A Lightweight and Privacy-preserving Distributed Multidimensional Data Trend Query Scheme with Fault-tolerant for Machine-as-a-Service

Tianci Zhao, Yuanjian Zhou, Chi Zhang, Weizhi Meng, *Senior Member, IEEE*, Zhengjun Jing

***Abstract*—In the Machine-as-a-Service (MaaS) model, enterprises can significantly reduce production costs by leasing equipment from original equipment manufacturers (OEM), while OEM can enhance equipment quality by utilizing equipment status data shared by enterprises. As such, MaaS is emerging as a very promising paradigm in modern manufacturing. However, the equipment failure data trend formed by the frequency of multiple types of failures may leak the private information of enterprise production, particularly when the same OEM leases the same type of equipment to multiple enterprises. Currently, there is no targeted and feasible solution to ensure the integrity, availability and privacy of multi-user and multi-dimensional data in the MaaS model. To address this gapt, this paper proposes a lightweight, privacy-preserving and fault-tolerant distributed trend query scheme for multi-dimensional data in MaaS environments. The proposed scheme ensures data privacy through local differential privacy (LDP), and guarantees fault tolerance and data integrity using Shamirs secret sharing and hash-based message authentication code (mac). To protect aggregated trend results, we design a weighted noise injection query algorithm based on LDP. Additionally, the scheme mitigates the risk of data leakage by introducing a blockchain-enabled proxy cloud server (CS). We formally verify the security properties of the proposed scheme, and our experimental evaluation demonstrates that it outperforms existing approaches in terms of computational efficiency and communication overhead.**

***Index Terms*—Machine-as-a-Service, Data trend query, Privacy-preserving, Fault-tolerant, Blockchain.**

I. INTRODUCTION

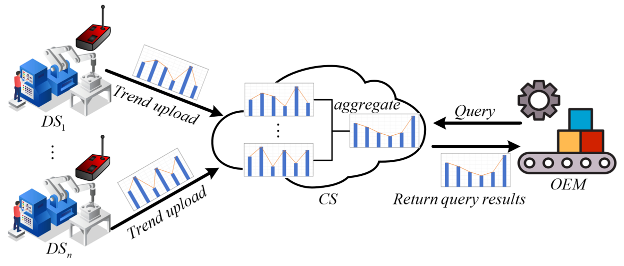
WITH the continuous advancement of Industrial Internet of Things (IIoT) technology, enterprises have experienced significant improvements in production efficiency. However, this progress has also accelerated the wear and tear of production equipment, leading to increased operational [1], [2]. In this context, Machine-as-a-Service (MaaS) has emerged as a novel industrial production model that disrupts traditional boundaries between equipment ownership and operational models [3], [4]. In the MaaS model, original equipment manufacturer (OEM) lease equipment to device users (DUs), who then utilize the leased machinery in their production processes. During operation, the equipment collects various types of datasuch as tool wear, heat dissipation faults, and other operational anomaliesthat may arise under similar production conditions across different DUs [5].

Corresponding author: Zhengjun Jing.

Tianci Zhao, Yuanjian Zhou, Chi Zhang, Zhengjun Jing are with the School of Computer Engineering, JiangSu University of Technology, Changzhou, Jiangsu, China.

Weizhi Meng is with the School of Computing and Communications, Lancaster University, LA1 4WA, United Kingdom.

Fig. 1. CS-based data trend query framework for MaaS



The multidimensional data composed of different data types can form a multidimensional data trend which reflects the operating status of leased devices. As shown in Fig. 1, considering the predictive maintenance work of enterprise, each device uploads the data trend to the cloud server (CS). Then, OEM queries the statistical results of multidimensional data aggregation trend result. During the query process, if the multidimensional data trend of a single device is not effectively protected, the production privacy information (e.g., product types, production efficiency, production capacity) of DU on the device may be leaked. Meanwhile, in the practical application of MaaS, the device sensors (DSs) will collect the multidimensional data generated by different DU using this device in the cycle time and upload these data to CS after processing them. CS computes, analyzes and stores these data and provides the data query services to data requester. To prevent the multidimensional data collected by DS from being stolen by others to leak the production privacy, researchers often make use of the data aggregation schemes [6] to protect data privacy. However, designing a multidimensional data trend query scheme involving multiple devices, multiple users, and various types of data for fault-tolerant in MaaS still faces some questions which need to be addressed.

The concept of data aggregation is widely used in IIoT [7], smart grid [8] and other technologies. The data aggregation can ensure the privacy of sensitive data through homomorphic encryption, differential privacy (DP) and so on. For example, Shen et al. [9] proposed a privacy-preserving data aggregation scheme based on the Chinese remainder theorem (CRT) and homomorphic encryption that encrypts data before transmission. However, the additional data encryption introduces new computation and storage overhead for the data owner, which is not suitable for DS with limited resources. Therefore, Zhao et al. [10] proposed a privacy-preserving multidimensional and multisubset data aggregation scheme based on CRT and

DP to reduced computation overhead while protecting data privacy. However, if OEM obtains the multidimensional data aggregation trend results of n and n-1 industrial area network (IAN), the privacy of multidimensional data from different IANs will inevitably be exposed. Furthermore, if one of the devices fails or the network connection is interrupted, resulting in data submission failure, the data aggregation process will also be affected, making it impossible for the control center to obtain the correct aggregation results. This is because the above schemes lack the fault-tolerant, which prevents them from coping with these unexpected situations.

Some researchers have proposed the data aggregation scheme combining with other technologies to solve the fault-tolerant problem. However, the extra operations in a fault-tolerant mechanism add extra computation and communication overhead to the system. Song et al. [11] proposed a fault-tolerant data aggregation scheme using the extended the Shamir secret sharing scheme. However, the reuse of shared key still needs additional computation and communication resources in this scheme. Lyu et al. [12] proposed a local differential privacy (LDP)-based data aggregation scheme that employs the one-time-pad (OTP) encryption algorithm and Gaussian mechanism to protect data privacy. However, this scheme needs to generate the secret key before each encryption operation. Moreover, when that a device fails requires fault-tolerant processing, interactions between nodes and a trusted authority (TA) are required, introducing the additional system overhead. Additionally, the data aggregation schemes [12], [13] that ensure data privacy through LDP typically result in approximate aggregated values. Therefore, our proposed scheme needs to strike a balance between data privacy and availability during the design.

However, when all the data is centralized in the CS, it can not avoid the centralized problems such as single point attack and vulnerability to attacks. To guarantee the security of data sharing, researchers have paid attention to the blockchain (BC), that use its decentralized and tamper proof features to replace CS. For example, Zhao et al. [14] proposed a BC-based privacy-preserving data aggregation scheme that does not require a trusted third party (TTP). Compared to the traditional CS, the decentralized structure of BC can effectively avoid the risk of data leakage caused by the single-point attack and centralized storage. To solve the above differential attack issue in the data aggregation scheme, Tao et al. [17] proposed a DP-based data query scheme through a three-stage framework. However, they only considered differential attack in numerical data, while multidimensional data aggregation trend results also face the same issue. Therefore, how to effectively defend against differential attack in multidimensional data aggregation trend result remains a key challenge in the data aggregation scheme. Meanwhile, some malicious adversaries may forgery or tamper with the data during transmission to disrupt the integrity of data. This leads to significant damage to the OEM's data analysis. Therefore, in MaaS, we not only need to protect the privacy of multidimensional data, but also should consider the data availability.

In this scheme, we proposed a lightweight and privacy-preserving distributed multidimensional data trend query scheme with fault-tolerant for MaaS. The main contributions of this paper are as follows:

1) We design a lightweight and privacy-preserving distributed multidimensional data trend query scheme by integrating the local differential privacy (LDP) with Shamir's secret sharing, ensuring privacy, integrity, and fault-tolerant without relying on a TTP. In addition, we adopt Hyperledger Fabric as a blockchain platform to enhance data security and enable secure data queries for OEM.
2) Our proposed scheme enables privacy-preserving aggregation of multidimensional data trend results through LDP and BC. This approach safeguards the data privacy of DUs while simultaneously fulfilling the data analysis needs of OEM.
3) The experimental evaluation demonstrates that our proposed scheme outperforms the existing schemes in terms of functions, computation and communication overhead.

The proposed scheme is organized as follows: Section II presents the related work. Section III presents preliminaries information. In Section IV, we outline the system model and the threat model, and design goals. The details of the proposed scheme are elaborated upon in Section V. A security analysis is conducted in Section VI. Section VII addresses evaluation analysis. Finally, Section VIII concludes the paper.

II. Related Work

In recent years, the various privacy-preserving data aggregation schemes for IIOT have been proposed. However, little attention has been paid to privacy concerns regarding about data query, especially in the term of privacy of multidimensional data aggregation trend results.

Data aggregation is a process of collecting and integrating information from various data sources, ensuring the privacy of sensitive data through anonymity, de-identification and other methods. However, most data aggregation schemes [18]–[20] only focus on the single dimensional data aggregation at present. For example, Gao et al. [19] proposed a distributed data aggregation scheme based on Paillier encryption algorithm which solves the problem of data privacy protection by introducing a random number mechanism. Su et al. [20] proposed a lightweight and efficient data aggregation scheme which addresses the data privacy by masks. However, the single-dimensional data aggregation scheme no lo nger meets the demands of data diversification in the information age. To solve this problem, Merad et al. [21] structured the multidimensional data by the homomorphic encryption algorithm to ensure the data privacy security. Shen et al. [22] proposed a data aggregation scheme based on homomorphic encryption algorithm which can not only resist malicious data mining attacks, but also output accurate aggregation results. However, most of the aggregation schemes based on homomorphic encryption algorithms require the high computation overhead. In order to reduce the computation overhead, Chen et al. [23] proposed a scalable multidimensional data aggregation scheme based on the Elliptic Curve Cryptogray (ECC) to support the flexible addition and removal of devices in this scheme. Peng et al. [24] proposed a data aggregation scheme based on Chinese Remainder Theorem (CRT) to improve the privacy protection ability of multidimensional data. Gai et al. [25] proposed a data aggregation scheme based on LDP and random response to protect the privacy of users. Jing et al. [26] proposed a privacy-preserving scheme for data trend based on local differential privacy and BC, which effectively prevents the privacy leakage. However, the prerequisite for these schemes to function correctly is that these nodes - such as collection nodes (e.g., DS), aggregation nodes (e.g., GateWay (GW)), and storage nodes (e.g., CS) - must operate normally. In MaaS, these nodes may fail to complete their works due to the network issue, hardware failure, and so on. Therefore, we need to design a method that meets the fault tolerance and security of stored data.

On the one hand, some researchers proposed various schemes [11]–[13], [27]–[29] to ensure the fault-tolerant of scheme. For example, Wang et al. [27] proposed a fault-tolerant multisubset data aggregation scheme which eliminates the dependence of trusted third part and can aggregate the user number and data in each region of domain. Since the fault-tolerant mechanism of this scheme, all undamaged devices need to upload the data again when a device fails. It will bring high computation and communication overhead. Ming et al. [28] proposed a fault-tolerant and efficient data aggregation scheme which applies the symmetric homomorphic encryption and ECC to ensure the data aggregation even if some device fail. Zhang et al. [29] proposed a fault-tolerant multidimensional data aggregation scheme based on Elliptic Curve ElGamal algorithm to ensure the accuracy of aggregation results when some devices fail.

On the other hand, to prevent the data privacy problems caused by CS storage, Li et al. [30] proposed a privacy-preserving data sharing scheme combining the BC and identity-based proxy re-encryption which achieves the access control in the aggregation process by re-encryption policy. To further enhance the application of BC, Hyperledger Fabric, as a enterprise-level BC platform, is introduced to provide a more flexible solution. Chen et al. [31] proposed a BC-based verifiable multidimensional data aggregation scheme which achieves the data integrity verification through the privacy preserving aggregation, Rivest Shamir Adleman (RSA) multiplicative homomorphic commitments and multi chain consortium BC architecture. It can be seem that the data integrity is important for data aggregation schemes. Wang et al. [32] proposed a distributed privacy-preventing data aggregation scheme using the BC and homomorphic encryption which ensures the data privacy and security.

Meanwhile, to ensure the data integrity, Guo et al. [33] proposed a distributed lightweight secure data aggregation scheme and applied the improved symmetric homomorphic encryption algorithm for batch authentication. Zhang et al. [34] proposed a batch authentication algorithm based on Paillier encryption algorithm and bilinear pairing in which two pairing operations are used to complete one authentication. However, this scheme needs a large number of bilinear pairing operations to complete batch authentication, resulting in excessive computation and communication overhead.

In summary, we propose a lightweight and privacy-preserving distributed multidimensional data trend query scheme with fault-tolerant for MaaS that supports the multidimensional data and protects the privacy of multidimensional data aggregation trend result through analyzing the advantages and disadvantages of existing schemes.

III. PRELIMINARIES

This section presents preliminaries from three aspects: the notation, local differential privacy and the Shamir secret sharing scheme.

A. Notation

TABLE I shows the notation and their description in our proposed scheme.

TABLE I
NOTATION DESCRIPTION IN OUR SCHEME

Notation	Description
TA	Trusted authority
DS	Device sensor
IAN	Industrial area network
GW	Gateway
BC	Blockchain
OEM	Original equipment manufacture
z/v	the number/index of IAN
n/i	the number/index of DS within a single IAN
w/j	the number/index of fault type
m/l	the number/index of GW
k	the threshold value
α	the corresponding entity service
d	the original multidimensional data
d'	the perturbed multidimensional data
$sh_l(d'_{ij})$	the l-th secret share of d'_{ij}
mac	the message authentication code
t_{cur}	the current time stamp
rd	the current round number
ε,δ	the privacy budget, slack factor
$Sh_l(j)$	the j-th aggregated message in GW_l
$\|\|$	the serial operation

B. Local Differential Privacy

Local differential privacy (LDP) is a technology that ensures personal data privacy and aims to provide strong privacy protection. Its definition used in the scheme is as follows:

(ε, δ)-LDP:(ε, δ)-LDP adopts the nearest neighbor mean estimation mechanism [35]. This mechanism satisfies (ε, δ)-LDP, and its principle is to perturb the true value x_{ij} to a nearby range with a high probability r to improve the data usability; to perturb it to a distant range with a low probability t to enhance data privacy, as shown in formulas 1 and 2. Here, x'_{ij} is the perturbed value and the parameter b decreases as ε increase. Through adjusting the value of b, a balance between data usability and privacy can be achieved.

$$t = \frac{1-2b\delta}{1+2be^{\varepsilon}}, r = \frac{e^{\varepsilon}+\delta}{1+2be^{\varepsilon}}, \\ b = \frac{e^{\varepsilon}-1-\varepsilon(e^{\varepsilon}+\delta)}{2(e^{\varepsilon}(1-e^{\varepsilon})+\varepsilon(e^{\varepsilon}+\delta))}. \tag{1}$$

$$P(x'_{ij}|x_{ij}) = \begin{cases} r, |x_{ij}-x'_{ij}| \le b \\ t, |x_{ij}-x'_{ij}| > b \end{cases} \tag{2}$$

C. The Shamir Secret Sharing Scheme

The Shamir secret sharing scheme was first proposed by Shamir [36] and Blakley [37] in 1979. Simply put, the Shamir secret sharing scheme is a (k,m) threshold sharing scheme in which the secret s is decomposed into m shares by constructing a $(k-1)$ order polynomial $f_k(s,x)$. Then, distribute them to m sharers. If the number of shares is greater than or equal to k, the secret s will be recovered. The main process and algorithms involved in the Shamir secret sharing scheme are as follows:

Allocation phase: The manager assigns the secret s to m shares P_i $(1 \le i \le m)$ through the polynomial 3 below:

$$f_k(x,s) = s + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}(mod\ p). \quad (3)$$

Among them, m is the number of sharers, k is the threshold value and p is a large prime number. The manager selects randomly a public value x_i for each sharer and computes the sub-share y_i according to the formula 4. Then, he sends the obtained share (x_i, y_i) to the corresponding sharer.

$$y_i = f_k(x_i, s), 1 \le i \le m. \quad (4)$$

Recovery phase: Given no less than k sharer (x_1,y_1),(x_2,y_2),...,(x_t,y_t), $f_k(x,s)$ can be reconstructed through the Lagrange interpolation theorem 5:

$$f_k(x,s) = \sum_{i=1}^{y}(y_i \prod_{j=1,j\neq i}^{k} \frac{x-x_j}{x_i-x_j})(mod\ p). \quad (5)$$

Once $f_k(x,s)$ is obtained, the secret s can be easily implemented through $s = f_k(0,s)$.

IV. BACKGROUND

In this section, we introduce the multidimensional data trend query framework, threat model, and design goals.

A. Multidimensional Data Trend Query Framework

As shown in Fig. 2, there are mainly six entities in this scheme: TA, DS, IAN, GW, BC and OEM.

1) **TA**. As an honest and trustworthy third party, TA responsible for generating the system parameters. Meanwhile, it registers each DS and GW to ensure the legitimacy of their identity and is responsible for initializing BC network and deploying the smart contract.

2) **DS**. DS is responsible for collecting multidimensional data generated by different DU using the device within the cycle time. Then, it will perturb and split the data. Finally, DS generates the mac and send the message to the corresponding GW.

3) **IAN**. IAN is an industrial local area network composed of different DS using to divide the physical areas of devices. IAN monitors its devices in real-time by connecting different devices.

4) **GW**. GW is an honest and curious entity responsible for collecting data from different DS, verifying and aggregating them. Then, GW generates the mac and corresponding aggregated data to send to BC.

5) **BC**. BC is built using the Hyperledger Fabric and is mainly responsible for verifying, storing aggregated data from GW and responding to inquiry requests from OEM.

6) **OEM**. OEM is a semi honest entity. By calling the query algorithm in the smart contract, OEM can query the trend aggregation results of multidimensional data.

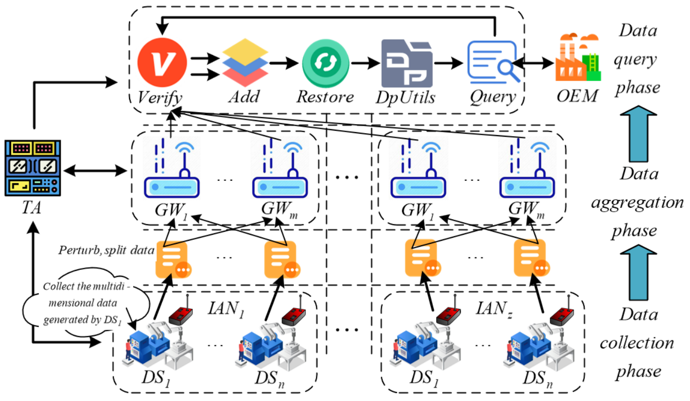


Fig. 2. Multidimensional data trend query framework

B. Threat model

In our proposed scheme, we divides the potential threats into the following two parts:

1) TA is only used for system initialization and entity registration. Meanwhile, we believe that GW is both honest and curious. This means while GW can accurately execute the plan, it will still attempt various methods to infer the user's privacy.
2) The external adversary $\mathcal{A}$ eavesdrops on messages exchanged between different parties, namely between DS and GW and between GW and BC. In addition, $\mathcal{A}$ can establish some proactive attack, such as forging captured messages or replay attacks, etc. Therefore, there may be a crisis for GW. Since GW is a powerful entity, destroying it requires a significant cost. Thus, it is assumed that no more than a certain number of GWs, not exceeding $k = \lceil m/2 \rceil - 1$, can be destroyed or compromised. However, the system still has $(m-k)$ GW available for restoring aggregated messages, in which $m - k \ge k + 1$.

C. Design goals

Our proposed scheme aims to design a lightweight and privacy-preserving distributed multidimensional data trend query scheme with fault-tolerant for MaaS, which ensures data privacy and integrity while meeting the fault-tolerant. Specifically, this scheme should meet the following design goals:

1) Privacy. Multidimensional data contains private data such as the production privacy of DU. Our scheme should protect the privacy and security of DU's production data.

Meanwhile, OEM can only obtain the overall aggregated trend result of the multidimensional data and cannot access other trends except for the overall trend.

2) Integrity. During the process of sending messages from the sender (e.g., DS and GW) to the receiver (e.g., GW and BC), the adversary may launch attacks such as forgery or tampering. Therefore, the receiver must have the capability to verify the integrity of data. Meanwhile, batch verification operations are introduced to ensure the efficient operation of the scheme.
3) Fault-tolerant. Due to environmental, human, and other factors, some DS and GWs have ceased to function. Thus, the proposed scheme should have fault tolerance function to ensure that the aggregation scheme can process in an order manner.
4) Decentralization. Decentralization means that the recovery process of ciphertext no longer relies on a centralized server, avoiding the unrealistic issue that the server must be honest or semi honest. Meanwhile, unlike the black box model of centralized scheme, decentralization method makes the scheme more transparent and trustworthy.
5) Other network attacks. (1) Anti-Spoofing Attack: Although our proposed scheme employs an effective verification mechanism, it ensures that the messages sent by each entity can be correctly identified, and effectively preventing illegal impersonation. (2) Anti-Man-in-the-Middle attack: An effective message verification mechanism is employed to prevent this attack. Even if an external adversary $\mathcal{A}$ intercepts the transmitted message, he will be unable to tamper with, forge or steal the message. (3) Anti-Collusion attack: Even if some entities cooperate to launch this attack, our proposed scheme employs a distributed multidimensional data sharing mechanism to ensure the data security.

V. OUR PROPOSED SCHEME

In this section, we introduce the proposed scheme, that mainly consists of six phases: initialization phase, registration phase, data collection phase, data aggregation phase, data query phase, and smart contract design.

A. Initialization phase

The TA executes the initialization algorithm to generate the system parameters through the following steps and sends them to the corresponding entity.

Step 1: On input a security parameter λ, the algorithm first sets the appropriate privacy budget ε and slack factor δ to DS and embeds ε, δ into it.

Step 2: Select randomly two security hash function H_1, H_2.

Step 3: Complete the initialization operation of the BC platform and deploy smart contracts.

Step 4: Select three appropriate parameters m,k,p (the number of shares, the threshold value and a large prime number) according to the Shamir secret sharing scheme and generates a $(k-1)$ order polynomial: $F(x) = a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1} + d\ (mod\ p)$.

B. Registration phase

When DS/GW/OEM sends a registration request with their real identity and the service that they need, TA executes the registration algorithm to generate the corresponding parameters and then sends them to DS/GW/OEM through a secure channel.

1) DS_i registration: On input DS_i's real identity id_{DS_i} and service α (send the collected data to GW), TA obtains the current time stamp and computes the identification $ID_{DS_i} = H_1(id_{DS_i}\|\alpha\|t_{cur})$. Finally, TA sends $\{id_{DS_i}, \alpha, t_{cur}, ID_{DS_i}\}$ to DS_i and sends $\{\alpha, ID_{DS_i}\}$ to GW.

2) GW_l registration: On input GW_l's real identity id_{GW_l} and service α (send the aggregated data to BC), TA obtains the current time stamp and computes the identification $ID_{GW_l} = H_1(id_{GW_l}\|\alpha\|t_{cur})$. Finally, TA sends $\{id_{GW_l}, \alpha, t_{cur}, ID_{GW_l}\}$ to GW_l and sends $\{\alpha, ID_{GW_l}\}$ to BC.

3) OEM registration: On input OEM's real identity id_{OEM} and service α (make a query transaction to BC), TA obtains the current time stamp and computes the identification $ID_{OEM} = H_1(id_{OEM}\|\alpha\|t_{cur})$. Finally, TA sends $\{id_{OEM}, \alpha, t_{cur}, ID_{OEM}\}$ to OEM and sends $\{\alpha, ID_{OEM}\}$ to BC.

C. Data collection phase

During this cycle time of using the device i, DS_i periodically (e.g., every week or month) collects multidimensional data generated by different DU. Then, DS_i perturbs and splits the collected data and generates the corresponding mac. Finally, DS_i sends the signed message to the corresponding GW. The details are as follows:

Step 1: Collect w-dimensional data $d_i = \{d_{i1}, d_{i2}, \ldots, d_{iw}\}$ $(j = 1, 2, \ldots, w)$ within the cycle time. Then, DS_i perturbs and splits d_i to obtain the $(w \times n)$ secret shares according to the algorithm 1.

Step 2: Compute the mac corresponding to the secret share $sh_l(d'_{ij})$: $mac^l_{ij} = H_2(i\|j\|l\|rd\|t_{cur}\|ID_{DS_i}\|sh_l(d'_{ij}))$ in which i is the index of DS, j is the index of data type, l is the index of secret share, rd is the current round number, t_{cur} is the current time stamp, ID_{DS_i} is DS_i's identification and $sh_l(d'_{ij}) = f_k(d'_{ij}, l)(mod\ q)$ $(l = 1, 2, \ldots, m)$.

Step 3: Send the signed message $\{i, j, l, rd, t_{cur}, sh_l(d'_{ij}), mac^l_{ij}\}$ to GW_l.

It is worth noting that the maximum power exponent of the Shamir secret sharing scheme polynomial used by each DS is the same, that is $k-1$. The other DS send their own message to the corresponding GW in the same way.

D. Data aggregation phase

Upon receiving the message $\{i, j, l, rd, t_{cur}, sh_l(d'_{ij}), mac^l_{ij}\}$ from DS_i, GW_l first verifies whether t_{cur} is fresh. If not, the message is discarded. Then, GW_l verifies whether the mac is valid. If not, the mac is discarded; otherwise, GW_l classifies the messages according to the services. Assume there are $(n \times w)$

Algorithm 1 Data process algorithm based LDP and Shamri secret sharing scheme

```
Input: The collected data $d_i$, privacy budget $\varepsilon$ and slack factor $\delta$;
Output: The $(w \times m)$ secret shares;
1: Refer to the formula 1, 2 and generate the noise $noise_i$;
2: Generate the perturbed multidimensional data $d'_i$:
3: for $j = 1, 2, \ldots, w$ do
4:     $d'_{ij} \leftarrow d_{ij} + noise_{ij}$;
5: end for
6: $d'_i = \{d'_{i1}, \ldots, d'_{ij}, \ldots, d'_{iw}\}$;
7: Refer to the formula 3,4 and divide $d'_{ij}$ into $m$ secret shares: $d'_{ij} \rightarrow \{sh_1(d'_{ij}), \ldots, sh_l(d'_{ij}), \ldots, sh_m(d'_{ij})\}$;
8: return $(w \times m)$ secret shares:
```

$$\begin{pmatrix} sh_1(d'_{i1}), \ldots, sh_l(d'_{i1}), \ldots, sh_m(d'_{i1}) \\ \vdots \\ sh_1(d'_{ij}), \ldots, sh_l(d'_{ij}), \ldots, sh_m(d'_{ij}) \\ \vdots \\ sh_1(d'_{iw}), \ldots, sh_l(d'_{iw}), \ldots, sh_m(d'_{iw}) \end{pmatrix}$$

messages from DS, which $\{1, 1, l, rd, t_{cur}, sh_l(d'_{11}), mac^l_{11}\}$, $\{1, 2, l, rd, t_{cur}, sh_l(d'_{12}), mac^l_{12}\}$, ..., $\{n, w, l, rd, t_{cur}, sh_l(d'_{nw}), mac^l_{nw}\}$. GW_l executes the following steps.

Step 1: Verify whether the messages $\{1,1,l,rd,t_{cur}, sh_l(d'_{11}), mac^l_{11}\}$, $\{1, 2, l, rd, t_{cur}, sh_l(d'_{12}), mac^l_{12}\}$, ..., $\{n, w, l, rd, t_{cur}, sh_l(d'_{nw}), mac^l_{nw}\}$ are valid:

$$mac^l_{ij} \stackrel{?}{=} H_2(i\|j\|l\|rd\|t_{cur}\|ID_{DS_i}\|sh_l(d'_{ij})). \tag{6}$$

Subsequently, to improve the verification efficiency, GW_l verifies the batch message by checking the following equation:

$$\sum_{i=1}^{n}\sum_{j=1}^{w} mac^l_{ij} \stackrel{?}{=} \sum_{i=1}^{n}\sum_{j=1}^{w} H_2(i\|j\|l\|rd\|t_{cur}\|ID_{DS_i}\|sh_l(d'_{ij})). \tag{7}$$

If the equation holds true, return 1 and continue to the next steps; otherwise, return $\perp$.

Step 2: After the verification, GW_l aggregates the data according to the addition homomorphism property by the following equation: $Sh_l(1) = \sum_{i=1}^{n} sh_l(d'_{i1}), Sh_l(2), \ldots, Sh_l(w)$.

Step 3: Compute the mac corresponding to the aggregated data $Sh_l(j)$: $mac^v_{jl} = H_2(j\|l\|v\|rd\|t_{cur}\|ID_{GW_l}\|Sh_l(j))$ in which v is the index of IAN, ID_{GW_l} is GW_l's identification and $Sh_l(j)$ is the j-th type of aggregated data.

Step 4: Send the aggregated message $\{j, l, v, rd, t_{cur}, Sh_l(j), mac^v_{jl}\}$ to BC.

E. Data query phase

Upon receiving the aggregated message $\{j, l, v, rd, t_{cur}, Sh_l(j), mac^v_{jl}\}$ from GW_l, the BC first verifies whether t_{cur} is fresh. If not, the aggregated message is discarded. Then, BC verifies whether the mac is valid. If not, the mac is discarded; otherwise, BC stores the aggregated messages on the chain according to the services. Assume there exists $(w \times m \times z)$ aggregated messages from GW, which $\{1, 1, 1, rd, t_{cur}, Sh_1(1), mac^1_{11}\}$, $\{2, 1, 1, rd, t_{cur}, Sh_1(2), mac^1_{21}\}$, ..., $\{w, m, z, rd, t_{cur}, Sh_m(w), mac^z_{wm}\}$. BC executes the following steps:

Step 1: Verify whether the aggregated messages $\{1, 1, 1, rd, t_{cur}, Sh_1(1), mac^1_{11}\}$, $\{2, 1, 1, rd, t_{cur}, Sh_1(2), mac^1_{21}\}$, ..., $\{w, m, z, rd, t_{cur}, Sh_m(w), mac^z_{wm}\}$ are valid:

$$mac^v_{jl} \stackrel{?}{=} H_2(j\|l\|v\|rd\|t_{cur}\|ID_{GW_l}\|Sh_l(j)). \tag{8}$$

Subsequently, to improve the verification efficiency, BC verifies the batch aggregated messages by checking the following equation:

$$\sum_{v=1}^{z}\sum_{j=1}^{w}\sum_{l=1}^{m} mac^v_{jl} \stackrel{?}{=} \sum_{v=1}^{z}\sum_{j=1}^{w}\sum_{l=1}^{m} H_2(j\|l\|v\|rd\|t_{cur}\|ID_{GW_l}\|Sh_l(j)). \tag{9}$$

If the equation holds true, return 1 and continue to the next steps; otherwise, return $\perp$.

Step 2: After the verification, the aggregated message is packaged into a block and broadcast to the BC. Finally, each BC node links the newly generated block to the local BC ledger to ensure the consistency of ledger. Similarly, once the newly generated block linking to the BC, the nature of BC such as immutability and so on can ensure the data integrity and authentication of all transactions in the block.

Step 3: When OEM needs to make a query, OEM makes a query transaction to the BC. Then, BC return the corresponding query transaction. The process is as follows:

1) According to the demand for querying IAN quantity, OEM generates a query transaction $q = \{id, number, startTime, endTime, query\}$ and computes the mac corresponding to q: $mac_{OEM} = H_2(q\|t_{cur}\|ID_{OEM})$. Then, OEM sends $\{q, t_{cur}, mac_{OEM}\}$ to BC.

2) Upon receiving the query message, BC first verifies whether t_{cur} is fresh. If not, the query message is discarded. Then, BC verifies whether the mac is valid. If not, the mac is discarded; otherwise, BC calls the algorithm 2 to obtain the multidimensional data aggregation trend results q^* with privacy-preserving.

A. Verify whether the query message is valid: $mac_{OEM} = H_2(q\|t_{cur}\|ID_{OEM})$. If it holds true, return 1 and continue to the next steps; otherwise, return $\perp$.

B. According to the OEM query, there exists the following two cases:

  * Case 1: When $1 <= number < z$, the multidimensional data in some IAN is queried. In this case, multidimensional data from $number$ IAN are randomly selected for summation. Because $1 < number < z$, not meeting the condition of $\sum_{t=1}^{z} noise^t_j = 0$ leads to strong noise $noise$ in the process of summing each other that cannot be eliminated. In the end, all types

of data are affected by strong noise $noise$ which changes the aggregation trend of multidimensional data in the query area, making the multidimensional data trend in a single IAN hidden in the aggregation trend of these data.

* Case 2: When $numebr = z$, the multidimensional data in all IAN is queried. The process follows the steps of Case 2 which is summed according to the fault type. And because $number = z$, the strong noise $noise$ can cancel each other out when summing.

Algorithm 2 Weighted noise injection query algorithm

```
Input: The query transaction $q = \{id, number, startTime, endTime, query\}$;
Output: The multidimensional data aggregation trend result $q^* = \{sum_1, \ldots, sum_w\}$;
1: Extract the identity $id$ from the query transaction $q$;
2: Verify whether it has data access right according to $id$. If it holds true, continues the next steps; otherwise, returns $\perp$;
3: According to the formula 5, restore the aggregated shares: $s_j^v = \sum_{l=1}^{k}(Sh_l(j) \cdot \prod_{r=1,r\neq l}^{k} \frac{x-x_r}{x_l-x_r})(mod\ p)$.
4: Obtain the aggregated data: $\begin{Bmatrix} s_1^1, \ldots, s_j^1, \ldots, s_w^1 \\ \vdots \\ s_1^v, \ldots, s_j^v, \ldots, s_w^v \\ \vdots \\ s_1^z, \ldots, s_j^z, \ldots, s_w^z \end{Bmatrix}$
5: Give the different privacy budgets: $\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_w\}$ according to the permissions of restored multidimensional data. Then, generate the corresponding strong noise $noise$ according to the formulas 1 and 2 and make the $noise$ meet the below conditions:
   $\sum_{v=1}^{z} noise_1^v = 0, \ldots, \sum_{v=1}^{z} noise_w^v = 0$;
6: Add the strong noise $noise$ to the corresponding multidimensional data:
7: for $v = 1, 2, \ldots, z$ do
8:     for $j = 1, 2, \ldots, w$ do
9:         $s_j^{v\prime} = s_j^v + noise_j^v$;
10:    end for
11: end for
12: Extract the $number$ from query transaction $q$. Then, select random $number$ groups multidimensional data to aggregate:
13: for $j = 1, 2, \ldots, w$ do
14:     for $v = 1, 2, \ldots, number$ do
15:         $sum_j += s_j^{v\prime}$;
16:     end for
17: end for
18: return $q^* = \{sum_1, \ldots, sum_w\}$.
```

F. Smart contract design

The smart contract designed by the proposed scheme includes the following parts: two data structures (**Meaage**, **QueryResult**) and five interfaces (**Add**, **Verify**, **Query**, **Restore**, **DpUtils**). As shown in algorithm 3 and4:

Algorithm 3 The structure of smart contract

```
1: Structure Message {
2:     id, url, data, type String;
3:     setTime, setUser String; }
4: Structure QueryResult {
5:     key String; message Message; }
```

1) The structure **Message** defines the storage form of data from the GW. The structure **QueryResult** defines the result form of multidimensional data aggregation trend when OEM make a query transaction.

2. The interface **Add** is used frequently, which store aggregated messages validated through the interface **Verify** onto the BC. In addition, OEM can obtain the multidimensional data aggregation trend result through the interface **Query**. Firstly, through calling the interface **Restore** to restore the multidimensional aggregated data for each IAN. Then, these data are perturbed using noise generated via the interface **DpUtils**. Finally, according to the query transaction, the corresponding multidimensional data aggregation trend results are returned.

Algorithm 4 The function of smart contract

```
1: Add($j, l, v, rd, t_{cur}, Sh_l(j), mac_{jl}^v$):
2: if Verify($mac_{jl}^v$) then
3:     Store the aggregated messages $\{j, l, v, rd, t_{cur}, Sh_l(j), mac_{jl}^v\}$ in the form of Message in blocks.
4: end if
5: Query($q, t_{cur}, mac_{OEM}$):
6: if Verify($mac_{OEM}$) then
7:     Restore($1 : ((Sh_1(1), \ldots, Sh_1(w)), \ldots, (Sh_l(1), \ldots, Sh_l(w)), \ldots, (Sh_k(1), \ldots, Sh_k(w))), \ldots, v, \ldots, z$) to obtain the aggregated data $(s_1^1, \ldots, s_w^1), \ldots, (s_1^v, \ldots, s_w^v), \ldots, (s_1^z, \ldots, s_w^z)$;
8:     DpUtils($\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_w\}$,$\delta$) to generate the strong noise $noise$ and add the strong noise $noise$ to the corresponding multidimensional data;
9:     According to the query transaction $q$, aggregate the data to form the multidimensional data aggregation trend result $q^*$ in the form of QueryResult;
10: end if
11: return $q^* = \{sum_1, \ldots, sum_w\}$.
```

VI. Security Analysis

In this section, we present a security analysis focusing on the privacy, integrity, fault-tolerant, decentralization and other network attacks.

A. Privacy

Theorem 1: OEM cannot infer the privacy of a single IAN based on the multidimensional data aggregation trend result.

Proof: In our proposed scheme, DS perturbs the multidimensional data and splits them into secret shares according to the algorithm 1. Then, DS outputs the message and sends them to the corresponding GW. GW verifies the message and aggregates the secret shares. Then, GW outputs the aggregated message and sends it to BC. Finally, OEM sends a query transaction q to the BC based on its own requirement and obtains the multidimensional data aggregation trend result.

In this process, single-dimensional data is never sent as a numberical value, and GW and BC can only obtain the message $\{i, j, l, t_{cur}, rd, sh_l(d'_{ij}), mac^l_{ij}\}$ and the aggregated message $\{j, l, v, t_{cur}, rd, Sh_l(j), mac^v_{jl}\}$. Meanwhile, the LDP perturbation operation in the algorithm 1 and 2 can protect the privacy of these data. Therefore, OEM cannot infer the multidimensional data trend of a single IAN without getting to LDP noise.

Assume that there are four IAN uploading the aggregated messages to BC in the current system, OEM sends four kinds of query transaction (query for single, two, three and all IANs) and obtains the following four conditions. As shown in the Fig. 3. As can be seen from the Fig. 3, the proposed scheme adds weighted noise to various types data according to the algorithm 2 to change the aggregation trend of the queried IAN data. Only when OEM query all, the noise cancels each other and the multidimensional data aggregation trend without the noise is shown.

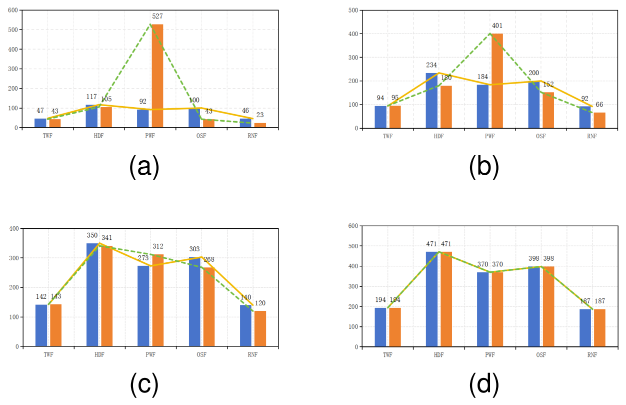


Fig. 3. Query results in different situations. (a) number=1. (b) number=2. (c) number=3. (d) number=4.

Meanwhile, in order to quantitatively analyze whether the added noise meets the requirements of the scheme, we compute the pearson correlation coefficient (PCC) between the original data trend and query result trend. The range of the PCC is between 0 and 1. A value close to 1 indicates a strong linear correlation between the two variables, while a value close to 0 suggests that there is almost no correlation between them. The Fig. 4 (a, b, c, d) corresponds respectively to Fig. 3 (a, b, c, d). The results show that the correlation between the original data trend and the query result trend under different query conditions is relatively low, further indicating that the introduction of noise in the algorithm 2 effectively disrupts the linear correlation between the data.

Theorem 2:The extend adversary $\mathcal{A}$ cannot obtain any data about the target DS.

Proof:In the process of data aggregation, the external adversary $\mathcal{A}$ cannot obtain obtain any information about noise.

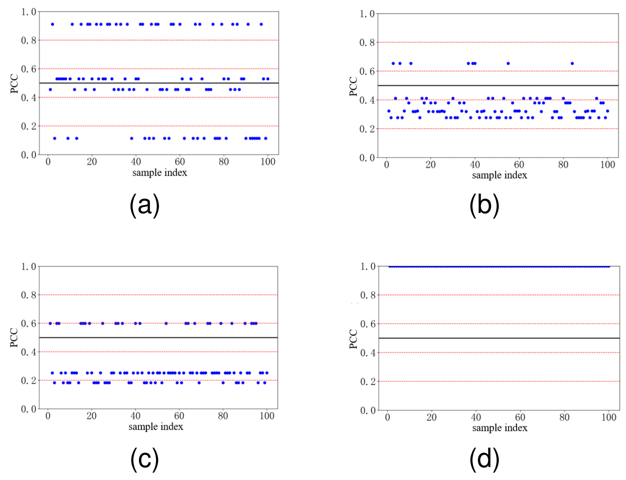
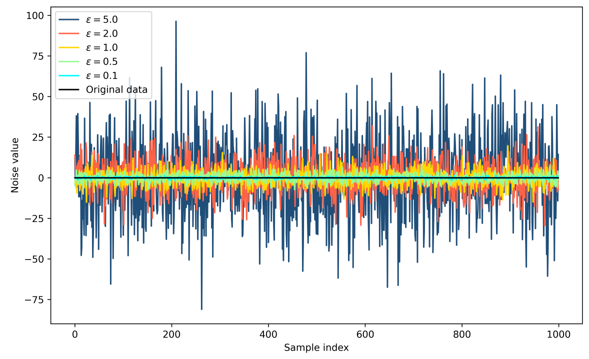


Fig. 4. PCC corresponding to query results in different situations. (a) number=1. (b) number=2. (c) number=3. (d) number=4.

Fig. 5. Comparison with original data under different privacy budgets



Meanwhile, the proposed scheme adopts the Shamir secret sharing scheme to achieve fault-tolerant which means that $\mathcal{A}$ cannot recover the original data without obtaining k or more secret shares. And this scheme achieves the resistance to collusive attack by introducing LDP noise. Thus, $\mathcal{A}$ cannot obtain any data about the target DS. Among them, LDP noise can protect the data privacy and the effect is shown in the Fig. 5. As can be seen from the Fig. 5 that different privacy budgets can achieve different privacy effects. The smaller the privacy budget, the larger the noise, providing stronger privacy protection; conversely, the larger the privacy budget, the smaller the noise.

B. Integrity

Theorem 3: DS can only pass the integrity verification of GW by sending the correct messages to GW.

Proof: Given $(n \times w)$ messages from DS, GW_l can verify these messages integrity by checking the equation 6 and 7. If they hold true, return 1 and continue to the next steps; otherwise, return $\perp$. Hence, if and only if DS sends the correct messages to GW, the integrity verification of GW can be passed.

Theorem 4: GW can only pass the integrity verification of BC by sending the correct aggregated message to BC.

Proof: Given $(w \times m \times z)$ aggregated messages from GW, BC can verify these aggregated messages integrity by checking the equation 8 and 9. If they hold true, return 1 and continue to the next steps; otherwise, return $\perp$. Hence, if and only if GW sends the correct aggregated messages to BC, the integrity verification of BC can be passed.

C. Fault tolerant

1) Fault-tolerant analysis:

- Fault-tolerant of GW: Under the same IAN, the system deploys m GW to work together. According to the assumption of threat model, a external adversary $\mathcal{A}$ can only destroy or compromise k GW and obtain their aggregated messages in the worst case. Then, due to the "all-or-nothing" property of the Shamir secret sharing scheme, at least k secret shares are required to recover. Therefore, the system still has $m-k$ GW which can be used normally for recovery, so as to ensure the proper functioning of the scheme.
- Fault-tolerant of DS: When some DS can not send the messages to GW normally, the GW can still perform the data aggregation, and BC can also restore the aggregated data. Assuming the total number of DS is n in a IAN, and the number of DS who cannot upload the message normally is $\hat{n}$, where $\hat{n} \in n$. The specific steps of GW and BC are as follows:

  Step 1: GW_l aggregates the messages:

$$Sh_l(1) = \sum_{i=1}^{\hat{n}\in n} sh_l(d'_{i1}), Sh_l(2), \ldots, Sh_l(w). \tag{10}$$

  Step 2: BC restores the aggregated messages:

$$s_j^v = \sum_{l=1}^{k}(Sh_l(j) \cdot \prod_{r=1, r\neq l}^{k} \frac{x - x_r}{x_l - x_r})(mod\ p). \tag{11}$$

  Step 3: BC can obtain the aggregated data:

$$\begin{Bmatrix} s_1^1, \ldots, s_j^1, \ldots, s_w^1 \\ \vdots \\ s_1^v, \ldots, s_j^v, \ldots, s_w^v \\ \vdots \\ s_1^z, \ldots, s_j^z, \ldots, s_w^z \end{Bmatrix} \tag{12}$$

.

2) Fault-tolerant probability:

Due to the characteristics of the Shamir secret sharing scheme, we can recover the data in the case of partial data loss. However, during the scheme implementation, there may be messages sent by DS that cannot be received by GW. Because that: 1) DS or GW may be offline due to network, hardware failure and so on; 2) Communication between DS and GW is interrupted by network, hardware failure and so on.

Next, we analyze the probability that GW fail to aggregate the data successfully. Let $P_{fail-DCP}$ denote the probability of data perturbation and split failure in the data collection phase. Similarly, let $P_{fail-DAP}$ denote the probability of data aggregation failure in the data aggregation phase. Assuming that the two phases are independent of each other, the probability of GW failure [38] can be expressed as:

$$P_{fail-GW} = P_{fail-DCP} + (1 - P_{fail-DCP})P_{fail-DAP}. \tag{13}$$

Obviously, the meaning of $P_{fail-DCP}$ is as follows:

$$P_{fail-DCP} = 1 - P(\text{at least k GW receive the messages sent fro m all DS}). \tag{14}$$

For GW, it can be further expressed as follows:

$$P_{fail-DCP} = 1 - \sum_{i=k}^{m} \binom{m}{i} (1-p_1)^{ni}(1-(1-p_1)^n)^{m-i}, \tag{15}$$

where p_1 is the probability that the DS sends the message incorrectly. But for the DAP, $P_{fail-DAP}$ is provided by:

$$P_{fail-DAP} = \sum_{i=m-(k-1)}^{m} \binom{m}{i} p_2^i (1-p_2)^{m-i}, \tag{16}$$

where p_2 is the probability that the secret shares collected by the GW is not successful.

D. Decentralization

In the proposed scheme, the storing aggregated messages and query algorithm is implemented as deployed smart contracts rather than using a center server. This allows us to avoid assuming a semi-honest or honest cloud server to execute our scheme as is the case with most existing solutions. Meanwhile, the consensus mechanism of BC ensures the correct execution of every operation. In practice, if the smart contract executor returns an incorrect result for some reason, but the malicious operation will be detected by other miners, the executor will get nothing. Therefore, the proposed scheme support decentralization.

E. Other network attacks

1) Anti-Spoofing Attack: Theorem 5: The data messages of external adversary $\mathcal{A}$ forging cannot pass the integrity verification by the receiving end.

Proof: In the proposed scheme, the external adversary $\mathcal{A}$ cannot generate the correct messages $\{i, j, l, rd, t_{cur}, sh_l(d'_{ij}), mac_{ij}^l\}$ and pass the integrity verification of GW without obtaining the target index i, j, l, the round number rd, the current time stamp t_{cur} and the secret share $sh_l(d'_{ij})$. Similarly, in this case, GW also cannot generate the aggregated messages and pass the integrity verification of BC. Therefore, the messages spoofed by $\mathcal{A}$ cannot pass the integrity verification of receiving end.

2) Anti-Man-in-the-Middle Attack: Theorem 6: The external adversary $\mathcal{A}$ cannot intercept, tamper with, or replay the messages between the sending end and the receiving end.

Proof: When $\mathcal{A}$ intercepts a message sent to the receiving end and attempts to modify the message, $\mathcal{A}$ must generate a matching signature; otherwise, it cannot pass the authentication of receiving end. In addition, $\mathcal{A}$ needs to know exactly the specific parameters of generating the message of sending end; otherwise, the data cannot be correctly recovered. Because the current time stamp is different from the old time stamp, $\mathcal{A}$ cannot use the previous data to pass the verification.

However, even if $\mathcal{A}$ has the ability to successfully complete the above operations and pass the verification of receiving end, there are still two possibilities: 1) In the data query phase, if this message is selected as one of the k threshold parameters, the data cannot be recovered, thus avoiding the production of wrong results; 2) If this message is not selected, it will not have any effect on the recovered data.

3) Anti-Collusion Attack: Theorem 7: In the same IAN, some GW jointly plan the multidimensional data is invalid.

Proof: In this scheme, this attack can be divided into the following two cases:

(1) The number of colluding GW is less than k: In this case, the Shamir secret sharing scheme requires at least k participants to together recover the data. If the number of colluding GW is less than k, the data cannot be properly recovered.

(2) The number of colluding GW is greater than or equal to k: In this case, the Shamir secret sharing scheme itself cannot effectively resist collusion attack initiated by k or more participants. However, we resistant the collusion attack through introducing the LDP noise into the data in the proposed scheme.

VII. EXPERIMENTAL EVALUATION

To evaluate the performance of the proposed scheme, we implemented it on a Window 10 (64-bit operating system, Intel Core i7-9750H CPU, 16.0 GB) and Ubuntu 18.04 (Dockers v20.101.21, Matplotlib 3.5.3, Flask 2.3.2, Numpy 1.25.1).

Our proposed scheme uses the dataset provided by UCI Machine Learning Database [5]. The dataset consists of 10,000 multidimensional fault data in which the fault types are: tool wear failure (TWF), heat dissipation failure (HDF), power failure (PWF), over strain failure (OSF) and random failure (RNF). There are two states of 0 or 1 for these five device fault types that 1 means that the device has a certain fault once and 0 means that the device has not a certain fault.

A. Functionality comparison

In this section, we compare the propose scheme with the existing schemes from five aspects of privacy, decentralization (only Chen et al. [31]), integrity, batch verification (only Shen et al. [22] and Gai et al. [25]) and fault-tolerant (only Chen et al. [23], Gai et al. [25] and Ming et al. [28]). Among them, F1: Privacy; F2: Decentralization; F3: Integrity; F4: Batch authentication; F5: Fault-tolerant. The performance comparison results are shown in TABLE II.

TABLE II
COMPARISON OF FUNCTIONALITY FEATURES

Scheme	F1	F2	F3	F4	F5
Merad et al. [21]	√	×	√	√	×
Shen et al. [22]	√	×	√	×	×
Chen et al. [23]	√	×	√	√	√
Gai et al. [25]	√	×	√	×	√
Ming et al. [28]	√	×	√	√	√
Zhang et al. [29]	√	×	√	√	×
Chen et al. [31]	√	√	√	√	×
Our scheme	√	√	√	√	√

B. Computation overhead

In the section, we compare the computation overhead of the proposed scheme with the existing schemes Merad et al. [21], Shen et al. [22], Chen et al. [23], Gai et al. [25], Ming et al. [28], Zhang et al. [29] and Chen et al. [31], in the data collection phase, data aggregation phase and data query phase.

Among them, the main computation operations are shown that t_{ecc}: denotes scale multiplication operation in elliptic curve cryptography; $t_{\mathbb{G}_T}$: denotes a exponential operations on $\mathbb{G}_T$ group; t_{m_N}: denotes multiplication operation on $\mathbb{Z}_N$; $t_{\mathbb{Z}}$: denotes a exponential operations on $\mathbb{Z}_q^*$ group; $t_{\mathbb{Z}_{N^*}^*}$: denotes a exponential operations on $\mathbb{Z}_{N^*}^*$ group; t_p: denotes a bilinear pairing operation; t_{mod}: denotes a mold taking operation; t_{add}: denotes a addition operation; t_{mul}: denotes a multiplication operation; t_{P_e}: denotes a homomorphic encryption; t_{P_d}: denotes a homomorphic decryption; t_{log}: denotes a discrete Logarithmic Operations; t_{s_g}: denotes a Shamir secret share generation; t_{s_r}: denotes a Shamir secret share restoration. The comparison of computation overhead between the proposed scheme and the existing schemes is shown in TABLE III.

Fig. 6 shows the comparison of computation overhead between the proposed scheme and the existing schemes. We can divide Fig. 6 into three parts: the data collection phase, the data aggregation phase and the data query phase.

In the data collection phase of Fig. 6, we can see that the proposed scheme has certain advantages over other schemes. The reason is that the proposed scheme perturbs the multidimensional fault data through the LDP technology and splits the perturbed data by the Shamir secret sharing scheme to meet the fault-tolerant.

In the data aggregation phase of Fig. 6, since the proposed scheme meets its fault-tolerant through the Shamir secret sharing scheme, the computation overhead of our scheme in the data aggregation phase is slightly larger than that of Gai et al. [25]. But Gai et al. [25] failed to achieve the batch authentication.

In the data query phase of Fig. 6, since the proposed scheme designs a weighted noise injection query algorithm for OEM query, the computation overhead of our scheme in the data aggregation phase is slightly larger than that of Gai et al. [25] and Chen et al. [31]. However, Chen et al. [31] doesn't meet the fault-tolerant.

In summary, the proposed scheme has certain advantages in terms of computation overhead compared with the other similar existing schemes.

C. Communication overhead

In the section, we compare the communication overhead of the proposed scheme with the existing schemes Merad et al. [21], Shen et al. [22], Chen et al. [23], Gai et al. [25], Ming et al. [28], Zhang et al. [29] and Chen et al. [31] in the DS to GW and GW to BC.

TABLE III
COMPARISON OF COMPUTATION AND COMMUNICATION OVERHEAD

Scheme	Computation overhead			Communication overhead	
	data collection	data aggregation	data query	DS-to-GW	GW-to-BC
Merad et al. [21]	$t_{ecc} + t_{P_e}$	t_{ecc}+(n+1)t_p	t_{P_d}+2t_p	$\|\mathbb{G}_1\|+\|\mathbb{Z}^*_{N^2}\|+\|id\|+\|t\|$	$\|\mathbb{G}_1\|+\|\mathbb{Z}^*_{N^2}\|+\|id\|+\|t\|$
Shen et al. [22]	t_{P_e}+2$t_{\mathbb{Z}}$+t_{ecc}	2nt_p+t_{ecc}	3t_p+t_{P_d}+2$t_{\mathbb{Z}}$	$3\|\mathbb{G}_1\|+\|\mathbb{Z}^*_{N^2}\|+\|t\|$	$3\|\mathbb{G}_1\|+\|\mathbb{Z}^*_{N^2}\|+\|t\|$
Chen et al. [23]	t_p+$t_{\mathbb{G}_T}$+t_{ecc}	2nt_p+t_{ecc}	3t_p+t_{log}	$\|\mathbb{G}_1\|+\|\mathbb{G}_T\|+\|id\|+\|t\|$	$\|\mathbb{G}_1\|+\|\mathbb{G}_T\|+\|id\|+\|t\|$
Gai et al. [25]	3t_{add}+t_{mul}	(4t_{add}+3nt_{mul})$\|X\|$	t_{add}+t_{mul}	$\|id\|+\|d\|$	$\|d\|+\|\phi(X)\|$
Ming et al. [28]	2.2t_{ecc}+3t_{m_N}	(0.1n+2)t_{ecc}	2(t_{ecc}+t_{mod})	$1.1(\|\mathbb{G}_1\|+\|\mathbb{Z}_N\|+\|\mathbb{Z}^*_q\|+\|t\|)+\|id\|$	$\|\mathbb{G}_1\|+\|\mathbb{Z}_N\|+\|\mathbb{Z}^*_q\|+\|id\|+\|t\|$
Zhang et al. [29]	2t_{ecc}	(n+2)t_{ecc}	4t_{ecc}	$2\|\mathbb{G}_1\|+\|\mathbb{Z}^*_q\|+\|id\|+\|t\|$	$2\|\mathbb{G}_1\|+\|\mathbb{Z}^*_q\|+\|id\|+\|t\|$
Chen et al. [31]	t_{add}+2t_{mod} +t_{mul}+$t_{\mathbb{Z}_{N_2^*}}$	(n+1)t_{add}+n(t_{mul} +$t_{\mathbb{Z}_{N_2^*}}$)+t_{mod}	n(t_{add}+t_{mul})	$\|\mathbb{Z}^*_{N^2}\|+\|id\|+\|d\|+\|t\|$	$\|d\|+\|t\|$
Our scheme	t_{add}+t_{s_g}	nt_{add}	t_{s_r}+t_{add}	$4\|ind\|+\|d\|+\|t\|+\|H\|$	$4\|ind\|+\|d\|+\|t\|+\|H\|$

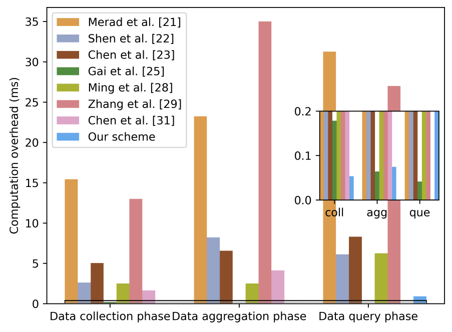


Fig. 6. Comparison of computation overhead

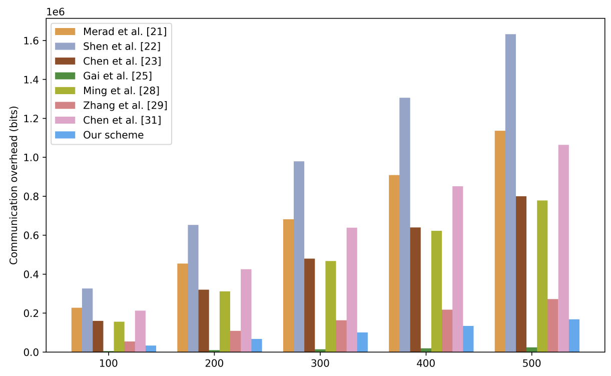


Fig. 7. Comparisons of communication overhead of DS-to-GW

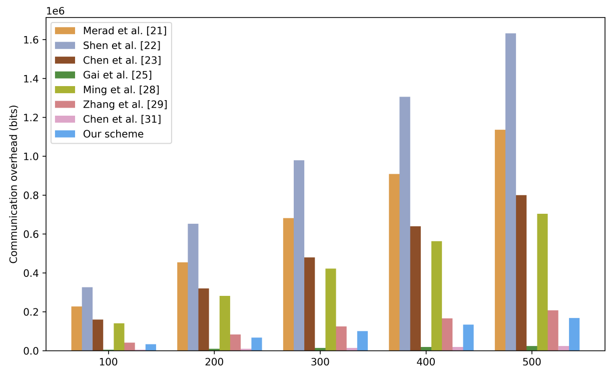


Fig. 8. Comparisons of communication overhead of GW-to-BC

Among them, the main communication operations are shown that $|\mathbb{G}_1|$: denotes the length of $\mathbb{G}_1$ group, is 1024 bits; $|\mathbb{G}_T|$: denotes the length of $\mathbb{G}_T$ group, is 1024 bits; $|\mathbb{Z}^*_q|$: denotes the length of $\mathbb{Z}^*_q$ group, is 160 bits; $|\mathbb{Z}^*_q|$: denotes the length of $\mathbb{Z}^*_q$ group, is 160 bits; $|\mathbb{Z}_N|$: denotes the length of $\mathbb{Z}_N$ group, is 1024 bits; $|\mathbb{Z}_{N^2}|$: denotes the length of $\mathbb{Z}_{N^2}$ group, is 2048 bits; $|H|$: denotes the length of hash value, is 256 bits; $|id|$: denotes the length of identity id, is 32 bits; $|t|$: denotes the length of identity id, is 32 bits; $|id|$: denotes the length of time stamp, is 32 bits; $|ind|$: denotes the length of index, is 8 bits; $|d|$: denotes the length of data, is 16 bits; $|\phi(X)|$: denotes the length of the estimated frequency value for each element, is 32 bits. The comparison of communication overhead between the proposed scheme and the existing schemes is shown in TABLE III.

Fig. 7 shows that the comparison of communication overhead from DS to GW in the 8 schemes, where the communication overhead increases with the increase of the DS number n. In the proposed scheme, DS sends the message $\{i, j, l, t_{cur}, rd, sh_l(d'_{ij}), mac^l_{ij}\}$ to GW. Therefore, the communication overhead is (4$|ind|$+$|d|$+$|t|$+$|H|$) bits =(4*8+16+32+256) bits = 336 bits.

Fig. 8 shows the comparison of communication overhead from GW to BC in the 8 schemes, where the communication overhead increases with the increase of the GW number m. In the proposed scheme, GW sends the aggregated message $\{j, l, v, t_{cur}, rd, Sh_l(j), mac^v_{jl}\}$ to BC. Therefore, the communication overhead is (4$|ind|$+$|d|$+$|t|$+$|H|$) bits =(4*8+16+32+256) bits = 336 bits.

D. Time latency and throughput

In this section, we use the Tape tool to test the performance of our scheme in the BC network. The results are shown in TABLE IV, Fig. 9 and Fig. 10. In TABLE IV, time latency (ms): denotes the time point from the start of the test to the reception of each block; Block: denotes the received block number; throughput: denotes that each block contains 50 transactions. TABLE IV shows that 1000 upload transactions and query transactions are sent to BC in increments of 200 transactions each to observe the time latency and throughput in BC. That's because the upload transactions involves the consensus mechanisms, block generation, etc., and consumes more resources. However, the query transactions is relatively simple, only needs to read the data that has been written and does not require complex operations. As the amount of sent transactions increases, the throughput of two types transaction also increases steadily. Among them, the overall trend of query transactions is smaller than that of upload transactions.

TABLE IV
TIME LATENCY AND THROUGHPUT

Time Latency (ms)		Block	Throughput
upload	query		
0	0	0	0
0.68	2.98	2	100
0.96	5.68	4	200
1.25	8.75	6	300
1.35	11.57	8	400
1.55	14.47	10	500
1.67	17.48	12	600
1.94	20.58	14	700
2.00	23.80	16	800
2.24	27.00	18	900
2.36	29.74	20	1000

Fig. 9 and Fig. 10 show respectively the performance of upload and query transactions. The delay fluctuation range is 0.5-4.6ms and 26-32.8ms respectively and the average delay is 2.36ms and 29.74ms, in which the delay of query transactions is greater than that of upload transactions. Meanwhile, the delay of two kinds of transactions decreases as the number of transaction sent increases and is inversely proportional to the throughput. Experiments show that the network performance of the proposed scheme is efficient.

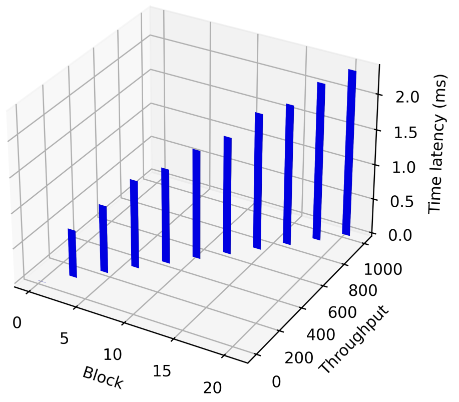


Fig. 9. The performance of upload transactions

VIII. CONCLUSION

In this work, we proposed a lightweight and privacy-preserving distributed multidimensional data trend query scheme with fault-tolerant for MaaS. The proposed scheme combines the advantages of LDP and the Shamir secret sharing scheme to address the secure data sharing in MaaS. Moreover, our scheme not only verifies the data integrity, but also reduces the computation overhead. Furthermore, we utilize BC as CS to solve problems such as single point attacks. By deploying a Hyperledger Fabric platform, we constructed a distributed structure that no longer relies on trusted third party, as compared to the traditional centralized structure. A comprehensive analysis in terms of security and evaluation prove

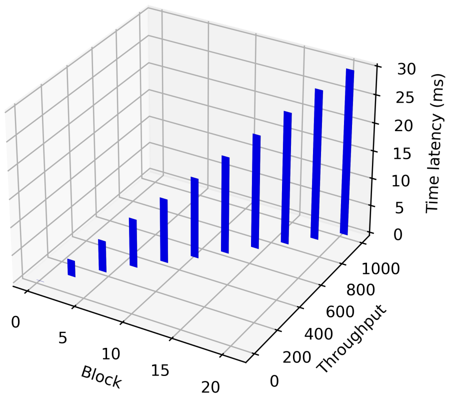


Fig. 10. The performance of query transactions

that the proposed scheme has significant advantages compared to existing schemes. To further improve the practicality of this scheme, we will carry out the following potential future work related to privacy preserving: strengthening privacy preserving mechanism, reducing costs, enhancing the resistance to attacks, and more.

REFERENCES

[1] X. Zheng and Z. Cai, Privacy-preserved data sharing towards multiple parties in industrial iots, IEEE journal on selected areas in communications, vol. 38, no. 5, pp. 968979, 2020.
[2] Islam M, Rehmani M H, Chen J. Differential privacy-based permissioned blockchain for private data sharing in industrial IoT[C]//Broadband Communications, Networks, and Systems: 12th EAI International Conference, BROADNETS 2021, Virtual Event, October 2829, 2021, Proceedings 12. Springer International Publishing, 2022: 77-91.
[3] Cui Tao. What is Internet of Things as a Service? [J]. Computer and Network, 2020, 46(06): 43.
[4] Tran V H, Lenssens B, Kassab A, et al. Machineasaservice: Blockchain-based management and maintenance of industrial appliances[J]. Engineering Reports, 2023, 5(7): e12567.
[5] "AI4I 2020 Predictive Maintenance Dataset." UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C5HS5C.
[6] Fan M, Zhang X. Consortium blockchain based data aggregation and regulation mechanism for smart grid[J]. IEEE Access, 2019, 7: 35929-35940.
[7] Shang S, Li X, Gu K, et al. A robust privacy-preserving data aggregation scheme for edge-supported iiot[J]. IEEE Transactions on Industrial Informatics, 2023, 20(3): 4305-4316.
[8] Tan Z, Cao F, Liu X, et al. LPPMM-DA: Lightweight Privacy-Preserving Multi-Dimensional and Multi-Subset Data Aggregation for Smart Grid[J]. IEEE Transactions on Smart Grid, 2024.
[9] Shen G, Xiao K, Tu J, et al. A privacy-preserving and robust aggregation scheme for multi-dimensional data in VANETs[J]. Computers and Electrical Engineering, 2025, 123: 110145.
[10] Zhao S, Xu S, Han S, et al. PPMM-DA: Privacy-preserving multidimensional and multisubset data aggregation with differential privacy for fog-based smart grids[J]. IEEE Internet of Things Journal, 2023, 11(4): 6096-6110.
[11] Song Z, Zhou T, Zhong W, et al. Fault-tolerant data aggregation scheme supporting fine-grained linear operation in smart grid[J]. IEEE Access, 2023, 11: 68525-68537.
[12] Lyu L, Nandakumar K, Rubinstein B, et al. PPFA: Privacy preserving fog-enabled aggregation in smart grid[J]. IEEE Transactions on Industrial Informatics, 2018, 14(8): 3733-3744.
[13] Bi M, Wang Y, Cai Z, et al. A privacy-preserving mechanism based on local differential privacy in edge computing[J]. China communications, 2020, 17(9): 50-65.
[14] Zhao C, Wang L, Liu Z, et al. BPRM: Blockchain-Based Privacy-Preserving and Robust Data Aggregation Supporting Multi-Functionality for Fog-Assisted Smart Grid[J]. IEEE Internet of Things Journal, 2024.

[15] Tong X, Hamzei M, Jafari N. Towards Secure and Efficient Data Aggregation in BlockchainDriven IoT Environments: A Comprehensive and Systematic Study[J]. Transactions on Emerging Telecommunications Technologies, 2025, 36(2): e70061.
[16] Abbas M M, MeradBoudia O R, Senouci S M, et al. Blockchainbased secure multifunctional data aggregation for fogIoT environments[J]. Concurrency and Computation: Practice and Experience, 2024, 36(21): e8212.
[17] Tao Y, Gilad A, Machanavajjhala A, et al. Differentially private explanations for aggregate query answers[J]. The VLDB Journal, 2025, 34(2): 20.
[18] Guan Z, Zhang Y, Wu L, et al. APPA: An anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT[J]. Journal of Network and Computer Applications, 2019, 125: 82-92.
[19] Gao R, Xue Y, Wang W, et al. Improved Scheme for Data Aggregation of Distributed Oracle for Intelligent Internet of Things[J]. Sensors, 2024, 24(17): 5625.
[20] Su Y, Li Y, Li J, et al. LCEDA: Lightweight and communication-efficient data aggregation scheme for smart grid[J]. IEEE Internet of Things Journal, 2021, 8(20): 15639-15648.
[21] Merad-Boudia O R, Senouci S M. An efficient and secure multidimensional data aggregation for fog-computing-based smart grid[J]. IEEE Internet of Things Journal, 2020, 8(8): 6143-6153.
[22] Shen H, Liu Y, Xia Z, et al. An efficient aggregation scheme resisting on malicious data mining attacks for smart grid[J]. Information Sciences, 2020, 526: 289-300.
[23] Chen Y, Martínez-Ortega J F, Castillejo P, et al. An elliptic curve-based scalable data aggregation scheme for smart grid[J]. IEEE Systems Journal, 2019, 14(2): 2066-2077.
[24] Peng C, Luo M, Wang H, et al. An efficient privacy-preserving aggregation scheme for multidimensional data in IoT[J]. IEEE Internet of Things Journal, 2021, 9(1): 589-600.
[25] Gai N, Xue K, Zhu B, et al. An efficient data aggregation scheme with local differential privacy in smart grid[J]. Digital Communications and Networks, 2022, 8(3): 333-342.
[26] Jing Z, Zhu Y, Zhao Q, et al. Equipment failure data trends focused privacy preserving scheme for Machine-as-a-Service[J]. Journal of Information Security and Applications, 2025, 89: 104000.
[27] Wang X, Liu Y, Choo K K R. Fault-tolerant multisubset aggregation scheme for smart grid[J]. IEEE Transactions on Industrial Informatics, 2020, 17(6): 4065-4072.
[28] Ming Y, Li Y, Zhao Y, et al. Efficient PrivacyPreserving Data Aggregation Scheme with Fault Tolerance in Smart Grid[J]. Security and Communication Networks, 2022, 2022(1): 5895176.
[29] Zhang S, Chang J, Wang B. A multidimensional data aggregation scheme of smart home in microgrid with fault tolerance and billing for demand resonse[J]. IEEE Systems Journal, 2023, 17(3): 4639-4649.
[30] Li K, Yang Y, Wang S, et al. A lightweight privacy-preserving and sharing scheme with dual-blockchain for intelligent pricing system of smart grid[J]. Computers Security, 2021, 103: 102189.
[31] CHEN Jianwei,WANG Shuyu,ZHANG Meiping, et al. Blockchain-based and verifiable multidimensional data aggregation and sharing scheme for smart grid[J]. Journal on Communications, 2024, 45(01): 167-179.
[32] Wang Y, Luo F, Dong Z, et al. Distributed meter data aggregation framework based on Blockchain and homomorphic encryption[J]. IET CyberPhysical Systems: Theory Applications, 2019, 4(1): 30-37.
[33] Guo C, Jiang X, Choo K K R, et al. Lightweight privacy preserving data aggregation with batch verification for smart grid[J]. Future Generation Computer Systems, 2020, 112: 512-523.
[34] Zhang Y, Chen J, Zhou H, et al. A privacy-preserving data aggregation scheme with efficient batch verification in smart grid[J]. KSII Transactions on Internet and Information Systems (TIIS), 2021, 15(2): 617-636.
[35] ZHANG Yue,ZHU Youwen,ZHOU Yuqian, et al. Mean Estimation Mechanisms under (ε,δ)-Local Differential Privacy[J]. Journal of Electronics Information Technology, 2023, 45(03): 765-774.
[36] Shamir A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613.
[37] G. R. Blakley, Safeguarding cryptographic keys, in Afips, 1979.
[38] He D, Kumar N, Lee J H. Privacy-preserving data aggregation scheme against internal attackers in smart grids[J]. Wireless Networks, 2016, 22: 491-502.