healthcare organizations in collecting vast volumes of public health data [4]. These health data have the potential to significantly contribute to the progression of medical research and the refinement of disease prediction methodologies. Nevertheless, the collection of health data still encounters substantial challenges.

Primarily, medical devices or sensors connected to the IoT generate massive data traffic, imposing considerable strain on the cloud server. To alleviate the pressure, researchers have introduced fog computing. Fog computing is a new type of computing architecture with significant characteristics [5], which has been widely used in healthcare [6, 7]. A typical fog computing framework is illustrated in Fig. 1. Fog nodes, distributed near user terminals and interfaced with multiple users, constitute indispensable elements within the architectural framework. These fog nodes can collect users' data, preprocess it, and upload it to the cloud servers, significantly reducing server pressure.

Secondly, health data is sensitive, and its leakage threatens individual privacy and identity security and could trigger legal issues and a crisis of trust. This paper contemplates a scenario in which data consumers, such as official authorities or healthcare organizations, initiate data collection tasks aligned with medical requisites. Subsequently, users autonomously select tasks, uploading health data to the healthcare system. The healthcare system, in turn, furnishes medical statistics to the data consumers. These medical statistics aid public authorities in understanding diseases, formulating health policies, and enabling medical researchers to explore new treatments and prevention measures [8]. Evidently, in most real-world applications, statistical values derived from health data prove adequate for meeting specified requirements. For example, an elevated average body temperature or blood pressure may predict the emergence and spread of an epidemic [9, 10]. Consequently, for sensitive health data, the statistical value can be made available to data consumers, but the raw personal data must remain confidential and inaccessible to the public. Privacy-preserving data aggregation is a popular solution, which involves concealing individual raw data by aggregating information from a cohort of users.

Researchers have proposed many privacy-preserving data aggregation schemes [11–17]. However, in most existing schemes, one round of data aggregation can only accomplish one task, which can not be efficiently applied to multi-task aggregation scenarios. Multi-task in this paper

*Corresponding author: Lei Wu
zhangxinzhe2580@163.com (X. Zhang); wulei@sdnu.edu.cn (L. Wu); xulj@sdas.org (L. Xu); lzn548@163.com (Z. Liu); suye@sdnu.edu.cn (Y. Su); wanghao@sdnu.edu.cn (H. Wang); weme@dtu.dk (W. Meng)
ORCID(s): 0009-0003-7149-4036 (X. Zhang); 0000-0002-2691-0243 (L. Wu)

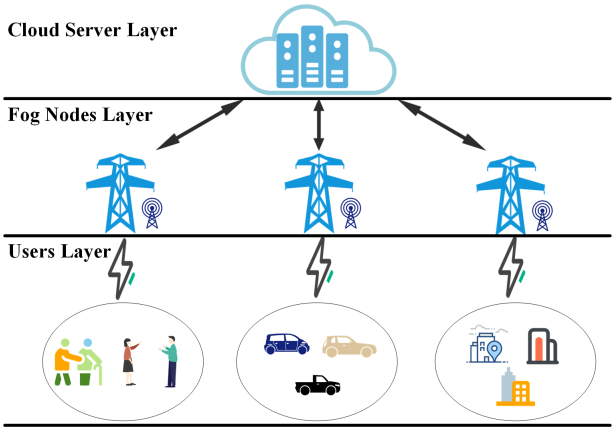


Figure 1: The framework of fog computing.

involves performing data aggregation for several different tasks in a single round. Notably, aggregating multiple tasks in a single round may raise new privacy concerns. For instance: 1) The user may selectively accept aggregated tasks based on considerations such as device limitations or privacy concerns. This choice information, referred to as the user's decision, likewise contains personal privacy, which can lead to inference attacks if leaked. 2) It is imperative to acknowledge that task publishers represent distinct bodies that may compete with each other. Therefore, safeguarding the content of each task becomes essential. Additionally, aggregators are usually untrustworthy in the real world, e.g., returning incorrect aggregation results or tampering with aggregated data. How to ensure the verifiability of task aggregation results is also a point of interest.

To address the above problems, we propose a new multi-task data aggregation scheme for healthcare. Our scheme leverages a CRT-based key management scheme to distribute the decryption key of task contents to the user. Furthermore, we employ Pedersen commitment to ensure the verifiability of aggregation results. The contributions of this paper are summarized as follows.

- We propose a new data aggregation framework for healthcare, which provides privacy-preserving for user's data, user's decision, and task aggregation results. Additionally, our scheme achieves multi-task data aggregation, enabling the simultaneous completion of multiple tasks within a single aggregation round.
- The privacy of task contents is achieved based on a CRT-based key management scheme, ensuring that only the user and the task publisher are aware of them. Utilizing the group key instead of individual key negotiation between the user and task publisher reduces the overhead.
- We guarantee the verifiability of the aggregation results by employing Pedersen commitment scheme, which prevents the adversary or aggregator from tampering with user data during the intermediate process.
- We analyze the proposed scheme and prove its security under the defined security model. We have conducted simulation experiments on the proposed scheme, which is practicable compared to other schemes.

The remainder of this paper is organized as follows. We discuss some related works in Section 2. Then, we review our preliminaries in Section 3. After that, we introduce our system model, threat model, and design goal in Section 4 and present our proposed scheme in Section 5, followed by security analysis and performance evaluation in Section 6 and Section 7, respectively. Finally, we conclude this paper in Section 8.

2. Related Works

Data privacy protection through data aggregation has become an attractive research topic. It is extensively applied in healthcare, smart grids, and crowd sensing. Here, we review our previous work.

In 2016, Han et al. [18] employed differential privacy techniques to facilitate the aggregation of various functions, such as mean and variance. However, the incorporation of noise in the process diminishes the utility and accuracy of the resultant data. Subsequently, Liu et al. [19] propose a scheme that eliminates the need for trusted third parties by creating virtual aggregation regions. Nevertheless, this scheme necessitates users to establish a foundation of trust, thereby constraining its applicability. In a different vein, Wu et al. [20] implemented a versatile data aggregation mechanism based on homomorphic two-trapdoor public key cryptography. However, this scheme requires multiple encryption and decryption operations, leading to a substantial computational overhead. To alleviate this burden, Su et al. [21] introduced a lightweight data aggregation scheme for smart grids, affording smart meters the flexibility to form aggregation zones autonomously. Regrettably, the aforementioned schemes are tailored for single-task scenarios and do not extend to the domain of multi-task aggregation.

Simultaneously, some scholars focus on multidimensional data scenarios. Li et al. [22] proposed a multi-subset aggregation scheme by aggregating different ranges of users' electricity consumption data using two super-growth sequences. Zuo et al. [23] introduced a distributed decryption scheme for multi-dimensional data aggregation without trusted privileges. They utilized the ElGamal homomorphic cryptosystem to resist the joint attack of gateways and control centers. Shen et al. [24] constructed the user's multi-dimensional data into a polynomial based on Horner's rule and devised a privacy-preserving aggregation scheme using the Paillier cryptosystem. However, it is essential to note that these schemes entail substantial computational and communication costs. To reduce the cost, Merad et al. [25] structured the multi-dimensional data and encrypted it into a single Paillier ciphertext. Peng et al. [26] used the CRT to encrypt a multi-dimensional vector of small

integers into a single ciphertext, preserving the linear homomorphism property in each dimension. Liu et al. [27] used the CRT to pack the multi-dimensional data and encrypt it using a key generated by the user in consultation with the control center. In a parallel vein, Cong et al. [28] developed a transformation method with counters that can encode multi-dimensional data into large integers. Building upon this foundation, they designed a multi-functional data aggregation scheme that supports linear, polynomial, and continuous functions.

While these schemes demonstrate efficacy in aggregating multi-dimensional data, their focus is predominantly on addressing the needs of a singular requester. Multi-tasking aggregation for multiple requesters remains challenging. Yan et al. [29] endeavored to address the issue by proposing a multi-task data aggregation scheme tailored for mobile crowdsensing. Unfortunately, their proposed scheme lacks consideration of the privacy protection of task contents and fails to achieve the verifiability of the aggregation results.

For the verifiability of aggregation results, Zhuo et al. [30] employed ring signatures to verify user identity and formalized verifiability as a computationally outsourced problem for result verification. Li et al. [31] proposed a publicly verifiable scheme enabling a public verifier to test an aggregation result. Guo et al. [32] realized authentication based on the password authenticated key exchange (PAKE) protocol, which can detect whether the data is uploaded and aggregated by legitimate. Zhang et al. [33] designed a verifiable aggregation scheme with key leakage resilience. After receiving user data, the fog node generates secondary verifiable encrypted data to ensure data integrity. They also proposed a verifiable privacy-preserving data aggregation scheme for smart grids [34]. They used the Boneh-Lynn-Shacham signature to generate an authenticator for each ciphertext. The control center can flexibly check encrypted data integrity without interaction with the aggregator gateway. Ding et al. [35] constructed an identity-based metering data aggregation scheme supporting batch verification by the collector. Despite their merits, these schemes are based on expensive operations such as bilinear pairing, leading to inefficient verification.

Therefore, we propose a new multi-task data aggregation scheme for healthcare. While protecting the privacy of health data and aggregation results, we ensure the confidentiality of task contents. In addition, we realize the verifiability of the aggregation results, mitigating the potential for unauthorized tampering by fog nodes and cloud servers.

3. PRELIMINARIES

3.1. Multi-task Data Aggregation

Our scheme considers the multi-task aggregation scenario, i.e., there are multiple task publishers in the system and the user needs to aggregate multiple different tasks in one round. Here, task publishers want their task contents to be known only by users. Users can retain the discretion to decide whether to accept tasks and which tasks to accept based on the task contents. This creates new privacy, i.e., task contents and user's decision privacy. Task contents can reflect the needs of the task publisher, and there is a potential loss of benefits if disclosed to competitors. The decision involves the user's preferences, privacy needs, medical conditions, etc. If it is leaked, an adversary can guess the user's health condition, which can violate personal privacy. Therefore, achieving confidentiality of task contents and user's decision is imperative.

Definition 1 (The Privacy of The User's Decision). *[29] Given $x \in \{0,1\}$ and a task T, if the user accepts the task then $x = 1$; otherwise $x = 0$. The adversary $\mathcal{A}$ wants to determine the user's decision, in other words, $\mathcal{A}$ wants to know the value of x. Suppose that the guess of $\mathcal{A}$ is denoted as $x' \in \{0,1\}$ if it is satisfied:*

$$pr[x' = x] \leq \frac{1}{2} + negl(),$$

where $negl()$ is a negligible function, then we say that the privacy of the user's decision is protected.

Definition 2 (The Privacy of The Task Contents). *If the task contents of task publishers are known only to themselves and users, cannot be accessed by other entities and adversaries. Then, we say that the privacy of task contents is guaranteed.*

3.2. Paillier Homomorphic Cryptosystem

The Paillier encryption [36] is a cryptosystem with additive homomorphism. It is described as the following three algorithms.

1) *KeyGen*: Choose two large prime numbers p, q. Let $n = p \cdot q$, and $\lambda = (p-1)(q-1)$. Let a function $L(u) = (u-1)/n$ and randomly choose a generator $g \in \mathbb{Z}^*_{n^2}$, which satisfies $\gcd(L(g^\lambda \bmod n^2), n) = 1$, where gcd() is the greatest common divisor function. Compute $\mu = (L(g^\lambda \bmod n^2))^{-1}$, and set public-private key pair $(pk = (n, g), sk = (\lambda, \mu))$.

2) *Enc*: For any plaintext $m \in \mathbb{Z}_n$, choose a random number $r \in \mathbb{Z}^*_n$ and compute the ciphertext $c = Enc(m) = g^m r^n \bmod n^2$.

3) *Dec*: Give a ciphertext $c \in \mathbb{Z}_{n^2}$, plaintext $m = Dec(c) = L(c^\lambda \bmod n^2)\mu \bmod n$.

Paillier cryptosystems satisfy additive homomorphisms such as $Dec(Enc(m_1) \cdot Enc(m_2)) = m_1 + m_2$, which is widely used in various privacy-preserving scenarios.

3.3. Pedersen Commitment

Pedersen commitment scheme [37] is a mechanism for ensuring data integrity and verifiability. It allows the promisor to prove the existence of a value to the receiver. Moreover, it satisfies perfectly hiding and computationally binding. Its construction is divided into three phases:

1) *Initialization*: Choose a multiplicative group $\mathbb{G}$ of order prime q. g, h are the generators of $\mathbb{G}$.

2) *Commitment*: Assume that the promisor has a message $m \in \mathbb{Z}_q$. He chooses a random number $r \in \mathbb{Z}_q$ as a

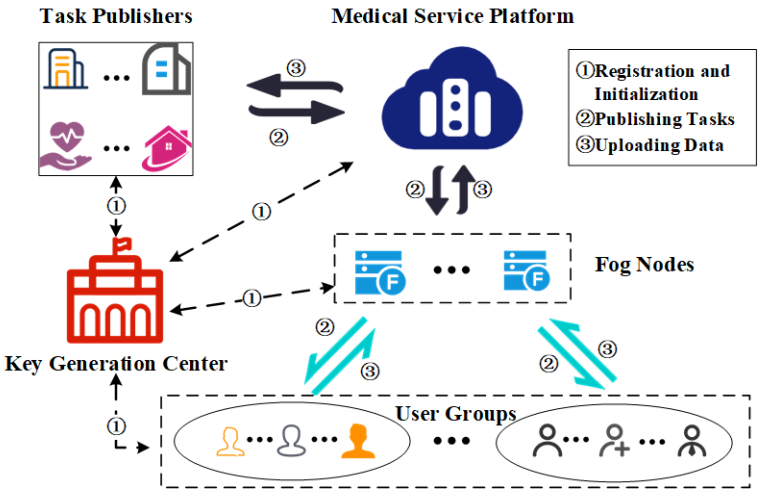


Figure 2: System model.

blind factor, calculates $com(m,r) = g^m h^r$, and sends it to the receiver.

3) *Verification*: The promisor transmits (m,r) to the receiver, who can verify the message by computing $g^m h^r \stackrel{?}{=} com(m,r)$.

In addition, Pedersen commitment scheme has homomorphism, such as $com(m_1,r_1) \cdot com(m_2,r_2) = com(m_1 + m_2, r_1 + r_2)$, for all $m_1, m_1, r_1, r_2 \in \mathbb{Z}_q$.

3.4. CRT-based Key Management Scheme

Min-Ho Park et al. [38] devised a key management scheme based on the Chinese remainder theorem (CRT). Assume that there are m pairs of large prime numbers $p_i, q_i, i \in [1,m]$. According to the RSA algorithm, we can obtain the public-private key $e_i, d_i, i \in [1,m]$, which satisfies $e_i \cdot d_i \equiv 1 \bmod \varphi(p_i q_i)$. Here, $\varphi()$ is Euler's totient function. If e_M and d_M are the master keys, used for the encryption and decryption, the following congruence equations are established:

$$P^{e_M} \equiv P^{e_i}(\mod p_i q_i), C^{d_M} \equiv C^{d_i}(\mod p_i q_i).$$

The sufficient condition for the above congruence equation are:

$$e_M \equiv e_i(\mod \varphi(p_i q_i)), d_M \equiv d_i(\mod \varphi(p_i q_i)).$$

If we let $x_i = (p_i - 1)/2, y_i = (q_i - 1)/2$, then the master key can be calculated as:

$$e_M \equiv \sum_{i=1}^{m} e_i M_i N_i(\mod 4x_1 y_1 x_2 y_2 \cdots x_m y_m),$$

where $M_i = \frac{x_1 y_1 x_2 y_2 \cdots x_m y_m}{x_i y_i}$, $M_i N_i \equiv 1(\mod 4x_i y_i)$.

An advantage of this scheme is that when one of the slave keys e_j is modified to e'_j, only the master key needs to be modified to $e'_M = e_M - e_j M_j N_j + e'_j M_j N_j$, while the other slave keys remain unchanged, which can reduce the cost of updating the group key due to the user joining or exiting.

4. MODELS AND DESIGN GOAL

4.1. System Model

As shown in Fig. 2, our system model consists of five parts: key generation center, task publishers, medical service platform, fog nodes, and users.

- *Key Generation Center* (KGC): It constitutes a trusted entity endowed with the responsibility of generating system parameters, initializing the system, ascertaining the identity of entities, and generating keys.
- *Task Publishers* (TPs): They represent official authorities or medical organizations seeking to analyze the health status of the populace or investigate specific disease characteristics. They generate a task T based on the requirements (e.g., what body data is needed, etc.) and send it to the medical service platform.
- *Medical Service Platform* (MSP): It constitutes a platform endowed with substantial storage and computational resources, responsible for sorting the tasks from TPs and publishing the task sequences to the fog nodes. When receiving the data uploaded by the fog nodes, it performs the final aggregation and sends the results to the TPs.
- *Fog Nodes* (FNs): Distributed in proximity to users, this infrastructure interfaces with multiple users, undertaking responsibilities that encompass the transmission of task sequences, collection of users' data, and uploading data to the MSP.
- *Users*: After receiving the task sequence, the user decrypts the task to obtain the task contents. Flexibility to accept the task according to their situation, collect data, and generate user's decision.

4.2. Threat Model

In our proposed scheme, for the internal part of the system, we consider KGC to be trusted since it is responsible

for the registration of the entities and generating system parameters and keys. The TPs, MSP, and FNs are honest but curious. That is, they will honestly follow the computations and steps specified in the protocol, but at the same time, they will try to record and analyze the information transmitted during the execution of the protocol to obtain additional private information related to other entities. Furthermore, we suppose the user is honest, will generate health data and decision vector correctly, and will not leak out the task contents. At the same time, we consider that there will be no collusion between the entities. It is reasonable since the different entities may have conflicts of interest, and they wish to maintain their reputations. For external threats, assuming an external adversary $\mathcal{A}$, can eavesdrop on communication and attempt to tamper with aggregated data to influence task results. Other active attacks, such as Dos, are out of the scope of this paper.

4.3. Design Goals

Our design goal is to propose a privacy-preserving data aggregation scheme for healthcare systems. Based on our system model and threat model, the following goals will be achieved:

- **Privacy of aggregated data:** Each user's raw health data can only be known by himself/herself and is not available to any other entity. The final aggregated results of the task can only be decrypted by its publisher and cannot be deciphered by any other entity.
- **Privacy of user's decision:** A user's personal choices regarding which tasks to choose can only be known to the user, and other entities are unaware of this decision.
- **Privacy of task contents:** The content of a task is explicitly known only to its publisher and users, while other entities can only know information such as the deadline.
- **Verifiability:** After receiving the final task results, the task publisher can verify that the results have not been tampered with by an adversary.
- **Efficiency:** The scheme should strive to achieve efficient computation and low communication overhead for situations where computational resources at the user's side are not sufficient.

5. OUR PROPOSED SCHEME

In this section, we will specifically describe our construction and detailed steps, which are organized into four phases: 1) system initialization; 2) task distribution and data collection; 3) data aggregation; and 4) decryption and verification. Table 1 lists the meanings of the symbols used in this paper.

Table 1
Notations and descriptions.

Notation	Description
p, q	Large prime numbers
$\mathbb{G}$	multiplicative group
g, h	The generator of $\mathbb{G}$
$Sign$	Signature algorithms such as Schnorr
H_1	Hash functions
ID, PID	The true identity, pseudonym of the entity
$E_{k_s}(*)$	AES encryption algorithm with key k_s
k_s	AES key generated by TP_s
$Enc(*)$, $Dec(*)$	The algorithm of Paillier encryption, decryption
D^*_{ij}, C^*_{ij}	Raw health data and decision vector for u_{ij}
D_{ij}, C_{ij}	Blinded health data and decision vector for u_{ij}
Com_{ij}	Pedersen Commitment of u_{ij}
D_{FN_i}, C_{FN_i}	The aggregation results of FN_i
D_{sum}, C_{sum}	The aggregation results of MSP
n	Number of users managed by each FNs
m, s	Number of FNs, TPs

5.1. Overview

An overview of our proposed scheme is presented in Fig. 3. The system initialization phase is initiated by the KGC, which generates system parameters and keys and conducts authentication and registration of entities. During the task distribution and data collection phase, each task publisher generates the task and encrypts the task contents using a symmetric encryption algorithm. Then, the task publisher employs the CRT-based key management scheme to securely distribute the symmetric key to users. Upon receiving task requests, the MSP sorts and distributes them to the FNs, which are further distributed to the users. Following the reception of the task sequence, the user can compute the symmetric key and decrypt the task to obtain the task contents. After this, the user can generate the decision vector and health data based on his/her circumstances. Finally, the user blinds the data, computes the commitment and signature, and uploads these messages to the FNs. In the data aggregation phases, the FNs aggregate the data of the users, and the MSP aggregates the data of the FNs. Eventually, after the task publisher receives the final aggregation results, it can obtain the task results and the number of participants according to the decryption rules.

5.2. System Initialization

System initialization is performed by KGC, which generates system parameters and provides registration services for entities. Here, for descriptive convenience, we assume the system has v task publishers, m fog nodes, and each fog node connecting n users, such as TPs= $\{TP_1, TP_2, \cdots, TP_v\}$, FNs = $\{FN_1, FN_2, \cdots, FN_m\}$, and $U_{FN_i} = \{u_{ij} | i \in [1, m], j \in [1, n]\}$.

5.2.1. Parameter generation

First, KGC selects m pairs of secure large prime numbers $p_i, q_i, i \in [1, m]$, generates the corresponding public-private key pairs e_i, d_i, and calculates the master key e_M as described in Section 3.4. Second, KGC additionally chooses two secure large primes p, q, a multiplicative group $\mathbb{G}$ of order q whose generators are g, h. Then, the KGC generates the Paillier public-private key pairs ($pk_{Enc} = (n, g)$, $sk_{Dec} =$

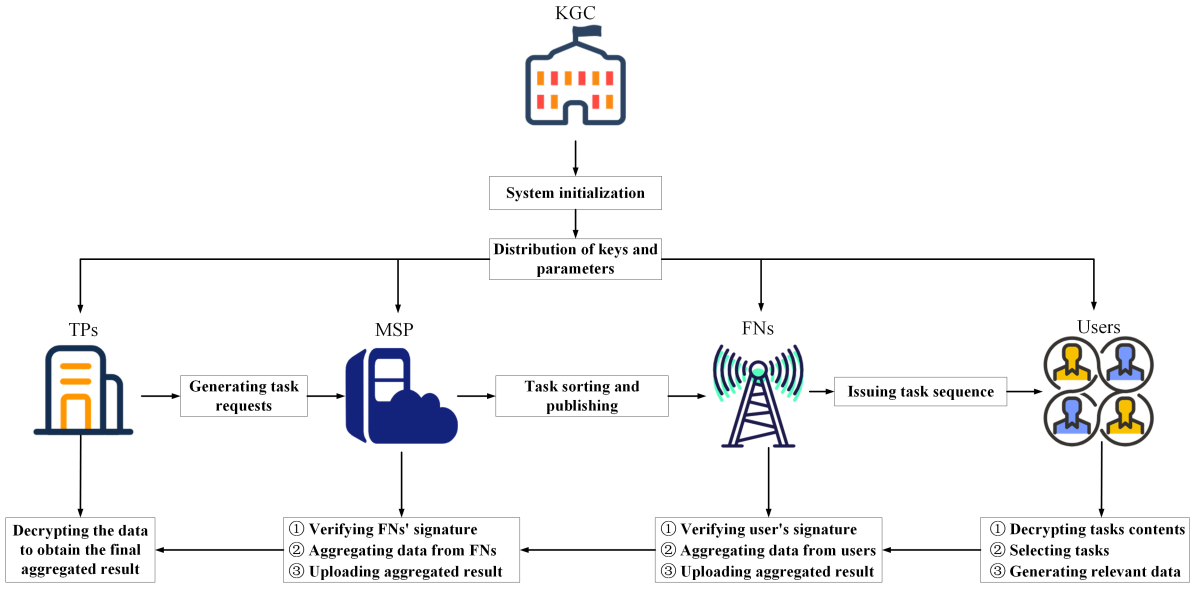


Figure 3: Overview of our scheme.

(λ, μ)), chooses a master private key $k_{mas} \in \mathbb{Z}_q^*$, a symmetric encryption algorithm (e.g., AES), a signature algorithm (e.g., Schnorr signature), and a secure one-way hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. Finally, KGC announces the system parameters $\{g, h, \mathbb{G}, H_1, pk_{Enc}, AES, Sign\}$.

5.2.2. *Entity registration*

The entities in the system are TPs, MSP, FNs, and users. They have to register with KGC using their respective unique identifier ID. Upon receiving the ID of the entities, KGC first verifies their real identities. Upon successful authentication, the KGC calculates their pseudonym $PID = k_{mas} \cdot H_1(ID)$ and generates the public-private key pairs of Schnorr signature, such as (sk_{MSP}, pk_{MSP}), (sk_{FN_i}, pk_{FN_i}), (sk_{TP_s}, pk_{TP_s}), and $(sk_{u_{ij}}, pk_{u_{ij}})$. Once the the entities receive the PID and signature public-private key pairs, they publicize the PID and signature public key. Then, the KGC transmits the private key sk_{Dec} of Paillier cryptographic algorithm, the master key e_M, and $N = p_1 q_1 p_2 q_2 \cdots p_m q_m$ to the successfully registered TPs individually through a secure channel. In addition, the KGC also needs to send the slave key $(e_i, d_i, p_i q_i)$ to the u_{ij}.

5.3. Task Distribution and Data Collection

5.3.1. *Task distribution*

In this phase, the task publisher can generate task requests and publish them to users. The execution steps for the task publisher are as follows.

- **Step 1:** The task publisher $TP_s, s \in [1, \upsilon]$ generates the task content T_s (e.g., physiological data needed) and task requirements Rq_s (e.g., data types, deadlines) according to his requirements.
- **Step 2:** TP_s chooses the key k_s of the AES algorithm and compute the ciphertext $C_s = {k_s}^{e_M} \bmod N$ and $E_{k_s}(T_s)$. Subsequently, TP_s constructs the task request $Q_s = PID_{TP_s} || C_s || E_{k_s}(T_s) || Rq_s$, where $||$, PID_{TP_s}, C_s, and $E_{k_s}(T_s)$ are the concatenation operator, the pseudonym of TP_s, the ciphertext of the key k_s, and the ciphertext of the task content T_s, respectively.
- **Step 3:** TP_s signs the task request, e.g., $Sign(Q_s)$, to confirm identity and ensure the integrity of the information. Then, TP_s sends $Q_s || Sign(Q_s) || T_{time}$ to the MSP, where T_{time} is the timestamp.

When the MSP receives the message from TP_s, it verifies the signature $Sign(Q_s)$, and after successful verification, it can receive Q_s; otherwise, it discards the message. Assume that the MSP can obtain a set of task requests $Q = \{Q_s | s \in [1, \upsilon]\}$. Then, the MSP can categorize and sort the task requests based on Rq_s, e.g., prioritize tasks with imminent deadlines. Note that the appropriate sorting method can be utilized here to improve efficiency. Since it is not part of our research, we do not expand on it. Assume that the sequence of tasks after MSP ordering is $(Q_1, Q_2, \cdots, Q_s)$. Eventually, the MSP distributes the task sequence to the FNs. FNs transmits it to users.

5.3.2. *Data collection*

In this phase, the user needs to decrypt the task contents, select the task, and generate health data and other information as required. When user u_{ij} receives $(Q_1, Q_2, \cdots, Q_s)$, he can extract $(C_1, C_2, \cdots, C_s)$ and obtain the key k_s through Eq. (1).

$$\begin{aligned} &(C_s)^{d_i} \bmod p_i q_i \\ &= ((k_s)^{e_M})^{d_i}) \bmod p_i q_i \\ &= ((k_s)^{1 \bmod \varphi(p_i q_i)}) \bmod p_i q_i \\ &= k_s \end{aligned} \tag{1}$$

After getting the key k_s, u_{ij} can execute the AES decryption algorithm to obtain the task contents T_s. Subsequently, u_{ij} can be tasked to generate health data and decision vector based on the task requirements and his situation.

We denote the health data of u_{ij} as $D^*_{ij} = (d^*_{1,ij}, d^*_{2,ij}, \cdots, d^*_{s,ij})$, and $d^*_{s,ij} = 0$ if and only if u_{ij} does not accept the task Q_s. Since s is a variable, which makes the length of the user's health data and decision vector are not fixed. Therefore, our scheme uses random value blinding to mask these data instead of using Paillier encryption to ensure the confidentiality of the data. u_{ij} chooses a random number $r_{ij,1}$ that masks $d^*_{s,ij}$ by computing $d_{s,ij} = d^*_{s,ij} + r_{ij,1} + k_s$. Then, u_{ij} generates a decision vector $C^*_{ij} = (c^*_{1,ij}, c^*_{2,ij}, \cdots, c^*_{s,ij})$. Here, if u_{ij} accepts the task Q_s, $c^*_{s,ij} = 1$; otherwise, $c^*_{s,ij} = 0$. u_{ij} chooses another random number $r_{ij,2}$ to generate the masking decision vector, such that $c_{s,ij} = c^*_{s,ij} + r_{ij,2} + k_s$. Here, $r_{ij,1}$ and $r_{ij,2}$ are sufficiently uniform random integers independently chosen by the user, which can be generated using the pseudo random number generator.

To prevent an adversary from tampering with the data, u_{ij} computes the commitment $Com_{ij} = g^{D_{ij}} h^{r_{ij,1}}$, where $D_{ij} = (d_{1,ij}, d_{2,ij}, \cdots, d_{s,ij})$. Ultimately, u_{ij} sends $(PID_{u_{ij}} || C_{ij} || D_{ij} || Enc(r_{ij,1}) || Enc(r_{ij,2}) || Com_{ij})$ to FN_i, where $Enc(r_{ij,1})$ and $Enc(r_{ij,2})$ are the Paillier ciphertext of $r_{ij,1}$ and $r_{ij,2}$, respectively. Note that the calculation and verification of the signature is essential during data transfer. The remainder of this paper will default to the entity that will compute and verify the signature.

5.4. Data Aggregation

In this phase, data aggregation is performed by FNs and MSP.

5.4.1. FNs Aggregating

When FN_i receives the message from u_{ij}, it first checks whether the $PID_{u_{ij}}$ is legitimate and verifies the identity of the u_{ij}. Then, FN_i stores all users who have passed the authentication in a new set, denoted as $U_i = \{u_{i1}, u_{i2}, \cdots\}$. We use $|U_i|$ to represent the number of elements in the set U_i. Subsequently, FN_i aggregates the data of U_i as follows.

$$\begin{cases} D_{FN_i} = \sum_{j=1}^{|U_i|} D_{ij} \\ C_{FN_i} = \sum_{j=1}^{|U_i|} C_{ij} \\ Com_{FN_i} = \prod_{j=1}^{|U_i|} Com_{ij} \\ Enc_{FN_{i,1}} = \prod_{j=1}^{|U_i|} Enc(r_{ij,1}) \\ Enc_{FN_{i,2}} = \prod_{j=1}^{|U_i|} Enc(r_{ij,2}) \end{cases} \tag{2}$$

Then, FN_i transmits $(PID_{FN_i}, D_{FN_i}, C_{FN_i}, Enc_{FN_{i,1}}, Enc_{FN_{i,2}}, Com_{FN_i}, |U_i|)$ to the MSP.

5.4.2. MSP Aggregating

After receiving the data from all FNs, the MSP first counts the number of all participating, e.g., $U_{sum} = \sum_{i=1}^{m} |U_i|$. Then, the MSP performs the final data aggregation, e.g.,

$$\begin{cases} D_{sum} = \sum_{i=1}^{m} D_{FN_i} \\ C_{sum} = \sum_{i=1}^{m} C_{FN_i} \\ Com_{sum} = \prod_{i=1}^{m} Com_{FN_i} \\ Enc_{sum,1} = \prod_{i=1}^{m} Enc_{FN_{i,1}} \\ Enc_{sum,2} = \prod_{i=1}^{m} Enc_{FN_{i,2}} \end{cases} \tag{3}$$

At this point, we can learn that the aggregation results of Q_s is

$$\begin{aligned} D_{sum,s} &= \sum_{U_{sum}} D_{s,ij} \\ &= \sum_{U_{sum}} (d^*_{s,ij} + r_{ij,1} + k_s) \\ &= \sum_{U_{sum}} d^*_{s,ij} + \sum_{U_{sum}} r_{ij,1} + U_{sum} \cdot k_s. \end{aligned}$$

The final decision vector of Q_s is

$$\begin{aligned} C_{sum,s} &= \sum_{U_{sum}} C_{s,ij} \\ &= \sum_{U_{sum}} (c^*_{s,ij} + r_{ij,2} + k_s) \\ &= \sum_{U_{sum}} c^*_{s,ij} + \sum_{U_{sum}} r_{ij,2} + U_{sum} \cdot k_s. \end{aligned}$$

Eventually, the MSP transmits the task aggregation result $(D_{sum}, C_{sum}, Enc_{sum,1}, Enc_{sum,2}, Com_{sum}, U_{sum})$ to the TPs.

5.5. Decryption and Verification

After receiving the aggregated results from the MSP, TP_s can calculate R_1 and R_2 according to Eq. (4).

$$\begin{cases} R_1 = Dec(Enc_{sum,1}) = \sum_{j=1}^{U_{sum}} r_{ij,1} \\ R_2 = Dec(Enc_{sum,2}) = \sum_{j=1}^{U_{sum}} r_{ij,2} \end{cases} \tag{4}$$

Then, the TP_s can validate the received aggregation results to prevent the adversary from tampering with the data according to Eq. (5).

$$Com_{sum} \stackrel{?}{=} g^{D_{sum}} h^{R_1} \tag{5}$$

After that, TP_s can obtain the task results and the number of users participating in this task according to the following

equation.

$$D_{agg,s} = \sum_{U_{sum}} D_{s,ij} - R_1 - U_{sum} \cdot k_s = \sum d^*_{s,ij}$$
$$C_{agg,s} = \sum_{U_{sum}} C_{s,ij} - R_2 - U_{sum} \cdot k_s = \sum c^*_{s,ij}$$

6. SECURITY ANALYSIS

In this section, we analyze the security of the scheme in detail and give proofs.

Theorem 1. *Under our threat model, the scheme enables the privacy of the user's decision.*

Proof 1. The decision vector $C^*_{ij} = (c^*_{1,ij}, c^*_{2,ij}, \cdots, c^*_{s,ij})$ of user u_{ij} is masked by random number $r_{ij,2}$ and the symmetric key k_s, i.e.,

$$\begin{cases} c_{s,ij} = 1 + r_{ij,2} + k_s, & \text{if } u_{ij} \text{ accept } Q_s; \\ c_{s,ij} = 0 + r_{ij,2} + k_s, & \text{otherwise.} \end{cases}$$

Adversary $\mathcal{A}$ can obtain $(PID_{u_{ij}}||C_{ij}||D_{ij}||Enc(r_{ij,1})||Enc(r_{ij,2})||Com_{ij})$ by eavesdropping. Since k_s is distributed to the user through the CRT-based key management scheme and $r_{ij,2}$ is a random number chosen by the user. Therefore, $\mathcal{A}$ is unable to obtain k_s and $r_{ij,2}$.

In the absence of $r_{ij,2}$ and k_s, adversary $\mathcal{A}$ cannot distinguish between $1 + r_{ij,2} + k_s$ and $0 + r_{ij,2} + k_s$, which means that $\mathcal{A}$ can only guess whether u_{ij} receives task Q_s. According to Definition 1, for $x' \in \{0, 1\}$ and $x \in \{0, 1\}$, there are only four possible scenarios, $x' = 0 \wedge x = 0$, $x' = 0 \wedge x = 1$, $x' = 1 \wedge x = 0$, and $x' = 1 \wedge x = 1$. Then, the probability that $\mathcal{A}$ guesses correctly is shown below.

$$\begin{aligned} &pr[x' = x] \\ &= pr[\{x' = 1 \wedge x = 1\} \vee \{x' = 0 \wedge x = 0\}] \\ &= pr[x' = 1 \wedge x = 1] + pr[x' = 0 \wedge x = 0] \\ &= \tfrac{1}{4} + \tfrac{1}{4} = \tfrac{1}{2} \end{aligned}$$

Therefore, the adversary $\mathcal{A}$ cannot know the user's decision. Similarly, entities within the system also cannot obtain a single user's decision. The privacy of the user's decision is guaranteed.

Theorem 2. *Under our threat model, the scheme enables the privacy of the task contents.*

Proof 2. After the task publisher TP_s generates the task contents T_s, he encrypts it using the AES key k_s. Then, TP_s distributes the key k_s to users through the CRT-based key management scheme. From the security of the AES algorithm, it is clear that no entity or adversary can decrypt the AES ciphertext to get the task contents T_s without the key k_s. To get the key k_s, the adversary must decrypt the ciphertext $C_s = {k_s}^{e_M} \bmod N$. The security of the CRT-based key management scheme has been proved in reference [38]. Therefore, no entity or adversary can obtain the key k_s without the slave key $(e_i, d_i, p_i q_i)$.

According to the design of our scheme, the slave key is generated by a trusted KGC and distributed through a secure channel when users register. And the users are honest, they will not disclose the slave key. Entities within the system and external adversaries cannot access the slave key to decrypt the ciphertext C_s. Therefore, they cannot decrypt the AES ciphertext to obtain the task content T_s. The privacy of the task contents is guaranteed.

Theorem 3. *Under our threat model, the scheme enables the privacy of the aggregated data.*

Proof 3. The raw health data $D^*_{ij} = (d^*_{1,ij}, d^*_{2,ij}, \cdots, d^*_{s,ij})$ of user u_{ij} is masked by random number $r_{ij,1}$ and a symmetric key k_s, i.e.,

$$d_{s,ij} = d^*_{s,ij} + r_{ij,1} + k_s, \text{for } s = 1, 2, \cdots, v;$$

Without $r_{ij,1}$ and k_s, adversary $\mathcal{A}$ and entities within the system cannot capture the raw data of the user. The privacy of the user's raw aggregated data is guaranteed. During the aggregation process, $r_{ij,1}$ is encrypted into Paillier ciphertext. The security of the Paillier encryption ensures that the adversary cannot decrypt the ciphertext without the key, which makes it impossible for adversary $\mathcal{A}$ and FNs to obtain intermediate aggregation results. Therefore, the privacy of data during the aggregation process is guaranteed. Similarly, adversary $\mathcal{A}$ and MSP cannot access the final aggregation task results.

The other task publisher $TP_j (j \neq s)$ can obtain R_1 and R_2 by Paillier decryption algorithm. Then, TP_j can compute

$$\sum_{U_{sum}} D_{s,ij} - R_1 = \sum d^*_{s,ij} + U_{sum} \cdot k_s$$
$$\sum_{U_{sum}} C_{s,ij} - R_2 = \sum c^*_{s,ij} + U_{sum} \cdot k_s$$

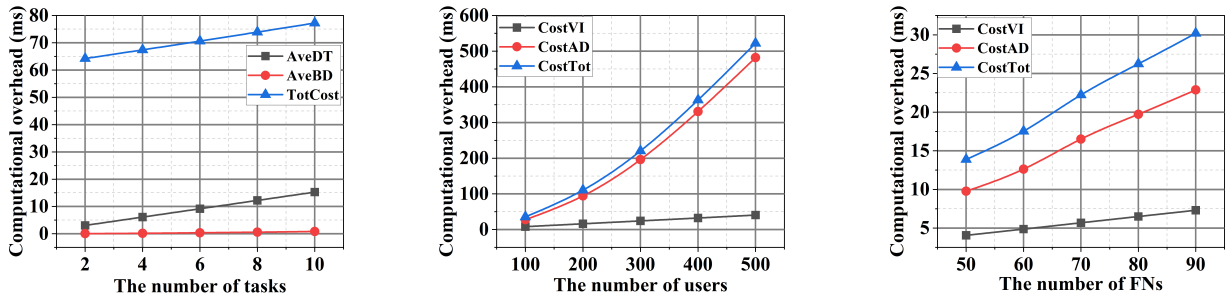
However, k_s is kept secret by TP_s and the users. Therefore, by the security of the CRT-based key management scheme, TP_j cannot obtain any information about k_s. The privacy of the aggregation results is guaranteed. To summarize, our scheme protects the privacy of aggregated data.

Theorem 4. *If the Pedersen commitment scheme is perfectly hiding and computationally binding, adversary $\mathcal{A}$ cannot tamper with the data and escape commitment verification.*

Proof 4. To ensure the verifiability of the task results, user u_{ij} computes Pedersen commitment $Com_{ij} = g^{D_{ij}} h^{r_{ij,1}}$ after generating D_{ij}. Assume that adversary $\mathcal{A}$ tampers the data D_{ij} of u_{ij} to D'_{ij}. For $\mathcal{A}$ to pass the commitment verification, he must successfully compute

$$Com'_{ij} = g^{D'_{ij}} h^{r_{ij,1}}.$$

However, $r_{ij,1}$ is kept secret by u_{ij} and is not known to other entities. Therefore, adversary $\mathcal{A}$ cannot successfully compute Com'_{ij}. Assuming that adversary $\mathcal{A}$ tampers with



(a) Computational cost changes with υ. (b) Computational cost changes with n. (c) Computational cost changes with m.

Figure 4: The variation of computation overhead with the number of tasks, users, and FNs.

data D_{ij} and submits the original commitment Com_{ij} to the system. If the commitment Com_{ij} passes the commitment validation of the task publisher. This means that there are two different messages D_{ij} and D'_{ij} with the same commitment value, which violates the computationally binding of the Pedersen commitment. Therefore, adversary $\mathcal{A}$ cannot tamper with the user data and successfully pass the commitment validation. Similarly, adversary $\mathcal{A}$ cannot successfully tamper with the aggregation results of FNs and MSP.

Based on the homomorphic nature of Pedersen commitment, it is known that TPs who possess R_1 and D_{sum} can easily verify the commitment values of the aggregated results. Therefore, our scheme prevents the adversary from tampering with the data during the aggregation process and achieves verifiability of the aggregation results.

7. PERFORMANCE EVALUATION

In this section, we evaluate the proposed scheme in terms of theory and concrete experiments. Subsequently, we compare our scheme with the schemes [24] and [34].

7.1. Theoretical Analysis

We regard the establishment of the system as a preprocessing operation. Therefore, we concentrate on the phase after the initialization of the system. First, the user can receive a sequence of tasks $(Q_1, Q_2, \cdots, Q_s)$ sorted by the MSP. After this, the user needs to perform s exponentiation operations to obtain the AES key and perform s AES decryption algorithms to retrieve the task contents. After obtaining the task contents, the user can choose the task and generate the corresponding health data and decision vector. To achieve data privacy, the user has to mask the raw data before uploading it to the FNs. The user has to execute four addition operations, two Paillier encryption operation, one commitment computation, and one signature computation. Therefore, the user's computational overhead is $s(T_{ex} + T_{DeAES}) + 4T_{Add} + 2T_{Pa} + T_{Co} + T_{Si}$, where T_{ex}, T_{DeAES}, T_{Add}, T_{Pa}, T_{Co}, and T_{Si} denote the overhead of an exponentiation operation, an AES decryption operation, an addition operation, a Paillier encryption operation, a commitment operation, and a signature operation, respectively.

The FNs authenticate the user's identity and aggregate the user's data, which requires performing n signature verifications, $3n$ addition operations, and $2n$ multiplication operations. Therefore, the computational overhead of FNs is $(T_{Vesi} + 3T_{Add} + 2T_{Mul})n$, where T_{Vesi} and T_{Mul} denote the overhead of a signature verification operation and a multiplication operation, respectively. Similarly, the computational overhead of the MSP is approximately $(T_{Vesi} + 3T_{Add} + 2T_{Mul})m$, where m is the number of FNs.

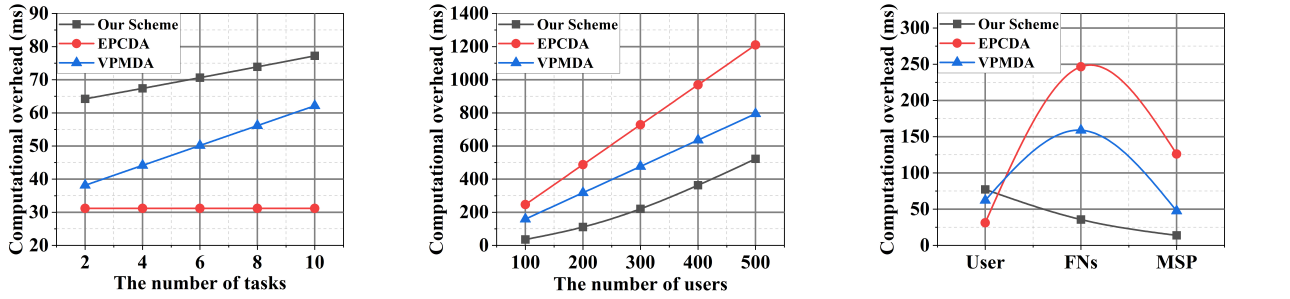
The TPs needs to generate the task requests, encrypt the task contents, and share the encryption key with the user through the key management scheme. TPs requires one AES encryption operation, one exponentiation operation, and one signature computation. After receiving the results returned by the MSP, the TPs must verify the commitments and decrypt the ciphertexts to obtain the task aggregation results. He needs to perform one commitment computation, two Paillier decryption operations, two multiplication operations, and four subtraction operations. Therefore, the computational overhead is $T_{AES} + T_{ex} + T_{Si} + 2T_{DePa} + 2T_{Mul} + 4T_{sub}$, where T_{AES}, T_{DePa} and T_{sub} denote the overhead of AES encryption algorithm, Paillier decryption operation and subtraction operation, respectively.

7.2. Experimental Evaluation

We implement our scheme based on Python programming language using cryptography library, hashlib library, etc. The hash function we chose is SHA-256. We utilize the elliptic curve SECP256K1 to construct the Pedersen commitment scheme. We deploy the code on the virtual machine Ubuntu 22.04.1 in VMware® Workstation 17 Pro (with 8G RAM and Intel® Core™ i5-8300H CPU @ 2.30GHz × 8).

7.2.1. Computational overhead

We assume that the number of users n varies from 100 to 500 with an interval of 100, the number of FNs m varies from 50 to 90 with an interval of 10, and the number of tasks υ in a round of aggregation varies from 2 to 10 with an interval of 2. We set the lengths of the user's health data, the task contents, the key of the AES, and the prime number of the Paillier cryptosystem to 128 bits, 256 bits, 256 bits, and 1024 bits, respectively.



(a) Computational cost changes with υ.

(b) Computational cost changes with n.

(c) The computational cost of user, FNs, and MSP. keep υ=10, n=100, and m=50.

Figure 5: Comparison of the computational overhead of the schemes.

The computational overhead of the task publisher TPs is essentially fixed. They encrypt the task contents, generate the task request, and distribute the key of AES using the key management scheme, which has an overhead of about 0.2 ms for this whole process. After receiving the aggregation results, TPs need to compute the commitment to determine whether the data has been tampered with and decrypt the task aggregation results. The total overhead of this process is about 75 ms. We provide the variation of the average time for the user to decrypt the task (AveDT), the average time for blinding the data (AveBD), and the total computational cost for each user (TotCost) in Fig. 4(a). To conveniently demonstrate the overhead of FNs and MSP, we set the number of tasks υ = 10. Specific experimental results are shown in Fig. 4(b) and Fig. 4(c), including the cost of verifying identity (CostVI), the cost of aggregating data (CostAD), and the total cost (CostTot). Please note that batch processing can be employed during the verification of signatures to further minimize overhead and optimize performance.

7.2.2. Communication overhead

Based on the experimental setup, we can know the length of the pseudonym, promise, Paillier ciphertext, and signature are 256 bits, 512 bits, 2048 bits, and 512 bits, respectively. We commence the assessment of communication overhead from the data collection phase. Since the length of the health data is set to 128 bits, the communication overhead of each entity is changed according to the number of tasks. If we set the number of tasks υ = 10, the number of users n = 100, and the number of FNs m = 50, then the communication overhead of the users, FNs, and MSP is approximately 1312 B, 108 KB, and 53 KB, respectively.

7.3. Comparison with Other Schemes

We compare our scheme with schemes [24] (denoted as EPCDA) and [34] (denoted as VPMDA). These two schemes are devised for the smart grid whose system framework is quite similar to our scheme. Table 2 lists the computation times for several basic operations.

Table 2
Calculation cost of basic cryptographic operations.

Symbol	Meaning	Times (ms)
T_{AES}	Encryption time of one AES algorithm	0.124055
T_{DAES}	Decryption time of one AES algorithm	0.043410
T_{com}	Time of commitment	1.028663
T_{sign}	Time of signature	0.049974
T_{Dsign}	Time of signature verification	0.082277
T_{ex}	Time of exponentiation in comparison scheme	0.771084
T_{bp}	Time of a bilinear pairing in comparison scheme	2.409638

7.3.1. Comparison of computational overhead

As FNs and MSP jointly bear the responsibility of aggregating data, the factors influencing their computational overhead exhibit notable similarities. Therefore, we concentrate our comparison on the user and FNs side. As shown in Fig. 5(a), comparatively, our scheme incurs a marginally higher computational overhead on the user side. This is because we protect the privacy of the task contents and achieve verifiability of the aggregation results with some more computational steps than the compared schemes. Turning to the FNs side, Fig. 5(b) illustrates the computational overhead for υ = 10, n = 100, 200, 300, 400, and 500. It can be seen that the computational overhead of FNs exhibits a roughly linear correlation with the number of users. Analysis of experimental results substantiates the superiority of our proposed scheme. For further demonstration, we set υ=10, n=100, m=50 and show the computational overhead of user, FNs, and MSP for each scheme in Fig. 5(c). In EPCDA, the user side needs to first compress the collected data. Next, the compressed data is encrypted with Paillier encryption. Finally, the user generates the corresponding signature. In VPMDA, the user needs to pack the data using a superincreasing sequence and then compute the Paillier ciphertext and signature. In contrast, our scheme not only needs to encrypt the data, compute the Paillier ciphertext and signature, but also compute the AES key, decrypt the task contents and compute the commitment, which makes the user-side overhead of this scheme slightly higher. At the FNs and MSP side, the aggregation of our scheme mainly performs the addition operation, while EPCDA and

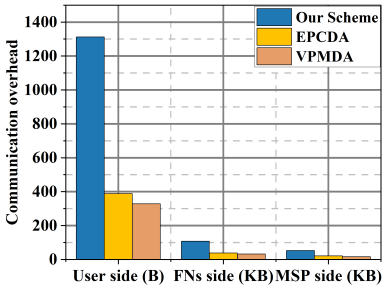


Figure 6: Comparison of communication overhead. Keep υ=10, n = 100, and m=50.

VPMDA contain extensive multiplication and pairing operations. As a result, the computational overhead at the FNs and MSP side is lower than the comparison schemes.

7.3.2. Comparison of communication overhead

Next, we proceed to analyze and compare the communication overheads. In our scheme, users transmit pseudonyms, masked user data, Paillier ciphertexts, Pedersen commitment, and signatures to FNs. Therefore, the total communication overhead is 7936 bits. The user side of EPCDA needs to send Paillier ciphertexts, identifiers, and signatures with a communication overhead of 3104 bits. The user side of VPMDA needs to send Paillier ciphertexts, signatures, and timestamps with a communication overhead of 2624 bits.

At the FNs side, each scheme needs to receive data from n users and aggregate the data into new data to deliver to the MSP. The communication overhead of our scheme, EPCDA, and VPMDA are approximately $7936n + 8064$ bits, $3104n + 3104$ bits, and $2624n + 2880$ bits, respectively. Similarly, the communication overhead of the MSP is about $8064m + 7880$ bit, $3104m + 3104$ bits, and $2880m$ bits, respectively. When we set n = 100 and m = 50, the communication overhead between the schemes is shown in Fig. 6. Our scheme has a high communication overhead on each entity's side. The main factors are: 1) our scheme requires not only users' data but also their decision vectors; 2) the encryption of the data is in the form of a masking operation, which is not a fixed value like the Paillier cipher. 3) our scheme requires transmission commitment values to guarantee the verifiability of the aggregation results.

From the experimental results, it is clear that our scheme has less overhead on the aggregator side. While experiencing a marginal increase in overhead on the user side, this expense is desirable and stems from our paramount emphasis on the security of our scheme. Particularly concerning task publishers, our approach meticulously safeguards the privacy of task aggregation results and ensures the confidentiality of task content and the verifiability of task outcomes. Sacrificing some performance for improved security is inevitable. Considering the nuanced interplay between security and efficiency, we are convinced that our scheme is secure, practicable, and efficient.

8. CONCLUSION

In this paper, we propose an efficient and privacy-preserving data aggregation scheme for healthcare. It can be employed as a versatile scheme applicable to many scenarios within the domain of mobile crowdsensing. This scheme efficiently aggregates multiple tasks while protecting the privacy of users and task publishers. We use a CRT-based key management scheme to distribute encryption keys and achieve protection of task contents. In addition, we ensure data integrity using Schnorr signature and achieve verifiability of task aggregation results using Pedersen commitment. Finally, theoretical analysis and experimental evaluation demonstrate that our scheme is secure and reliable.

In the future, our research will explore two prospective directions. For intricate application environments, we contemplate the development of a multi-functional data aggregation scheme, encompassing features such as variance, maxima, and other relevant metrics. Moreover, to incentivize active user participation in aggregation tasks, we will investigate the integration of data aggregation with incentive mechanisms, formulating a comprehensive aggregation scheme incorporating both rewards and penalties.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62472266, 62472265, 62071280, 62302280, and 62172258, in part by the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, under Grant 2023ZD021.

References

[1] C. Luo, X. Liu, W. Xue, Y. Shen, J. Li, W. Hu, A. X. Liu, Predictable privacy-preserving mobile crowd sensing: A tale of two roles, IEEE/ACM Transactions on Networking 27 (1) (2019) 361–374.

[2] C. Wang, L. Han, G. Stein, S. Day, C. Bien-Gund, A. Mathews, J. J. Ong, P.-Z. Zhao, S.-F. Wei, J. Walker, et al., Crowdsourcing in health and medical research: a systematic review, Infectious diseases of poverty 9 (2020) 1–9.

[3] J. Li, T. Wang, B. Yang, Q. Yang, W. Zhang, K. Hong, Abcrowdmed: A fine-grained worker selection scheme for crowdsourcing healthcare with privacy-preserving, IEEE Transactions on Services Computing (2023).

[4] W. H. Organization, UNICEF, et al., Crowdsourcing in health and health research: a practical guide, Tech. rep., World Health Organization (2018).

[5] J. Ni, K. Zhang, X. Lin, X. Shen, Securing fog computing for internet of things applications: Challenges and solutions, IEEE Communications Surveys & Tutorials 20 (1) (2017) 601–628.

[6] C. Guo, P. Tian, K.-K. R. Choo, Enabling privacy-assured fog-based data aggregation in e-healthcare systems, IEEE Transactions on Industrial Informatics 17 (3) (2020) 1948–1957.

[7] Z. Ahmadi, M. Haghi Kashani, M. Nikravan, E. Mahdipour, Fog-based healthcare systems: A systematic review, Multimedia Tools and Applications 80 (2021) 36361–36400.

[8] L. S. Hughes, R. L. Phillips, J. E. DeVoe, A. W. Bazemore, Community vital signs: taking the pulse of the community while caring for patients, The Journal of the American Board of Family Medicine 29 (3) (2016) 419–422.

[9] N. Nasser, Q. Emad-ul Haq, M. Imran, A. Ali, I. Razzak, A. Al-Helali, A smart healthcare framework for detection and monitoring of covid-19 using iot and cloud computing, Neural Computing and Applications (2021) 1–15.

[10] L. J. Laffin, H. W. Kaufman, Z. Chen, J. K. Niles, A. R. Arellano, L. A. Bare, S. L. Hazen, Rise in blood pressure observed among us adults during the covid-19 pandemic, Circulation 145 (3) (2022) 235–237.

[11] B. O. Soufiene, A. A. Bahattab, A. Trad, H. Youssef, Lsda: lightweight secure data aggregation scheme in healthcare using iot, in: Proceedings of the 10th International Conference on Information Systems and Technologies, 2020, pp. 1–4.

[12] X.-D. Wang, W.-Z. Meng, Y.-N. Liu, Lightweight privacy-preserving data aggregation protocol against internal attacks in smart grid, Journal of Information Security and Applications 55 (2020) 102628.

[13] F. A. Almalki, B. O. Soufiene, Eppda: an efficient and privacy-preserving data aggregation scheme with authentication and authorization for iot-based healthcare applications, Wireless Communications and Mobile Computing 2021 (2021) 1–18.

[14] W. Zhang, S. Liu, Z. Xia, A distributed privacy-preserving data aggregation scheme for smart grid with fine-grained access control, Journal of Information Security and Applications 66 (2022) 103118.

[15] H. Liu, T. Gu, M. Shojafar, M. Alazab, Y. Liu, Opera: Optional dimensional privacy-preserving data aggregation for smart healthcare systems, IEEE Transactions on Industrial Informatics 19 (1) (2022) 857–866.

[16] S. B. Othman, F. A. Almalki, C. Chakraborty, H. Sakli, Privacy-preserving aware data aggregation for iot-based healthcare with green computing technologies, Computers and Electrical Engineering 101 (2022) 108025.

[17] X. Yan, B. Zeng, X. Zhang, Privacy-preserving and customization-supported data aggregation in mobile crowdsensing, IEEE Internet of Things Journal 9 (20) (2022) 19868–19880.

[18] S. Han, S. Zhao, Q. Li, C.-H. Ju, W. Zhou, Ppm-hda: Privacy-preserving and multifunctional health data aggregation with fault tolerance, IEEE Transactions on Information Forensics and Security 11 (9) (2016) 1940–1955.

[19] Y. Liu, W. Guo, C.-I. Fan, L. Chang, C. Cheng, A practical privacy-preserving data aggregation (3pda) scheme for smart grid, IEEE Transactions on Industrial Informatics 15 (3) (2018) 1767–1774.

[20] H. Wu, L. Wang, G. Xue, Privacy-aware task allocation and data aggregation in fog-assisted spatial crowdsourcing, IEEE Transactions on Network Science and Engineering 7 (1) (2019) 589–602.

[21] Y. Su, Y. Li, J. Li, K. Zhang, Lceda: Lightweight and communication-efficient data aggregation scheme for smart grid, IEEE Internet of Things Journal 8 (20) (2021) 15639–15648.

[22] S. Li, K. Xue, Q. Yang, P. Hong, Ppma: Privacy-preserving multisubset data aggregation in smart grid, IEEE Transactions on Industrial Informatics 14 (2) (2017) 462–471.

[23] X. Zuo, L. Li, H. Peng, S. Luo, Y. Yang, Privacy-preserving multidimensional data aggregation scheme without trusted authority in smart grid, IEEE Systems Journal 15 (1) (2020) 395–406.

[24] H. Shen, M. Zhang, J. Shen, Efficient privacy-preserving cube-data aggregation scheme for smart grids, IEEE Transactions on Information Forensics and Security 12 (6) (2017) 1369–1381.

[25] O. R. Merad-Boudia, S. M. Senouci, An efficient and secure multidimensional data aggregation for fog-computing-based smart grid, IEEE Internet of Things Journal 8 (8) (2020) 6143–6153.

[26] C. Peng, M. Luo, H. Wang, M. K. Khan, D. He, An efficient privacy-preserving aggregation scheme for multidimensional data in iot, IEEE Internet of Things Journal 9 (1) (2021) 589–600.

[27] Z. Liu, Z. Cao, X. Dong, X. Zhao, T. Liu, H. Bao, J. Shen, Epmda-fed: Efficient and privacy-preserving multidimensional data aggregation scheme with fast error detection in smart grid, IEEE Internet of Things Journal 9 (9) (2021) 6922–6933.

[28] C. Peng, M. Luo, P. Vijayakumar, D. He, O. Said, A. Tolba, Multi-functional and multidimensional secure data aggregation scheme in wsns, IEEE Internet of Things Journal 9 (4) (2021) 2657–2668.

[29] X. Yan, W. W. Ng, B. Zhao, Y. Liu, Y. Gao, X. Wang, Fog-enabled privacy-preserving multi-task data aggregation for mobile crowd-sensing, IEEE Transactions on Dependable and Secure Computing (2023).

[30] G. Zhuo, Q. Jia, L. Guo, M. Li, P. Li, Privacy-preserving verifiable set operation in big data for cloud-assisted mobile crowdsourcing, IEEE Internet of Things Journal 4 (2) (2016) 572–582.

[31] T. Li, C. Gao, L. Jiang, W. Pedrycz, J. Shen, Publicly verifiable privacy-preserving aggregation and its application in iot, Journal of Network and Computer Applications 126 (2019) 39–44.

[32] C. Guo, X. Jiang, K.-K. R. Choo, X. Tang, J. Zhang, Lightweight privacy preserving data aggregation with batch verification for smart grid, Future Generation Computer Systems 112 (2020) 512–523.

[33] X. Zhang, C. Huang, C. Xu, Y. Zhang, J. Zhang, H. Wang, Key-leakage resilient encrypted data aggregation with lightweight verification in fog-assisted smart grids, IEEE Internet of Things Journal 8 (10) (2020) 8234–8245.

[34] X. Zhang, C. Huang, Y. Zhang, S. Cao, Enabling verifiable privacy-preserving multi-type data aggregation in smart grids, IEEE Transactions on Dependable and Secure Computing 19 (6) (2021) 4225–4239.

[35] Y. Ding, B. Wang, Y. Wang, K. Zhang, H. Wang, Secure metering data aggregation with batch verification in industrial smart grid, IEEE Transactions on Industrial Informatics 16 (10) (2020) 6607–6616.

[36] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: International conference on the theory and applications of cryptographic techniques, Springer, 1999, pp. 223–238.

[37] T. P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: Annual international cryptology conference, Springer, 1991, pp. 129–140.

[38] M.-H. Park, Y.-H. Park, H.-Y. Jeong, S.-W. Seo, Key management for multiple multicast groups in wireless networks, IEEE Transactions on Mobile Computing 12 (9) (2012) 1712–1723.