Future Generation Computer Systems 131 (2022) 28–42

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

PANDA: Lightweight non-interactive privacy-preserving data aggregation for constrained devices

Mei Wang [a], Kun He [a,*], Jing Chen [a,b,**], Ruiying Du [a,c], Bingsheng Zhang [d], Zengpeng Li [e]

[a] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[b] Rizhao Institute of Information Technology, Wuhan University, Rizhao, 276800, China
[c] Collaborative Innovation Center of Geospatial Technology, Wuhan, 430079, China
[d] School of Cyber Science and Technology, Zhejiang University, Hangzhou 310007, China
[e] School of Cyber Science and Technology, Shandong University, Qingdao 266237, China

ARTICLE INFO

Article history:
Received 29 August 2021
Received in revised form 3 December 2021
Accepted 17 January 2022
Available online 24 January 2022

Keywords:
Privacy-preserving data aggregation
Trusted Execution Environment

ABSTRACT

Privacy-preserving data aggregation is becoming a demanding necessity for many promising scenarios, e.g., health care analysis. Sensitive data are collected and aggregated in a privacy-preserving approach using current Internet of Things (IoT) technology, leading to immense challenge and consequent interest in developing secure computing algorithms for individual and organizational data. However, most existing solutions focus on specific evaluations (e.g., SUM), and they use heavy cryptographic techniques, which are far from practice for constrained IoT devices. The Trusted Execution Environment (TEE, implemented with Intel SGX) can assist in computing arbitrary functions and avoiding resource-consuming operations. Nevertheless, TEE is subject to several challenges because TEE is vulnerable to limited resource and even function violations, e.g., the attacker may bypass the boundary of TEE. In this paper, we propose a lightweight non-interactive privacy-preserving data aggregation scheme for resource-constrained devices, named PANDA, where TEE is introduced to bypass the trusted entities requirement and heavy overhead. Additionally, PANDA explores the certificate-aided function authorization to prevent leakage from unauthorized functions, and designs the public verifiable certificate management to detect the abnormal behaviors of the host to mitigate the external host compromise. We formalize PANDA with rigorous security analysis to indicate privacy protection against the compromised aggregator and analyst. The evaluation results show that PANDA has constant online communication cost and lightweight computation overhead for constrained devices, which is suitable for IoT applications.

1. Introduction

With the wise combination of cloud computing and the Internet of Things (IoT), cloud-enabled IoT has become an irresistible force to change the model of production and lifestyle. Smart city, health care, virtual reality, etc., these sophisticated IoT applications demand more from the computing power, storage capacity, and battery capacity of IoT terminals. The main superiority of cloud-enabled IoT is to carry out the resource-consuming tasks on a cloud server rather than on the resource-limited IoT terminals, which grows the functionality scope and eases pressure on IoT devices. For example, in the health care analysis systems, the emerging wearable devices and applications can detect physical conditions from heart rate to oxygen levels, which provide abundant data for advanced medical development. The users' health data are transmitted to the cloud server (aggregator), which attempts to compute an aggregation function (e.g., SUM, MAX, or VAR) on these data (Fig. 1).

However, secure data aggregation in a privacy-preserving approach is still a demanding task. It arouses practical concerns since the trusted cloud server is hard to implement in the real world. Thus, the concept of *Privacy-preserving Data Aggregation (PDA)* emerged to solve this problem. PDAs refer to the processes of selecting and analyzing relevant data to get the desired results for certain purposes, which have been widely studied in various scenarios, such as mobile sensing [1,2], fog computing [3–5], wireless sensor networks [6–8], and machine learning [9,10]. Nevertheless, most of them do not apply to IoT scenarios due to the high computation overhead and frequent interactions that

* Corresponding author.
** Corresponding author at: Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China.
E-mail addresses: hekun@whu.edu.cn (K. He), chenjing@whu.edu.cn (J. Chen).

https://doi.org/10.1016/j.future.2022.01.007
0167-739X/

Table 1
Comparison with previous works.

Scheme	Techniques	No trusted aggregator	Non-interactive	Arbitrary functions	Constant online communication
Groat–He–Forrest [6]	Adding Camouflage values	✓	×	×	×
Jung et al. [11]	Multi-variate polynomial evaluation	✓	✓	×	×
Zhang–Chen–Zhong [1]	Bitwise-XOR HE	×	✓	✓	×
Mohassel–Zhang [10]	Secret share + MPC	✓	✓	×	×
Bonawitz et al. [9]	MPC	✓	✓	×	×
Gong et al. [12]	Unique sequence number	✓	×	✓	×
Guo–Tian–Cho et al. [4]	Symmetric HE	×	✓	✓	×
Zhao et al. [5]	Somewhat HE	✓	✓	×	×
PANDA	Authenticated encryption + TEE	✓	✓	✓	✓

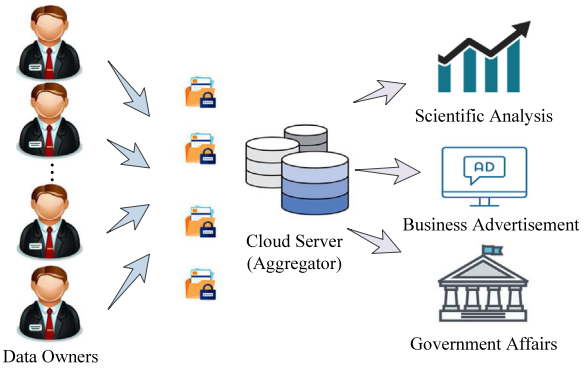


Fig. 1. The diagram of data aggregation. Data owners send their data to the aggregator in a secure manner, e.g., encryption. The aggregator provides different aggregation services for various applications, e.g., scientific analysis, business advertisement, and government affairs.

are relatively expensive for constrained IoT devices, and existing PDAs that focus on specific functions cannot satisfy the diversity demand of smart services.

1.1. Motivations

Depending on the interaction between data owners and the aggregator, existing PDA solutions can be classified into two categories: the *interactive* and the *non-interactive* schemes.

Interactive solutions usually distribute the process to end devices, in which the data are maintained and manipulated by data owners, who interact with the aggregator repeatedly. The most representative interactive solutions are *federated learning* based schemes [9,13–15] and *secure Multi-Party Computation (MPC)* based schemes [16–19]. In federated learning [20–22], a centralized model is generated by aggregating parameter updates trained locally on end devices. However, sensitive information still may be inferred from the parameter updates [9,13,15]. Similarly, MPC [23–26] is a subfield of cryptography that aims to jointly compute a function over participants' inputs while ensuring privacy, which is an evident approach for PDA. Unlike federated learning, MPC involves only the data owners, there is no need for a trusted aggregator. However, the interactivity of MPC makes it inefficient in PDA scenarios, which generally involve massive data.

Instead of interactive approaches, non-interactive solutions usually aggregate data by transmitting them to a center acquired to undertake the bulk of computations. No more frequent interactions are required between data owners and the aggregator. The most significant technique in non-interactive PDA solutions is *Homomorphic Encryption (HE)* [27,28] that is widely used for privacy-preserving outsourced computation [29–31]. Specifically, HE allows generating an encrypted result directly on the ciphertexts, corresponding to the result on the plain items. There have been many works [4,32–34] that addressed PDA by using HE due to its ciphertext-based operating capability. However, these solutions utilized computation on ciphertexts, which also introduced expensive cryptographic operations.

In addition, most existing PDA schemes [35,36] focus on specific functions (e.g., SUM), which means a specific protocol should be redeployed once the aggregation function is changed. To compute arbitrary functions, Zhang–Chen–Zhong [1] utilized a unique sequence number to enable the aggregator to obtain the exact distribution of the data aggregation. Gong et al. [12] improved [1] by introducing the Diffie–Hellman key exchange, which brought in additional interactions. Moreover, they ignored the fact that "arbitrary functions" may cause leakage from certain functions (e.g., $f(x) = x$ depicted in Section 6).

In a nutshell, it is still an open question that:

Is it possible to design a lightweight PDA solution, which can compute arbitrary functions on massive multi-source data with constrained IoT devices?

We answer this question in the affirmative and conclude our challenges and contributions as follows.

1.2. Challenges and contributions

Below, we sketch PANDA, a non-interactive PDA system deployed in the cloud-enabled IoT scenarios aiming to address this above-mentioned open question, along with associated technical explanations. Concretely, the cloud server is tailored with the Trusted Execution Environment (TEE, implemented with Intel SGX), an environment executing code with high trust. Note that introducing TEEs in PDA is not trivial since there are two challenges to be solved: (1) **Resource limitation**. IoT scenarios generally involve massive records across a large population of data owners. Given the situation of constrained terminals and the TEE with very limited storage, a lightweight function evaluation mechanism for PDA should be provided to remedy this challenge. (2) **Uncontrolled functions**. PANDA suffers the underlying vulnerability of function violations, which manifests in two problems. One is, the goal of PDAs for arbitrary functions raises the vulnerability of leakage from functions that data owners do not authorize. The attacker may maliciously initiate a task of computing a problematic function to infer sensitive information, which cannot be mitigated only by the TEE. The other problem is, TEE provides a secure, integrity-protected processing environment, but the compromised host may bypass the boundary of TEE via polluting the aggregation function input. For instance, the host may tamper with the function (i.e., program code), or replace it with another one (commonly included as *replay attacks*).

We give the comparison with previous works in Table 1, and conclude our main contributions as follows.

- To the best of our knowledge, it is the first work to propose a non-interactive PDA model in the cloud-enabled IoT scenarios, which can compute arbitrary functions and requires no trusted entities. We introduce a cloud server as an aggregator equipped with the TEE to provide PDA services. The constant online communication and low computation cost (data owners) make sense in reality, especially in constrained environments, e.g., the IoT environment.
- To mitigate the resource limitation, we propose a *Privacy-preserving Function Evaluation* (PFE) scheme to compose a lightweight PDA solution. We exploit the TEEs to facilitate the PFE and achieve compact key management. PFE guarantees the security and privacy of sensitive data, and involves mostly lightweight symmetric operations.
- We avoid uncontrolled functions by employing function authorization and verification approach. Particularly, we design the *Certificate-aided Function Authorization* (CFA) mechanism with the public verifiable certificate management. The data owners serve as the authority to approve the requested function, which eliminates privacy leakage caused by unauthorized functions. The certificates are recorded with the public ledger, where any tampering with the requested aggregation function can be detected.
- We present PANDA, a lightweight non-interactive PDA system based on our model. We prove that PANDA is secure by giving security definitions and analysis. We also show that PANDA is practical via both theoretical and experimental evaluation.

1.3. Organization

We show the basic notation and main background knowledge in Section 2. Section 3 gives the system architecture, goals, and the threat model. We propose PANDA in Section 4, present our design of function evaluation and key management in Section 5. Section 6 depicts the function authorization and verification mechanism. We give the security analysis and evaluation in Sections 7 and 8, respectively. Finally, Section 9 reviews the related work, and we conclude this paper in Section 10.

2. Preliminaries

2.1. Notation

The frequently used notations are summarized in Table 2.

Table 2
Notations.

Notation	Description
$n \in \mathbb{N}$, $[n]$	The number of data owners, the set $\{1, \ldots, n\}$
$\mathcal{P}_i$, $\mathcal{P}_0$	The ith data owner with id $ID_i (i \in [n])$, the analyst
$\mathcal{CS}$	The cloud server equipped with $\mathcal{TEE}_{\mathcal{CS}}$
$\mathcal{KDC}$	The key distribution center equipped with $\mathcal{TEE}_{\mathcal{KDC}}$
$\mathcal{PL}$	The public ledger
$\mathcal{D}$, d	The plain data family and the bit length of its element
$\mathcal{R}$	The result data family
$\mathbf{v}_i$	$\mathbf{v}_i = \{v_{i,1}, \ldots, v_{i,m}\}$ is an m-dimensional vector possessed by $\mathcal{P}_i$, where each item $v_{i,j} \in \mathcal{D}$
$\{\mathbf{v}_i\}$	The vector set $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$
$\{\mathbf{v}_i\}/\mathbf{v}_t$	The vector set $\{\mathbf{v}_1, \ldots, \mathbf{v}_{t-1}, \mathbf{v}_{t+1}, \ldots, \mathbf{v}_n\}$, $t \in [n]$
$\{\mathbf{c}_i\}$	The encoded vector set $\{\mathbf{c_1}, \ldots, \mathbf{c_n}\}$ for $\{\mathbf{v}_i\}$
$\{CERT_i\}$	The certificate set $\{CERT_1, \ldots, CERT_n\}$
$\mathcal{F}$, $\lvert\mathcal{F}\rvert$	The aggregation function family, the size of $\mathcal{F}$
$f \in \mathcal{F}$	An arbitrary aggregation function
W	The function whitelist
$\|$	The symbol of concatenation
λ	The security parameter

2.2. Authenticated encryption

Authenticated encryption [37] is a particular form of symmetric encryption which provides confidentiality and integrity guarantees simultaneously. An authenticated encryption scheme AE consists of three algorithms: AE.Gen, AE.Enc, and AE.Dec. The key generation algorithm AE.Gen takes the security parameter 1^λ as input and outputs a private key k^{AE}. The encryption algorithm AE.Enc takes the key k^{AE} and a message m as input, then outputs a ciphertext c. The decryption algorithm AE.Dec takes k^{AE} and c as input, then outputs the original message m or an error symbol $\perp$. AE should provide both correctness and security. The correctness means that for all keys k^{AE} and all messages m, $\mathsf{AE.Dec}(k^{\mathsf{AE}}, \mathsf{AE.Enc}(k^{\mathsf{AE}}, m)) = m$. We require AE to satisfy the security property as follows. For any Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$ that is given several ciphertexts of its chosen messages encrypted under a randomly generated key k^{AE} (where k^{AE} is invisible to $\mathcal{A}$), $\mathcal{A}$ can distinguish between two newly generated ciphertexts under k^{AE} with negligible probability. Furthermore, the advantage is negligible that $\mathcal{A}$ forges a new valid ciphertext (not included in the ones it received) under the same k^{AE}. In this paper, we employ a desired authenticated encryption scheme AE via generic *encrypt-then-authenticate* approach (e.g., used in SSL/TLS [38]), combining a private-key encryption scheme DE with a message authentication code scheme MA (details are presented in Section 5).

2.3. Trusted execution environment

A *Trusted Execution Environment (TEE)* is a secure area of the processor, isolating from its surrounding environment, which can defend against threats outside the TEE, e.g., the host machine attempts to learn the secret information of the program running in the TEE. There are several TEE instances, including Intel Software Guard Extensions (SGX) [39], ARM TrustZone [40], and AMD Secure Encrypted Virtualization (SEV) [41]. In this paper, we implement TEEs by the Intel SGX, which is a set of processor extensions to Intel x86. A TEE built by the Intel SGX has three main functionalities this paper depends on: *Isolation*, *Sealing*, and *Attestation*. *Isolation* indicates that the data and code running inside the TEE cannot be read or modified by any other external process. *Sealing* means that all data transmitted to the host machine are sealed with a secret key embedded within the TEE. *Attestation* implies that the code, data, metadata, and outputs of the program within a TEE can be attested by generating an unforgeable report with a specific signing key and instructions.

2.4. Public ledger

The public ledger was named from the traditional accounting system that records transaction information like commodity names, quantities, and prices. Such record-keeping mechanisms facilitate general public query, verification, as well as supervision. In general, most applications currently employ a centralized architecture that depends on a central authority to authenticate, validate, or process transactions, which is at risk of the single point of failure problem. A public ledger is decentralized to avoid the dictatorship of the central authority, and all records are written in the ledger only when the related parties reach a consensus. With the enormous popularity of cryptocurrency-based blockchain systems, which can also be generalized as a specific type of public ledger, the applications of the public ledger are favored by academia, business, and industry. All items kept in the public ledger are given timestamps and signed uniquely, which can be viewed and verified by any entity involved in this ledger. The public verifiability property can naturally be used to mitigate the malicious operations to TEEs (e.g., [42,43]). However,

a vital observation of the public ledger is that the entire state of the ledger must be exposed for public verification, and the public ledger usually has very limited computing power. In the data aggregation scenario, the vast amounts of data burden on the process, let alone computing via the public ledger. Thus, we utilize public ledger only to process the validity of the aggregation function (Section 6.4), to avoid inefficient massive data processing.

3. System architecture and model

3.1. System architecture

As described in Section 1, we are motivated to propose a non-interactive PDA model for constrained devices in the cloud-enabled IoT scenarios. Specifically, there are five types of roles (also expressed as entities) in our design: the *cloud server* $\mathcal{CS}$, the *analyst* $\mathcal{P}_0$, the *data owners* $\mathcal{P}_i$ ($i \in [n]$), the *key distribution center* $\mathcal{KDC}$, and the *public ledger* $\mathcal{PL}$, where both $\mathcal{CS}$ and $\mathcal{KDC}$ are equipped with secure TEE $\mathcal{TEE}_{\mathcal{CS}}$ and $\mathcal{TEE}_{\mathcal{KDC}}$, respectively. Both the analyst and the data owners are set as constrained terminals with limited resources. The relationship between the five kinds of entities in our system is shown in Fig. 2.

Each data owner $\mathcal{P}_i$ holds a private m-dimensional vector $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,m})$, it instantly stores the masked versions of $\mathbf{v}_i$ on the remote cloud server $\mathcal{CS}$ to reduce the operation and storage costs (any number of vectors are allowed actually, here we assume each data owner has only one vector for convenience). When the analyst $\mathcal{P}_0$ demands for an aggregation result from the multi-sourced data held by some data owners, $\mathcal{P}_0$ outsources the complex computation task to the cloud, i.e., computing aggregation function f (e.g., SUM, MAX, or VAR), on the masked data stored in the database of the $\mathcal{CS}$. The key distribution center $\mathcal{KDC}$ is introduced to help the constrained terminals manage the keys, and the public ledger $\mathcal{PL}$ is employed to facilitate the public verification for function authorization.

Remark 1. To accord with the real-life scenarios closely, we set five roles as above in this work. In essence, the analyst can be merged with the cloud server as an aggregator. That is, the cloud server is entitled to initiate an aggregation task. Besides, any data owner can also issue an aggregation request as the analyst, in addition to supervising with their data.

3.2. Goals

Our design goal is to develop a lightweight PDA system for constrained devices with the following properties.

- **No Trusted Aggregator**: We claim that the PDA scheme requires no trusted aggregator because a totally trusted aggregator is challenging to find in practice.
- **Arbitrary Functions**: This PDA system supports arbitrary functions, which means it can compute any function no matter linear or non-linear.
- **Data-Owner-Specified Functions**: The functions should be authorized by data owners before aggregation to prevent leakage from functions.
- **No Heavy Overhead**: Both the interaction overhead between entities and the data owners' computation overhead are low. In addition, data owners do not need to be online all the time during the aggregation.

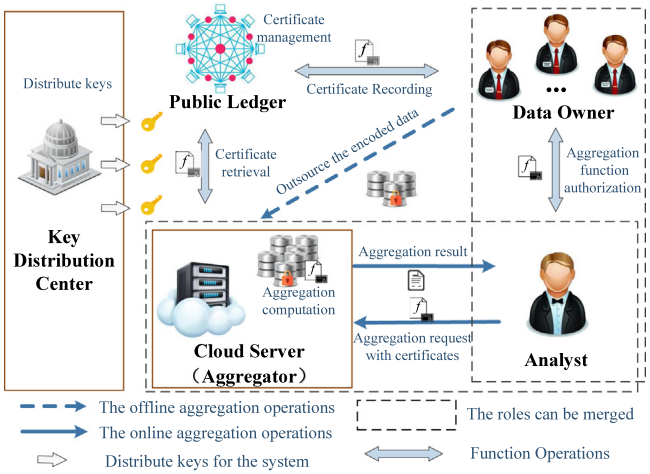


Fig. 2. The system architecture.

3.3. Threat model

In this paper, we require the privacy of any data owner should be protected in the case of the following threats.

- The cloud server may be compromised. It may try to learn more information beyond the allowed scope of protocols, change its inputs or outputs, even tamper with the program code outside the TEE.
- The analyst may be compromised. It can maliciously make an arbitrary function request, deviate from the protocol, or make replay attacks.
- The corruption to a minority of entities is acceptable. The compromised cloud server may collude with the analyst, or some data owners (the attacker has access to the corrupted data owners' plaintexts). Beyond that, the analyst may collude with any data owner.
- We assume that the key distribution center sets up the system as instructed initially, and the decentralized public ledger does not allow the majority collusion.

Remark 2. In this threat model, we only require the key distribution center is initialized securely, which is feasible. After that, the sensitive operations are confined in the TEE, and no honesty demands on the key distribution center any longer. In reality, the assumption that no majority collusion attack for the public ledger is reasonable, e.g., given the enormous size of the blockchain, a so-called 51% attack is almost certainly not worth the effort and likely impossible.

4. PANDA: Practical and Non-interactive Privacy-preserving Data Aggregation

4.1. Overview

We intend to introduce a cloud server armed with the TEE as the aggregator to charge in aggregation privately. Specifically, we propose PANDA, a lightweight non-interactive PDA system in the cloud-enabled IoT paradigm. PANDA is naturally split into four phases as Fig. 3: (1) the *initialization phase* sets the preparation up, mainly generating the system parameters and keys; (2) in the *vector encoding* phase, each data owner encodes and sends their vectors to the cloud server; (3) in the *function authorization* phase, each data owner generates a certificate to authorize the data-owner-specified function launched by the analyst, and the

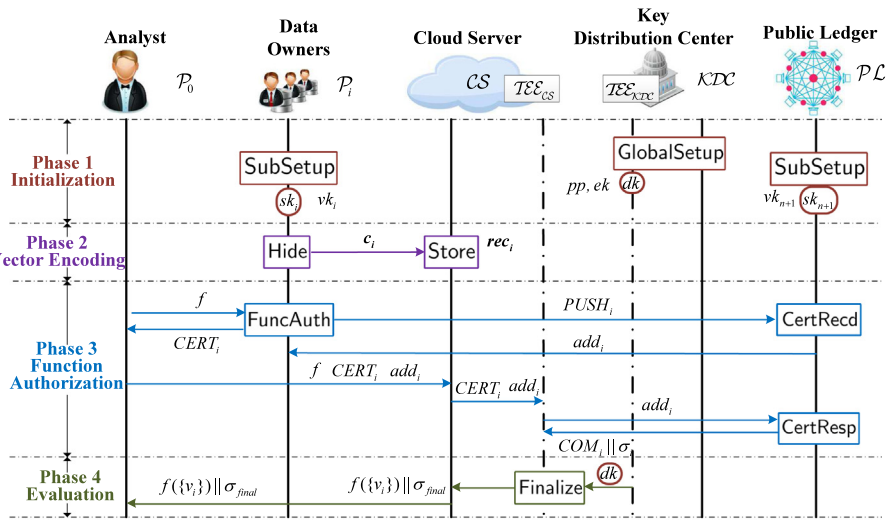


Fig. 3. The procedure of PANDA. The colored boxes represent the algorithms in PANDA, the colored circles represent the secret items stored locally by the corresponding entities, and the colored arrows represent the messages transmitted between entities.

certificates must be verified before evaluation; (4) in the final *evaluation* phase, the cloud server evaluates function f on the encoded vectors and outputs an aggregation result. Note that PANDA is designed in the online/offline mode. The first two phases work offline, and the other two phases usually run online. For clarity, PANDA is mainly composed of two underlying blocks: the *Privacy-preserving Function Evaluation* PFE (Section 5) and the *Certificate-aided Function Authorization* CFA (Section 6). PFE is designed as a lightweight function evaluation scheme to execute the aggregation in the TEE embedded on the cloud server with privacy and efficiency. CFA is proposed to avoid privacy leakage from functions, which is verified in the cloud server's TEE with the aid of the public ledger (Section 6.4). Only the functions that are verified valid via CFA can be executed in PFE. We define algorithms of non-interactive PDA in Section 4.2 and integrally depict the PANDA in Section 4.3.

4.2. Non-interactive privacy-preserving data aggregation

Definition 1 (*Non-interactive PDA*)**.** A non-interactive PDA scheme Π is a tuple of eight PPT algorithms (GlobalSetup, SubSetup, Hide, Store, FuncAuth, CertRecd, CertResp, Finalize) that satisfies the correctness requirement below.

Initialization Phase:

- GlobalSetup$(1^\lambda, n) \rightarrow (pp, (ek, dk))$: it takes the security parameter λ, number of data owners n as input, and outputs the system parameters pp and the encoding key pair (ek, dk). The public system parameters pp include all inputs to GlobalSetup algorithm and some other information if needed. This algorithm is run by $\mathcal{KDC}$. All entities receive (ek, pp). Only $\mathcal{TEE}_{\mathcal{KDC}}$ holds dk.
- SubSetup$(pp) \rightarrow (sk, vk)$: it takes the system parameters pp as input, and outputs the signature key pair (sk, vk). This algorithm is run by $\mathcal{P}_i$ $(i \in [n])$ and $\mathcal{PL}$. Each $\mathcal{P}_i$ gets (sk_i, vk_i) $(i \in [n])$, $\mathcal{PL}$ gets (sk_{n+1}, vk_{n+1}).

Vector Encoding Phase:

- Hide$(ek, \mathbf{v}_i) \rightarrow \mathbf{c}_i$: this algorithm is run by $\mathcal{P}_i$, it takes as input the encoding key ek, the data vector $\mathbf{v}_i$, outputs the encoded vector $\mathbf{c}_i$.
- Store$(ID_i, \mathbf{c}_i) \rightarrow \mathbf{rec}_{i,ID_i}$: this algorithm is run by $\mathcal{CS}$, it takes the data owner id ID_i as input, and outputs a record $\mathbf{rec}_{i,ID_i}$.

Function Authorization Phase:

- FuncAuth$(sk_i, W_i, f) \rightarrow (CERT_i, PUSH_i)/\perp$: this algorithm is run by $\mathcal{P}_i$, it takes as input the signing key sk_i, the function whitelist W_i, the requested function f, outputs the certificate $CERT_i$ and a corresponding $PUSH_i$, or $\perp$.
- CertRecd$(PUSH_i) \rightarrow add_i$: this algorithm is run by $\mathcal{PL}$, it takes as input the $PUSH_i$, and outputs the address add_i that $PUSH_i$ locates.
- CertResp$(add_i) \rightarrow (COM_i \parallel \sigma_i)$: this algorithm is run by $\mathcal{PL}$, it takes the address add_i as input, and outputs the item COM_i on add_i, and a proof σ_i.

Evaluation Phase:

- Finalize$(COM_i \parallel \sigma_i, CERT_i, \{\mathbf{rec}_i\}, dk) \rightarrow f(\{\mathbf{v}_i\} \parallel \sigma_{final}) / \perp$: this algorithm is run in $\mathcal{TEE}_{\mathcal{CS}}$ (via $\mathcal{CS}$), it takes as input the commitment COM_i appended with the proof σ_i, the certificate $CERT_i$, the encoded record set $\{\mathbf{rec}_i\}$ and the decoding secret key dk, outputs the evaluation result $f(\{\mathbf{v}_i\})$ appended with the proof σ_{final}, or $\perp$.

Correctness. For all $\lambda, n \in \mathbb{N}$, all $(pp, (ek, dk))$ generated by GlobalSetup$(1^\lambda, n)$, any (sk_i, vk_i) generated by $\mathcal{P}_i$ $(i \in [n+1]$, $\mathcal{PL}$ refers to $\mathcal{P}_{i+1})$, any data space $\mathcal{D}$ and vector set $\mathbf{v}_i$ derived from $\mathcal{D}$, any data owner id ID_i, any function family $\mathcal{F}$ and any function $f \in \mathcal{F}$, any function whitelist W_i, if

- $\mathbf{c}_i \leftarrow$ Hide$(ek, \mathbf{v}_i)$,
- $\mathbf{rec}_{i,ID_i} \leftarrow$ Store$(ID_i, \mathbf{c}_i)$,
- $(CERT_i, PUSH_i) \leftarrow$ FuncAuth(sk_i, W_i, f),
- $add_i \leftarrow$ CertRecd$(PUSH_i)$,
- $(COM_i \parallel \sigma_i) \leftarrow$ CertResp(add_i), then

Finalize$(COM_i \parallel \sigma_i, CERT_i, \{\mathbf{rec}_i\}, dk) = f(\{\mathbf{v}_i\}) \parallel \sigma_{final}$.

4.3. PANDA

PANDA combines the PFE scheme and the CFA scheme in a reasonably lightweight manner. The main procedure of PANDA is shown in Fig. 3, and a complete description of detailed PANDA is given in Fig. 4.

For the IoT scenarios, securing huge amounts of data is an intractable issue. PANDA is a non-interactive PDA system, where the processing of a huge quantity of data is accomplished in offline phases. The initialization phase consists of GlobalSetup and SubSetup, and the vector encoding phase is composed of Hide and Store. In addition, encoding the data involves only

Ingredients.

- A Privacy-preserving Function Evaluation scheme PFE := (PFE.Setup, PFE.CKeyGen, PFE.Encoding, PFE.Decoding, PFE.Eval).
- A Certificate-aided Function Authorization scheme CFA := (CFA.Setup, CFA.SKeyGen, CertGen, CertVrfy).
- A digital signature scheme Sn :=(Sn.Sign, Sn.Vrfy).
- A commitment scheme Commit.

GlobalSetup$(1^\lambda, n) \rightarrow (pp, (ek, dk))$.

- Run CFA.Setup to get fpp.
- Set $pp = (\lambda, n, \mathcal{D}, \mathcal{F}, fpp)$.
- Run PFE.Setup to get (ek, dk) in $\mathcal{TEE}_{\mathcal{KDC}}$.

SubSetup$(pp) \rightarrow (sk_i, vk_i)$.

- Run CFA.SKeyGen to get (sk_i, vk_i) $(i \in [n+1])$.

Hide$(ek, \mathbf{v}_i) \rightarrow \mathbf{c}_i$.

- Run PFE.Encoding$(ek, \mathbf{v}_i) \rightarrow \mathbf{c}_i$.

Store$(ID_i, \mathbf{c}_i) \rightarrow \mathbf{rec}_{i,ID_i}$.

- Set $\mathbf{rec}_{i,ID_i} = \mathbf{c}_i$.

FuncAuth$(sk_i, W_i, f) \rightarrow (CERT_i, PUSH_i)/\perp$.

- Run CFA.CertGen$(sk_i, W_i, f) \rightarrow CERT_i/\perp$.
- Parse a random r_i from $CERT_i$.
- Run Commit$(pp, CERT_i, r_i)$ to get COM_i.
- Set $PUSH_i = (ID_i, COM_i)$.

CertRecd$(PUSH_i) \rightarrow add_i$.

- Store $PUSH_i$ on add_i.

CertResp$(add_i) \rightarrow (COM_i || \sigma_i)$

- Find the $PUSH_i$ according to the address add_i.
- Parse $PUSH_i$ as (ID_i, COM_i).
- Run Sn.Sign to generate a signature σ_i for COM_i.

Finalize$(COM_i, \sigma_i, CERT_i, \mathbf{rec}_i, dk) \rightarrow (f(\mathbf{v}_i), \sigma_{final})/\perp$.

- Run Sn.Sign to generate a signature σ_i' for the COM_i. If $\sigma_i' \neq \sigma_i$, outputs $\perp$; otherwise, next step.
- Run Commit$(pp, CERT_i, r_i) \rightarrow COM_i'$. If $COM_i' \neq COM_i$, outputs $\perp$; otherwise, next step.
- Run CFA.CertVrfy$(vk_i, CERT_i, f) \rightarrow 1/0$. If 0, output $\perp$; otherwise, next step.
- Run PFE.Eval$(dk, f, \{\mathbf{rec}_i\})$ to get the evaluation result $f(\{\mathbf{v}_i\})$.
- Run Sn.Sign to generate a signature σ_{final} for $f(\{\mathbf{v}_i\})$.
- Output $f(\{\mathbf{v}_i\}) || \sigma_{final}$.

Fig. 4. A complete description of PANDA.

lightweight symmetric encryption, which is naturally suitable for constrained data owners with limited resources. Even though we employ an asymmetric signature scheme in online phases of PANDA, it is executed once by each entity for one function. The computation consumption is tiny and acceptable compared to the large-scaled data. Specifically, the function authorization phase involves only maximum operations for the $\mathcal{CS}$ and generating a certificate by each $\mathcal{P}_i$, which are both efficient. In addition, the interactions between the analyst $\mathcal{P}_0$ and the n data owners $\mathcal{P}_i$ can be executed in parallel. The bandwidth overhead is only the length of one query with a certificate in each interaction. Thus, PANDA makes minimal use of consuming operations, and its performance mainly benefits from the efficiency of the building blocks (Sections 5 and 6).

PANDA needs the aid of the public ledger serving as a public verifiable proof so that the privacy is maintained when behavior disorder is detected. We also introduce the signature scheme Sn and the commitment Commit (Section 6.4) to solidate PANDA. To preserve the secret key, a secure channel between $\mathcal{TEE}_{\mathcal{KDC}}$ and $\mathcal{TEE}_{\mathcal{CS}}$ is established after the remote attestation, which is naturally achieved by Intel SGX. Certainly, PANDA's performance is partly due to the TEE, in which masked data are decoded directly and then aggregated practically.

5. Function evaluation and key management

5.1. Technical statement

To obtain the accurate aggregation result while hiding the data against the compromised aggregator, we introduce an aggregator equipped with a TEE to achieve an efficient PDA. However, the resource limitation of constrained terminals proves to be the stumbling block to our tentative plan. To remedy this, we propose a *Privacy-preserving Function Evaluation* (PFE) scheme to compose a lightweight PDA solution. PFE guarantees the security and integrity of the data, and involves mostly lightweight symmetric operations.

However, when it comes to the symmetric encryption involving massive IoT users, how to exchange a secret key becomes an inevitable issue. It triggers two problems: (1) data aggregation usually involves a large number of data owners, implying that a large number of secret keys must be stored in the TEE, which only has very limited storage, and a large number of secure channels are required between data owners and the TEE to transmit the keys; (2) the TEE executing aggregation tasks is stateless that the secret keys will be erased when the processes finish. Thus, we introduce a key distribution center $\mathcal{KDC}$ equipped with a secure $\mathcal{TEE}_{\mathcal{KDC}}$ storing the secret key exclusively. $\mathcal{TEE}_{\mathcal{KDC}}$ regularly outputs reports to prove it is working properly, and will not be destroyed once it is built. We design the PFE with a customized key management policy.

5.2. Definitions of function evaluation and key management

The *Key Management Policy* and the *Privacy-preserving Function Evaluation* (PFE) are defined below.

Definition 2 (*Key Management Policy*)**.** The key pair (ek, dk) is generated in the $\mathcal{TEE}_{\mathcal{KDC}}$, where the public key ek is published to all parties by the $\mathcal{KDC}$, and the secret key dk is stored locally in $\mathcal{TEE}_{\mathcal{KDC}}$. The dk is transmitted to $\mathcal{TEE}_{\mathcal{CS}}$ via a secure channel, if and only if the remote attestation report from $\mathcal{TEE}_{\mathcal{CS}}$ is verified valid by $\mathcal{TEE}_{\mathcal{KDC}}$.

Definition 3 (*Privacy-preserving Function Evaluation*)**.** A Privacy-preserving Function Evaluation (PFE) scheme is a tuple of five PPT algorithms (Setup, CKeyGen, Encoding, Decoding, Eval) that satisfies the consistency property below.

- Setup$(1^\lambda) \rightarrow pp$: this setup algorithm is executed once by $\mathcal{KDC}$ to initialize the system. On input a security parameter 1^λ, this algorithm outputs the system parameters pp, which is published in public and received by all entities.

- $\mathsf{CKeyGen}(pp) \rightarrow (ek, dk)$: this encoding key generation algorithm is run in the $\mathcal{TEE}_{\mathcal{KDC}}$, on input the system parameters pp, generates the key pair (ek, dk), where the public key ek is published via the $\mathcal{KDC}$, the secret key dk is kept locally in $\mathcal{TEE}_{\mathcal{KDC}}$, and transmitted to $\mathcal{TEE}_{\mathcal{CS}}$ when the *key manage policy* in Definition 2 is satisfied.
- $\mathsf{Encoding}(ek, \mathbf{v}_i) \rightarrow \mathbf{c}_i$: this encoding algorithm is executed by data owners $\mathcal{P}_i$, on input the public key ek and plain vector $\mathbf{v}_i$, outputs the encoded vector $\mathbf{c}_i$ that is comprised of the ciphertext and MAC tag.
- $\mathsf{Decoding}(dk, \mathbf{c}_i) \rightarrow \mathbf{v}_i$: this decoding algorithm can be run by $\mathcal{P}_i$ or $\mathcal{TEE}_{\mathcal{CS}}$, on input the secret key dk and encoded vector $\mathbf{c}_i$, outputs the corresponding plain vector $\mathbf{v}_i$.
- $\mathsf{Eval}(dk, f, \{\mathbf{c}_i\}) \rightarrow f(\{\mathbf{v}_i\})$: this evaluation algorithm is executed in the $\mathcal{TEE}_{\mathcal{CS}}$, on input the secret key dk, the function f, and the encoded vector set $\{\mathbf{c}_i\}$, then this algorithm outputs the evaluation result $f(\{\mathbf{v}_i\})$.

Consistency. For all $\lambda \in \mathbb{N}$, any pp generated by $\mathsf{Setup}(1^\lambda)$, any (ek, dk) generated by $\mathsf{CKeyGen}(pp)$, any value $\mathbf{v}_i \in \{\mathbf{v}_i\}$ $(i \in [n])$ and any function $f \in \mathcal{F}$, if $\mathsf{Encoding}(ek, \{\mathbf{v}_i\}) = \{\mathbf{c}_i\}$, then we have $\mathsf{Decoding}(dk, \mathsf{Encoding}(ek, \mathbf{v}_i)) = \mathbf{v}_i$, and $\mathsf{Eval}(dk, f, \{\mathbf{c}_i\}) = f(\{\mathbf{v}_i\})$.

5.3. Our privacy-preserving function evaluation construction

We give an instantiated privacy-preserving function evaluation construction, and refer to this as PFE from here onwards. We employ the authentication encryption composing of the message authentication code [37] $\mathsf{MA} = (\mathsf{Mac}, \mathsf{Vrfy})$ and the private-key encryption $\mathsf{DE} = (\mathsf{Enc}, \mathsf{Dec})$ described in Fig. 5.

At this point, there are several findings about PFE. Firstly, the challenge of resource limitation has been solved. The symmetric encryption computation overhead is affordable for constrained data owners. Secondly, the input data are encrypted with authentication encryption, and the output can be protected with digital signature by $\mathcal{TEE}_{\mathcal{CS}}$, any modification can be detected easily. Thirdly, there are no complicated key agreements between the cloud server and the massive constrained data owners. Based on Definition 2, we conclude that dk is protected with the direct interactions between TEEs. No entity can touch dk from beginning to end. And only one key needs to be transmitted via a secure channel, such that all entities can generate ciphertexts that can only be decrypted in the TEE. Finally, the secret key dk can be kept in $\mathcal{TEE}_{\mathcal{KDC}}$ properly.

6. Certificate-aided function authorization and verification

6.1. Certificate-aided function authorization

This section addresses the challenge of uncontrolled functions. We observe that the information revealed in the aggregation process depends on the choice of functions to some extent. For instance, we assume an analyst retrieved some information via social engineering that someone $\mathcal{P}_i$ is suffering from heart disease. Then, if the analyst queries on the exact criteria, the patient's data may be revealed if an unauthorized function is executed, e.g., $f(x) = x$. Different from most previous works, to avoid the leakage from functions, we build the *Certificate-aided Function Authorization* mechanism, such that $\mathcal{P}_i$ decides whether functions can be performed.

Let λ be the security parameter, $\mathcal{G}$ be a polynomial-time algorithm that takes as input 1^λ and (except possibly with negligible probability) outputs a description of a cyclic group $\mathbb{G}$, whose order is q ($\|q\| = \lambda$) and g is the generator. PFE is constructed as follows.

- $\mathsf{Setup}(1^\lambda) \rightarrow pp$: on input a security parameter 1^λ, generate the system parameters $pp = (n, \mathcal{G}, \mathcal{D}, \mathcal{R}, \mathcal{F})$.
- $\mathsf{CKeyGen}(pp) \rightarrow (ek, dk)$: on input 1^λ, run $\mathcal{G}(1^\lambda)$ to generate $(\mathbb{G}, q, g)$; then choose a random $x \in \mathbb{Z}_q$ and set $h := g^x$; specify a function $H : \mathbb{G} \rightarrow \{0,1\}^{2\lambda}$; finally output the encoding key $ek = (\mathbb{G}, q, g, h, H)$ and the decoding key $dk = x$.
- $\mathsf{Encoding}(ek, \mathbf{v}_i) \rightarrow \mathbf{c}_i$: on input an encoding key ek and the vector $\mathbf{v}_i$ $(i \in [n])$, for each $v_{i,j} \in \mathbf{v}_i$ $(j \in [m])$, choose a random $y_i \in \mathbb{Z}_q$ to compute $u'_i = g^{y_i}$ and let $k_i^{\mathsf{DE}} \| k_i^{\mathsf{MA}} := H(h^{y_i})$; compute $u_{i,j} \leftarrow \mathsf{DE.Enc}_{k_i^{\mathsf{DE}}}(v_{i,j})$ and $t_{i,j} \leftarrow \mathsf{MA.Mac}_{k_i^{\mathsf{MA}}}(u_{i,j})$, then $c_{i,j} = (u_{i,j} \| t_{i,j})$, output the ciphertext $\mathbf{c}_i = (u'_i, c_{i,1}, \ldots, c_{i,m})$.
- $\mathsf{Decoding}(dk, \mathbf{c}_i) \rightarrow \mathbf{v}_i$: on input the decoding key dk and a ciphertext $\mathbf{c}_i = (u'_i, c_{i,1}, \ldots, c_{i,m})$ where $c_{i,j} = (u_{i,j} \| t_{i,j})$, if $u'_i \notin \mathbb{G}$ output $\perp$, else compute $k_i^{\mathsf{DE}} \| k_i^{\mathsf{MA}} := H(u'^{dk}_i)$; if $\mathsf{MAC.Vrfy}_{k_i^{\mathsf{MA}}}(u_{i,j}, t_{i,j}) \neq 1$ then output $\perp$; otherwise let $v_{i,j} = \mathsf{DE.Dec}_{k_i^{\mathsf{DE}}}(u_{i,j})$ and output $\mathbf{v}_i := \{v_{i,1}, \ldots, v_{i,m}\}$.
- $\mathsf{Eval}(dk, f, \{\mathbf{c}_i\}) \rightarrow f(\{\mathbf{v}_i\})$: on input the decoding key dk, an aggregation function f, a ciphertext set $\{\mathbf{c}_i\}$, for each $c_{i,j} \in \{\mathbf{c}_i\}$, output $\perp$ if $u'_i \notin \mathbb{G}$, else compute $k_i^{\mathsf{DE}} \| k_i^{\mathsf{MA}} := H(u'^{dk}_i)$; if $\mathsf{MAC.Vrfy}_{k_i^{\mathsf{MA}}}(u_{i,j}, t_{i,j}) \neq 1$ then output $\perp$; otherwise compute each $v_{i,j} := \mathsf{DE.Dec}_{k_i^{\mathsf{DE}}}(u_{i,j})$ to obtain the plaintext set $\{\mathbf{v}_i\} = \{v_{i,1}, \ldots, v_{i,m}\}$. Finally compute and output $f(\{\mathbf{v}_i\})$.

Fig. 5. A Privacy-preserving function evaluation (PFE) scheme.

6.2. Definitions of certificate-aided function authorization

At first, we define the *data-owner-specified function* as below.

Definition 4 (*Data-Owner-Specified Function*)**.** Let $\mathcal{F}$ be a function family. For any $f \in \mathcal{F}$, a certificate $CERT_i$ is generated to authorize f if $f \in W_i$, where $W_i = \{f_{i,1}, \ldots, f_{i,s_i}\}$ is called a *function whitelist* of $\mathcal{P}_i$, $f_{i,u} \in \mathcal{F}$, $s_i \leq |\mathcal{F}|$, $s_i \in \mathbb{N}$, and $u \in [s_i]$. The functions in W_i are called *data-owner-specified functions*.

Note that each data owner periodically updates its whitelist policy. $f \in W_i$ means there exist an $f_{i,u} \in W_i$ where $f_{i,u} = f$, i.e., f is "data-owner-specified". Then, the *Certificate-aided Function Authorization* mechanism is constructed in Definition 5, which is jointly performed by data owners and the aggregator (actually, the $\mathcal{TEE}_{\mathcal{CS}}$) before the function evaluation.

Definition 5 (*Certificate-aided Function Authorization*)**.** A Certificate-aided Function Authorization (CFA) scheme is a tuple of four PPT algorithms (Setup, SKeyGen, CertGen, CertVrfy) that satisfies the consistency property below.

- $\mathsf{Setup}(1^\lambda) \rightarrow pp$: this trusted setup algorithm is executed once by $\mathcal{KDC}$ to initialize the system. On input a security parameter 1^λ, it outputs the system parameters pp, which is published to all other parties.
- $\mathsf{SKeyGen}(pp) \rightarrow (sk_i, vk_i)$: this signature key generation algorithm is executed by data owners $\mathcal{P}_i$, on input the system

Let λ be the security parameter, $f \in \mathcal{F}$ be an aggregation function, W_i be $\mathcal{P}_i$'s function whitelist, CFA is constructed below.

- $\mathsf{Setup}(1^\lambda) \to pp$: on input a security parameter 1^λ, generate the system parameters $pp = (n, \mathcal{G}, \mathcal{D}, \mathcal{R}, \mathcal{F})$.
- $\mathsf{SKeyGen}(pp) \to (sk_i, vk_i)$: on input the system parameters pp, run $\mathcal{G}(1^\lambda)$ to generate $(\hat{\mathbb{G}}_i, \hat{q}_i, \hat{g}_i)$ where $i \in [n]$; then choose a random $\hat{x}_i \in \mathbb{Z}_{\hat{q}_i}$ and set $\hat{h}_i := \hat{g}_i^{\hat{x}_i}$; finally output the signing key $sk_i = \hat{x}_i$, and the verifying key $vk_i = (\hat{\mathbb{G}}_i, \hat{q}_i, \hat{g}_i, \hat{h}_i)$.
- $\mathsf{CertGen}(sk_i, W_i, f) \to CERT_i/\ \perp$: on input a signing key sk_i, a function whitelist W_i, and a function f, output a certificate $CERT_i = \{ref_i, f, \varepsilon_{i,u}, r_i, Sig_i\}$ if $f = f_{i,u} \in W_i$, where ref_i is the reference number to identify this request uniquely, $\varepsilon_{i,u}$ records the relevant information about this certificate such as the involved entities' ids, r_i represents random coins, and $Sig_i = \mathsf{Sn.Sign}_{sk_i}(ref_i, f, \varepsilon_{i,u}, r_i)$; otherwise output $\perp$.
- $\mathsf{CertVrfy}(vk_i, CERT_i, f) \to 1/0$: on input the verifying key vk_i, the certificate $CERT_i$, and a function f, output 1 if $\mathsf{Sn.Vrfy}_{vk_i}(Sig_i) = 1$ and $f \in CERT_i$; otherwise outputs 0.

Fig. 6. A Certificate-aided function authorization (CFA) scheme.

parameters pp, output the signature key pair (sk_i, vk_i), where the signing key sk_i is kept secret to sign the certificate, and the verifying key vk_i is published in public.

- $\mathsf{CertGen}(sk_i, W_i, f) \to CERT_i/\ \perp$: this certificate generation algorithm is executed by data owners, on input the signing key sk_i, the function whitelist W_i, and the aggregation function f from the analyst. If f is a data-owner-specified function, a certificate $CERT_i$ including a signature is generated and transmitted to the analyst, and recorded in a sound manner simultaneously; otherwise outputs $\perp$.
- $\mathsf{CertVrfy}(vk_i, CERT_i, f) \to 1/0$: this certificate verification algorithm is run in the $\mathcal{TEE}_{\mathcal{CS}}$, on input the verifying key vk_i, the certificate $CERT_i$, and the function f, outputs 1 if $f \in CERT_i$ hold and the signature of $CERT_i$ is valid; otherwise this algorithm outputs 0.

Consistency. For all $\lambda \in \mathbb{N}$, any pp generated by $\mathsf{Setup}(1^\lambda)$, any (sk_i, vk_i) generated by $\mathsf{SKeyGen}(pp)$, we have $1 \leftarrow \mathsf{CertVrfy}(vk_i, CERT_i, f)$ if $CERT_i \leftarrow \mathsf{CertGen}(sk_i, W_i, f)$.

6.3. Our certificate-aided function authorization construction

Similarly, we give an instantiated certificate-aided function authorization scheme construction, and refer to this as CFA from here onwards. We employ the digital signature $\mathsf{Sn} = (Sign, Vrfy)$ to construct CFA in Fig. 6.

Note that, we arrange the data owner $\mathcal{P}_i$ as the authority to decide whether to generate a valid certificate for function f or not, and the $\mathcal{TEE}_{\mathcal{CS}}$ runs CertVrfy to verify the certificates. That is, only the data-owner-specified functions are allowed to be computed on the data in $\mathcal{TEE}_{\mathcal{CS}}$, and vice versa. This effectively prevents the leakage from the functions that the data owners authorized. We only provide a kind of thought for this issue, the detailed authorization strategy is beyond the scope, which is not discussed in this paper.

Certificate Verification Protocol CVP:

Parameters: n parties $\mathcal{P}_i$ ($i \in [n]$), the analyst $\mathcal{P}_0$, the cloud server $\mathcal{CS}$ with secure $\mathcal{TEE}_{\mathcal{CS}}$, the public ledger $\mathcal{PL}$. The system parameters pp, the certificate $CERT_i$, the random coins r_i, the commitment scheme Commit.

Protocol:

1. On input the generated certificate $CERT_i$ and the random coins r_i, $\mathcal{P}_i$ executes $\mathsf{Commit}(pp, CERT_i, r_i) \to COM_i$, and records COM_i with id ID_i in the public ledger, and sends $(CERT_i, r_i)$ to $\mathcal{P}_0$.
2. $\mathcal{P}_0$ sends $(f, CERT_i, r_i)$ to $\mathcal{CS}$.
3. $\mathcal{CS}$ initializes $\mathcal{TEE}_{\mathcal{CS}}$, which sends the remote attestation to $\mathcal{PL}$.
4. $\mathcal{PL}$ generates a signature σ_i for COM_i, and responds $\mathcal{TEE}_{\mathcal{CS}}$ with $COM_i \| \sigma_i$.
5. $\mathcal{TEE}_{\mathcal{CS}}$ verifies σ_i to make sure COM_i comes from $\mathcal{PL}$; then executes $\mathsf{Commit}(pp, CERT_i, r_i) \to COM_i'$. Outputs 1 if $COM_i = COM_i'$; otherwise outputs 0.

Fig. 7. A Certificate verification protocol CVP.

6.4. Public verifiable certificate management

We have considered using authentication encryption to protect data and a certificate-aided method to authorize the functions. It is easy to see that while the tampering with input or output of the TEE and leakage from functions are prevented, a compromised host still may tamper with the program (e.g., replace it with an old function, or with an old certificate). To mitigate this, we consider an alternative solution to validate the function by combining the TEE with the public ledger (e.g., the blockchain) to manage the certificates. However, utilizing the public ledger means exposure to all parties involved, which increases the risk of certificate theft. Thus, we introduce a commitment scheme Commit in our design. Commit takes the system parameters pp, a message M, and random coins r as input, outputs a commitment COM that can be verified via recalculating the commitment on the same message and coins.

In particular, when $\mathcal{P}_i$ creates a certificate $CERT_i$ for function f requested by $\mathcal{P}_0$, it also generates a cryptographic commitment COM_i for $CERT_i$ and records it in the public ledger. We require that the public ledger can prevent stored items from being modified or eliminated, and generate a proof (e.g., a signature) to demonstrate the publication is valid. Any party including a TEE can verify this proof to guarantee the received items are authentic. We present an improvised *Certificate Verification Protocol* CVP in Fig. 7. To aid understanding, we present the random coins r_i independent of the certificate $CERT_i$ in CVP. We omit the description of r_i in Section 4.3, because r_i can be included in $CERT_i$ to guarantee its integrity.

Remark 3. We model the public ledger as an unforgeable proof of publication, which has the ability to provide public verifiability, signature, and timestamps generation properties. Actually, our design is also inspired by real-world decentralized public ledger, which can be implemented by realistic ledgers, of which many "proof-of-work" blockchains provide a weaker unforgeable guarantee. It is exactly expensive to forge a proof of publication. These notions may provide sufficient security in real-world applications.

Our starting point is to guarantee the function validation. At first, the certificate is protected with commitment, which avoids

$\mathsf{SecGame}^{\mu-\mathrm{p}}_{\Pi,\mathcal{A}}(1^\lambda, n)$:
$pp \leftarrow \mathsf{Setup}(1^\lambda)$;
$(ek, dk) \leftarrow \mathsf{CKeyGen}(pp)$;
$t \leftarrow \mathcal{A}(n)$;
$(t, \{\mathbf{v}_i\}/\mathbf{v}_t, \mathbf{v}_t^0, \mathbf{v}_t^1) \leftarrow \mathcal{A}(ek, t, n, \mathcal{D})$;
$b \xleftarrow{\$} \{0, 1\}$;
$\mathbf{v}_t \leftarrow \mathbf{v}_t^b$;
$\{\mathbf{v}_i\} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$;
$\{\mathbf{c}_i\} \leftarrow \mathsf{Encoding}(ek, \{\mathbf{v}_i\})$;
$f(\{\mathbf{v}_i\}) \leftarrow \mathsf{Eval}(dk, f, \{\mathbf{c}_i\})$;
$b' \leftarrow \mathcal{A}(\{\mathbf{c}_i\}, f(\{\mathbf{v}_i\}))$;

Fig. 8. μ-privacy game.

privacy exposure. Secondly, the public verifiability property of the public ledger provides the integrity of certificate commitment. Finally, the appended signature proves it is from the public ledger, and the timestamp property of the public ledger protects it from the certificate replay attacks. The online computation and communication cost of function validation is tiny compared to the total overhead, which applies to the constrained terminals.

7. Security

7.1. Security model

In Definition 6 we define the μ-privacy property under the compromised cloud server that has access to the encoded data and the final result. This μ-privacy definition implies that a compromised cloud server can infer some information about the plain data with negligible probability $\mu(\lambda)$.

Definition 6 (μ-*privacy*)**.** Let $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a set of n data owners. A non-interactive PDA scheme Π is μ-private if for all $n \in \mathbb{N}$, and all PPT adversary $\mathcal{A}$ in $\mathsf{SecGame}^{\mu-\mathrm{p}}_{\Pi,\mathcal{A}}(1^\lambda, n)$ (Fig. 8), there exists a negligible function μ s.t. the advantage of $\mathcal{A}$

$$\mathsf{Adv}^{\mu-\mathrm{p}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[b' = b] - \frac{1}{2}| \leq \mu(\lambda).$$

Remark 4. In the μ-privacy game, $\mathcal{A}$ selects the plain messages for each data owner, which means $\mathcal{A}$ is also allowed to collude with any data owner. We can conclude that a scheme Π is collusion-resistant against the compromised cloud server if Π achieves μ-privacy.

The analyst is entitled to initialize the function f, implying that an unauthorized function (e.g., $f(x) = x$) can be included in the query such that the result reveals something sensitive. In Definition 7 we define the $\mathcal{F}$-hiding property under the malicious analyst, which means that a malicious analyst learns nothing about the plain data even though he initializes f maliciously.

Definition 7 ($\mathcal{F}$-*hiding*)**.** Let $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a set of n data owners. A non-interactive PDA scheme Π is $\mathcal{F}$-hidden if for all $n \in \mathbb{N}$, all function $f \in \mathcal{F}$ and all PPT adversary $\mathcal{A}$ in $\mathsf{SecGame}^{\mathcal{F}-\mathrm{h}}_{\Pi,\mathcal{A}}(1^\lambda, n, f)$ (Fig. 9), there exists a negligible function negl s.t. the advantage of $\mathcal{A}$

$$\mathsf{Adv}^{\mathcal{F}-\mathrm{h}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

Remark 5. In the $\mathcal{F}$-hiding experiment, we let the adversary $\mathcal{A}$ decide the challenge function $f \in \mathcal{F}$, but what $\mathcal{A}$ obtains is only the approximate evaluation result. $\mathcal{A}$ also is allowed to collude with any data owner. We can conclude that a scheme Π is collusion-resistant against the malicious analyst if this scheme Π achieves $\mathcal{F}$-hiding security.

$\mathsf{SecGame}^{\mathcal{F}-\mathrm{h}}_{\Pi,\mathcal{A}}(1^\lambda, n, f)$:
$pp \leftarrow \mathsf{Setup}(1^\lambda)$;
$(ek, dk) \leftarrow \mathsf{CKeyGen}(pp)$;
for i = 1 : n do
  $(sk_i, vk_i) \leftarrow \mathsf{SKeyGen}(pp)$
    for j = 1 : $\mathbf{s_i}$ do
    $f_{i,j} \leftarrow \mathcal{F}$
    end for
  $W_i = (f_{i,1}, \ldots, f_{i,s_i})$
end for
$(f, t) \leftarrow \mathcal{A}(\mathcal{F}, n)$;
for i = 1 : n do
$\{CERT_i\} \leftarrow \mathsf{CertGen}(sk_i, W_i, f)$
end for
$(f, t, \{CERT_i\}, \{\mathbf{v}_i\}/\mathbf{v}_t, \mathbf{v}_t^0, \mathbf{v}_t^1) \leftarrow \mathcal{A}(\{CERT_i\}, n, \mathcal{D})$
$b \xleftarrow{\$} \{0, 1\}$;
$\mathbf{v}_t \leftarrow \mathbf{v}_t^b$;
$\{\mathbf{v}_i\} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$;
$\{\mathbf{c}_i\} \leftarrow \mathsf{Encoding}(ek, \{\mathbf{v}_i\})$;
$1/0 \leftarrow \mathsf{CertVrfy}(vk_i, CERT_i, f)$;
$f(\{\mathbf{v}_i\}) \leftarrow \mathsf{Eval}(dk, f, \{\mathbf{c}_i\})$;
$b' \leftarrow \mathcal{A}(f(\{\mathbf{v}_i\}))$;

Fig. 9. $\mathcal{F}$-hiding game.

7.2. Security analysis

To show PANDA is secure and does not leak any information except the final result, we need the following theorems.

Theorem 1. *If* DE *is CPA-secure, then* PANDA *achieves the* μ*-privacy property.*

Proof Sketch 1. For the compromised cloud server $\mathcal{A}$, it observes all the encoded items from the n data owners $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. $\mathcal{A}$ selects $\mathcal{P}_t$ ($t \in [n]$) as the challenge data owner from the n data owners $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, chooses $n-1$ uniform vector messages $\{\mathbf{v}_i\}/\mathbf{v}_t$ ($i \in [n]$, $\mathbf{v}_i \in \mathcal{D}$), then chooses two random messages $\mathbf{v}_t^0, \mathbf{v}_t^1 \in \mathcal{D}$ and sends $(\{\mathbf{v}_i\}/\mathbf{v}_t, \mathbf{v}_t^0, \mathbf{v}_t^1)$ to $\mathcal{C}$. Upon receiving the query, $\mathcal{C}$ chooses $\mathbf{v}_t^b$ where $b \in \{0, 1\}$ and let $\mathbf{v}_t = \mathbf{v}_t^b$ to compose a new message set $\{\mathbf{v}_i\} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, then runs the $\mathsf{Encoding}(ek, \{\mathbf{v}_i\})$ algorithm to convert $\{\mathbf{v}_i\}$ to the ciphertext set $\{\mathbf{c}_i\}$ where $\mathbf{c}_i = \mathsf{Encoding}(ek, \mathbf{v}_i)$, then runs $\mathsf{Eval}(dk, f, \{\mathbf{c}_i\})$ to obtain an result $f(\{\mathbf{v}_i\})$, sends $\{\mathbf{c}_i\}$ and $f(\{\mathbf{v}_i\})$ to $\mathcal{A}$. As the DE is CPA-secure, and f is not chosen by the adversary $\mathcal{A}$, so $\mathcal{A}$ cannot distinguish whether $b = 0$ or $b = 1$ with non-negligible probability. Therefore, the advantage of the adversary $\mathcal{A}$ is

$$Adv^{\mu-p}_{\mathsf{PDA},\mathcal{A}}(\lambda) := |\Pr[b' = b] - \frac{1}{2}| \leq \mu(\lambda)$$

where $\mu(\lambda)$ is a negligible function μ of λ. Therefore, we consider PANDA is secure against the compromised cloud server who only attempts to infer private information of data owners without malicious behaviors.

In PANDA, we combine the TEE with several cryptographic tools to limit the malicious behaviors of the cloud server. When

Table 3
Comparison with previous work.

Scheme		Communication		Computation		Storage
Bonawitz et al. [9]		*Data owner* $O(n+m)$	*Server* $O(n^2+nm)$	*Data owner* $O(n^2+nm)$	*Server* $O(n^2\ m)$	*Data owner* $O(n+m)$
Guo–Tian–Cho et al. [4]		*Data owner* $O(m)$	*Server* $O(nm)$	*Data owner* $O(nm)$	*Server* $O(nm)$	*Data owner* $O(nm)$
PANDA	*Online*	$O(1)$	$O(2n+1)$	$O(1)$	$O(nm+n+1)$	*Server*
	Offline	$O(m)$	$O(nm)$	$O(m)$	0	$O(nm)$

the compromised cloud server $\mathcal{A}$ tries to tamper with the ciphertexts, the unforgeability property provided by the message authentication code makes the modification detectable. Similarly, when the compromised cloud server attempts to tamper with the function that is included in the certificate, the modification can also be found easily because: (1) the certificate contains a signature created by its data owner; (2) the committed version is recorded in the public ledger, which means the undetectable modification requires a mass of computation cost; (3) the public ledger provides a proof (e.g., a digital signature) for the stored committed certificate. So the malicious cloud server can tamper with the ciphertext and function with very negligible advantage.

Above all, we can conclude that the malicious cloud server can infer nothing from the final output in PANDA.

Theorem 2. *If each data owner $\mathcal{P}_i$ has established a sound whitelist W_i,* Sn *is unforgeable, then* PANDA *achieves $\mathcal{F}$-hiding property against the malicious analyst.*

Proof Sketch 2. We show the $\mathcal{F}$-hiding property via the game in Fig. 9 and we assume all the sound whitelists $\{W_i\}$ are setup properly in advance. For the compromised analyst $\mathcal{A}$, it learns nothing about the vectors except the approximate result $f(\{\mathbf{v}_i\})$, but can initiate an arbitrary function $f \in \mathcal{F}$. As described in the game, the challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ to generate the public parameter pp, also runs $\mathsf{CKeyGen}(pp)$ and $\mathsf{SKeyGen}(pp)$ to generate keys. Then $\mathcal{A}$ chooses $\mathcal{P}_t$ $(t \in [n])$ as the challenge data owner from the n data owners $\{\mathcal{P}_i\}$ $(i \in [n])$, chooses a query function $f \in \mathcal{F}$ and submits (f, t) to the challenger $\mathcal{C}$ for authentication. $\mathcal{C}$ runs $\mathsf{CertGen}(sk_i, W_i, f)$ to generate the certificate set $\{CERT_i\}$ for the data owners who agree to compute f on their data. Then $\mathcal{A}$ chooses $n-1$ random vector messages from the plain data space $\mathcal{D}$ as $\{\mathbf{v}_i\}/\mathbf{v}_t$, and uniformly chooses two messages $\mathbf{v}_t^0$ and $\mathbf{v}_t^1$. Finally, $\mathcal{A}$ sends $(f, t, \{CERT_i\}, \{\mathbf{v}_i\}/\mathbf{v}_t, \mathbf{v}_t^0, \mathbf{v}_t^1)$ to $\mathcal{C}$. Upon receiving the messages, $\mathcal{C}$ randomly chooses $\mathbf{v}_t^b$ $(b \in \{0,1\})$ as $\mathbf{v}_t$, then encrypts all the messages by running the $\mathsf{Encoding}(ek, \{\mathbf{v}_i\})$ algorithm to obtain the ciphertext set $\{\mathbf{c}_i\}$. We define the set D_0 as $D_0 = \{\mathbf{c}_1, \ldots, \mathbf{c}_t^0, \ldots, \mathbf{c}_n\}$, and the set D_1 as $D_1 = \{\mathbf{c}_1, \ldots, \mathbf{c}_t^1, \ldots, \mathbf{c}_n\}$, where the two sets differ on only one record. Afterwards, $\mathcal{C}$ runs $\mathsf{CertVrfy}(vk_i, CERT_i, f)$ to verify the validation of the request f and the corresponding certificate $CERT_i$. If both f and $CERT_i$ are proved valid, $\mathcal{C}$ runs $\mathsf{Eval}(dk, f, \{\mathbf{c}_i\})$ to obtain the output $f(\{\mathbf{v}_i\})$ and sends $f(\{\mathbf{v}_i\})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess b'.

If $b = b'$ always holds, it means the adversary $\mathcal{A}$ can distinguish the two results $f(D_0)$ and $f(D_1)$ with non-negligible probability. Since Sn provides unforgeability, the advantage of $\mathcal{A}$ is denoted as

$$Adv^{\mathcal{F}-h}_{\mathsf{PDA},\mathcal{A}}(\lambda) := |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$$

is negligible.

Replay attacks are among the most concerning issues of the TEEs. That is, the malicious analyst may replay old certificates to the TEE along with new inputs. In PANDA, we put a unique reference number into every certificate, and store it in the public ledger that generates specific timestamps for the transactions. Thus, the replay attacks are obviously avoided.

We can conclude that the malicious analyst can infer nothing from the final output in PANDA.

8. Evaluation

We summarize PANDA's performance in this section. The overhead of both data owners and the cloud server can be divided into online and offline parts. The offline cost involves the initialization and vector encoding phases, and the online cost involves the authentication and evaluation phases.

8.1. Theoretical analysis

We analyze the communication and computation performance for each data owner and the cloud server, respectively.

Computation cost of each data owner: $O(1)$ online computation and $O(m)$ offline computation. Each data owner $\mathcal{P}_i$'s computation cost can be broken up as (1) selecting s_i functions and creating the function whitelist which can be done in advance, (2) generating the signing-verifying key pair, then encoding the vector elements, which takes $O(m)$ time and (3) generating and committing a certificate for each query $O(1)$. Note the first two items are included as the offline cost, the third item is online cost.

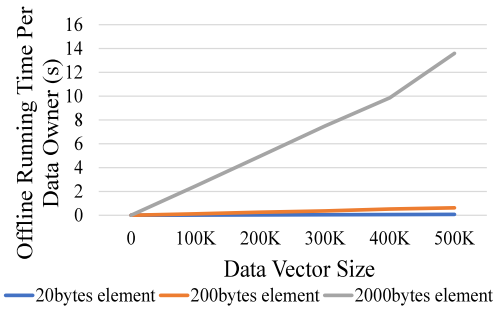
Communication cost of each data owner: $O(1)$ online communication and $O(m)$ offline communication cost. The communication cost of each data owner can be broken up as (1) sending a ciphertext of an m-dimensional vector, including a ciphertext of the secret key, m ciphertexts of DE, and m message authentication codes of MA to the cloud server, which are considered as offline cost, (2) sending a certificate to the analyst and recording a committed version in the public ledger cost $O(1)$ which are considered as online cost.

Computation cost of the cloud server: $O(nm+n+1)$ online computation cost. The computation cost of the cloud service provider can be divided into (1) verifying the certificates from the analyst costs $O(n)$, (2) evaluating the nm ciphertexts in total $O(nm)$, (3) signing the result costs $O(1)$. Note both the two items are online cost.

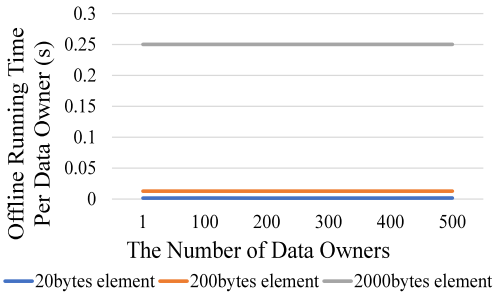
Communication cost of the cloud server: $O(n+1)$ online communication and $O(nm)$ offline communication cost. The communication cost of the cloud server can be divided into (1) receiving the $O(m)$ ciphertexts for n data owners, (2) receiving the query and the certificates from the analyst costs $O(n+1)$, (3) requiring and receiving the proofs and commitments from the public ledger cost $O(n)$. Note the first item is offline cost and the remaining items are online cost.

Storage cost of the cloud service provider: $O(nm)$. The cloud server must store the encrypted items for each data owner, which is $O(nm)$ in total.

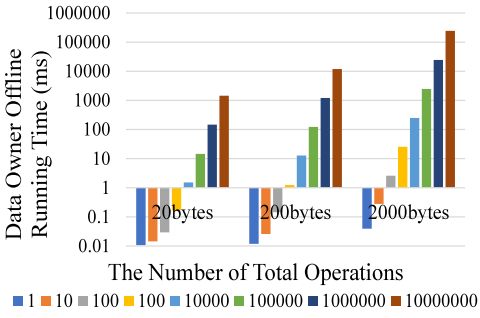
We compare our scheme with [4,9] on communication and computation cost, because as far as we know, they have the representative performance among the existing solutions. The theoretical complexity of the two schemes is given in Table 3. In



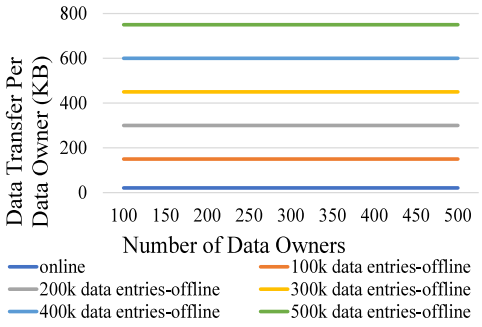
(a) m grows when $n = 50$.



(b) n grows when $m = 10K$.



(c) mn increases and different sets of bars correspond to different element length.



(d) n increases and different lines correspond to different m.

Fig. 10. Data owner running time and data transfer costs.

summary, our scheme has more efficient communication performance because online consumption is only $O(1)$ for data owners and $O(2n + 1)$ for the server. For the computation, our scheme is also not worse than [4,9], the server undertakes most online computation $O(nm + n + 1)$ and each data owner performs only $O(m)$ offline operations. For the storage, the server needs $O(nm)$ storage consumption in our work due to the outsourced setting where the server has powerful computation and storage capacity and no need for the data owners' storage capacity. In contract, each data owner must store $O(n + m)$ items in [9].

8.2. Experimental evaluation

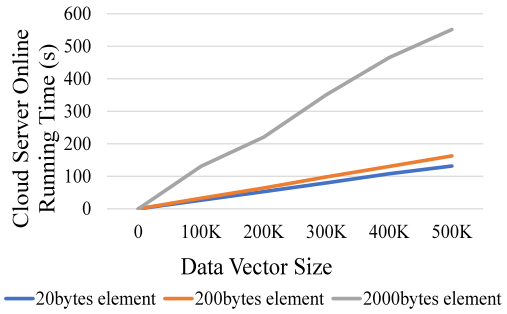
To measure the performance of PANDA, we implemented a prototype with the Intel(R) SGX SDK 2.0 in simulation mode hosting on the server with Ubuntu 18.04.2. The CPU is E5-2630 V3 with 2 processors 2.40 GHz and 2.39 GHz. The memory installed is 128 GB. We use a desktop as the data owner with AMD Ryzen 7 7100 Eight-Core Processor and 8.00 G RAM. For the key encapsulation mechanism, we used Elliptic-Curve Integrated Encryption scheme over the NIST P-256 curve. For the encoding scheme, we used AES-GCM scheme with 128-bit length keys.

In our simulations, we have negligible online communication for the data owners due to the non-interactive model, and also excellent offline communication because our symmetric encoding scheme has very low ciphertext expansion. For the computation, each data owner only undertakes offline encryption operations, while almost all the online computation cost comes from the cloud server that has plenty of computation and storage capacity.

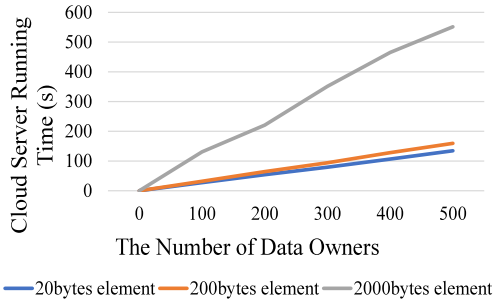
For each data owner, both the online computation and communication costs are negligible because data vectors are outsourced to the cloud server in the offline phase. As seen in Figs. 10a and 10b, the offline running time per data owner increases linearly with the data vector size, and is correlated with the element length of the data vector, but does not change as the number of data owners increases (also implied in Table 4). This is because each data owner performs outsourcing respectively without redundant interactions between them. In Fig. 10c, the offline running time per data owner increases as the number of total operations increases to tens of millions, and also is influenced by the vector element length. In Fig. 10d, each data owner's communication cost changes as the number of data owners increases, the online communication cost is tiny, while the offline communication cost regularly differs as the data vector size changes.

In the case of the cloud server, online communication and offline computation costs are close to negligible intuitively. The Figs. 11a and 11b show that the online running time of the cloud server increases linearly with the data vector size and the number of data owners, and also changes significantly as the vector element length increases (also implied in Table 4). Fig. 11c shows a similar data distribution with data owners, that the online running time of the cloud server increases with the number of total operations, but is slightly slower than the case of the data owner. Fig. 11d shows that the cloud server's online transfer cost is very low, but the online data transmission cost increases linearly with the number of data owners.

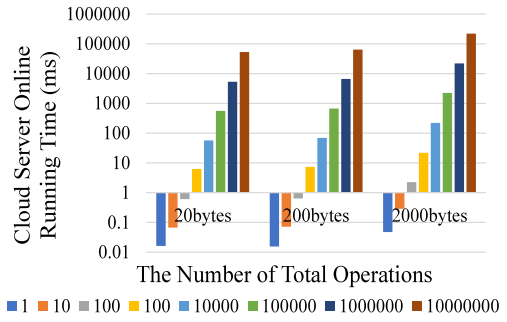
To measure the computation pressure on the constrained devices, we also use Google Pixel 4 as the constrained data owner with Qualcomm Technologies, Inc SM8150 (msmnile) 2.84G Eight-Core Processor, 5.34G RAM and 50.01G storage. For the data owner side, we implement the encoding scheme with AES-GCM with 128-bit key by utilizing android.security.keystore library. The running time (ms) of the data encoding phase for different terminals is presented in Table 5, where each column shows the time corresponding to the total encoding operations. We can conclude that the running time of Google Pixel 4 also increases with the number of encoding operations, but has weaker sensitivity for the element length. The possible reason is the libraries that we utilize are different. In brief, the cost of the encoding phase is more stressful for the Google Pixel 4 than the desktop, but still practical and acceptable enough for resource-limited devices.



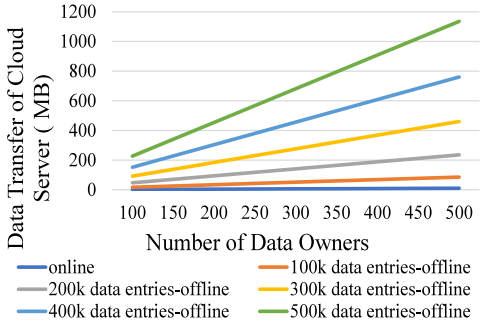
(a) m grows when $n = 50$.



(b) n grows when $m = 10K$.



(c) mn increases and different sets of bars correspond to different element length.



(d) n increases and different lines correspond to different m.

Fig. 11. Cloud server running time and data transfer costs.

Table 4
Running time (ms) for data processing. The data vector size is fix to 50k entries with 20-byte entries.

Role	n	1	100	200	300	400	500
Data owner	Offline	7.58175	8.28175	7.45817	7.81759	7.48317	7.87665
Cloud server	Online	256.38	27 169.66	53 970.30	79 292.42	10 647.11	134 520.072

Table 5
Running time (ms) of data encoding phase for different terminals (data owners).

Element length	Data owner	1	10	50	100	500	1000	10,000	100,000	200,000	500,000
20 bytes	Desktop	0.01072	0.0144	0.02459	0.02926	0.09254	0.16751	1.51635	14.56354	27.8646	74.3569
	Google Pixel 4	2.8	10.9	46.5	94.9	472.9	997.8	11 118.2	170 165.1	346 891.1	851 170.9
200 bytes	Desktop	0.01187	0.02582	0.08128	0.14474	0.62855	1.24336	12.80814	123.0111	256.875	624.876
	Google Pixel 4	3.5	11.5	49.1	96.9	475.4	1006.9	11 461.6	17 734.2	349 141.6	871 969.1
2000 bytes	Desktop	0.0392	0.2781	1.26832	2.59231	11.25994	25.47169	250.1696	2463.99	4976.98	13 596.88
	Google Pixel 4	4.0	12.1	50.2	98.9	477.2	1017.9	11 595.4.3	181 795.8	356 749.2	919 560.8

9. Related work

As a promising data process approach, PDA has been studied extensively. As illustrated in Fig. 12, most existing PDA schemes can be divided into two categories: *interactive* PDA schemes and *non-interactive* PDA schemes.

Interactive PDA. Interactive PDA solutions ensure data privacy by maintaining the data locally. Each data owner continuously processes their data and sends the processed data to the aggregator, which cannot retrieve any raw data. Interactive PDA solutions usually require multiple interactions. To the best of our knowledge, the most commonly used interactive PDA approaches are the *federated learning* based, the *MPC* based, and the *distributed differential privacy (DP)* based approaches. Federated learning [20–22] distributes the machine learning process over to edge devices, where each data owner maintains a private database, and trains a shared global model jointly with other data owners. Although data owners only upload the updates of the model trained locally instead of their database, these updates still may reveal sensitive information [9,13,15]. Bonawitz et al. [9] proposed a communication-efficient secure aggregation protocol for high-dimensional data, which allows the server to compute the sum of data vectors in a federated learning setting. ManGon [13] designed a privacy-preserving system for the federated setting, which can guarantee both the data and model privacy. Chamikara et al. [15] designed a distributed perturbation algorithm for privacy preservation of horizontally partitioned data in federated learning. [26] is also an obvious solution to achieve the privacy goals we are aiming for. The notion of secure multi-party computation starts initially from the Garbled Circuit [23] which solves the millionaires' problem. Secure multi-party computation aims to privately compute functions $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})$ on n entities' inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. HIE-CHO-BER [16] introduce a computational protocol for biometric research. In addition, there are some other representative works [17–19] presented in the past few years. However, most cryptographic primitives served

for MPC solutions are significantly expensive with a large amount of interactions among all entities, which is not friendly with a huge number of data owners or large input sizes.

Distributed differential privacy [44] is also a promising candidate technique in achieving PDA. The notion of differential privacy [44] makes it possible to compute a statistical result on multiparty private sets while maintaining privacy. Dwork et al. [44] presented efficient distributed protocols for generating shares of random noise, while involving interactions among all multiple parties. Rastogi et al. [45] also considered distributed time-series data and proposed the first differentially private aggregation scheme, but there is nothing we can do about the linear combinations of multi-source items. Shi et al. [46] designed a distributed data randomization procedure to guarantee the differential privacy of the outcome statistic, even when some participants might be compromised. Cha–Shi–Son [47] combined applied cryptography and differential privacy techniques to explore novel PDA mechanisms that offer provable guarantee of user privacy against the untrusted aggregation. These works achieve privacy preserving, but they usually appear with the costly cryptographic techniques (e.g., HE). Additionally, interactive PDAs strongly depend on the reliability of the edge devices, which are required to be online all the time.

Non-interactive PDA. Non-interactive PDA solutions aggregate data by sending the data to the aggregator all at once in a secure way. Each data owner processes the data in advance and stores the processed data in the aggregator before aggregation. Non-interactive PDA solutions have no interaction between the data owner and the aggregator during aggregation. As far as we know, the non-interactive PDA research mainly centers on the HE based, the *functional encryption* (FE) based, and the *unique sequence number* based approaches. HE [27,28] is a form of encryption that allows generating an encrypted result directly on the ciphertexts, corresponding to the result on the plain items. Castelluccia et al. [35] proposed a symmetric key additively homomorphic encryption to lead the aggregation operations on ciphertexts, but they assumed a trusted aggregator. Then Rieffel et al. [32] designed a construction that does not require a trusted aggregator, but it consumes high storage and computation source against collusion. Li et al. [33] presented a sum aggregation protocol based on [35] without the trusted aggregator, and studied the privacy-preserving Min computation. Zhang et al. [2] utilized XOR HE to securely compute the Min and kth Min values. Burkhalter et al. [34] presented a system that provides scalable and real-time analytics over large volumes of homomorphic ciphertexts. However, these schemes based on HE either only focus on specific function or are inefficient due to this costly cryptographic primitive. *Functional encryption* [48], which enables users to learn a specific function of encrypted data with privacy preserved, can also achieve secure non-interactive PDA. Similarly, it aggregates data privately on ciphertexts, but has poor performance in the massive data setting. Zhang–Chen–Zhong [1] and Gong et al. [12] utilized the *unique sequence number* to realize non-interactive PDAs which support for arbitrary aggregation functions. However, their solutions reveal exact distribution of the data to the aggregator, such leakage cannot be ignored in our scenarios.

10. Conclusions

To make PDA more applicable to the constrained devices in the IoT scenario, we presented PANDA, a novel lightweight non-interactive PDA scheme in the cloud-enabled IoT paradigm which ensures that data owners' inputs are invisible to other entities. To prevent the compromised aggregator from obtaining sensitive information, we designed the cloud server with trusted hardware

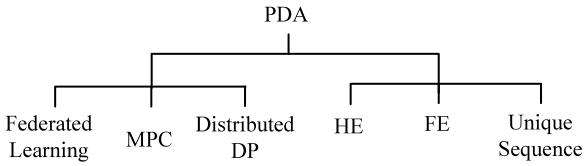


Fig. 12. Categories of PDA schemes.

(Intel SGX in our implementation), where an isolated TEE is built in which sensitive data are manipulated. The data are further protected by the certificate-aid function authentication mechanism and the public verifiable ledger. PANDA liberates data owners from expensive operations and frequent interactions compared to existing schemes. Both the interaction overhead between entities and the computation overhead of the data owners are very low, and the data owners store nothing, which makes it appropriate for constrained devices. Furthermore, the data owners have no need to be online all the time during aggregation, and PANDA is secure against collusion between minority entities.

CRediT authorship contribution statement

Mei Wang: Conceptualization, Methodology, Validation, Investigation, Writing – original draft. **Kun He:** Validation, Writing – review & editing, Project administration, Funding acquisition. **Jing Chen:** Methodology, Writing – review & editing, Funding acquisition. **Ruiying Du:** Project administration, Supervision, Funding acquisition. **Bingsheng Zhang:** Writing – review & editing, Formal analysis, Methodology. **Zengpeng Li:** Validation, Formal analysis, Visualization.

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.

Acknowledgments

This research was supported in part by the National Key R&D Program of China under grant No. 2021YFB2700200; by the National Natural Science Foundation of China under grants No. U1836202, 61772383, 62172303, 61802214, 62076187; by the National Natural Science Foundation of Hubei Province, China under grant No. 2020CFB628.

References

[1] Y. Zhang, Q. Chen, S. Zhong, Privacy-preserving data aggregation in mobile phone sensing, IEEE Trans. Inf. Forensics Secur. 11 (5) (2016) 980–992.
[2] Y. Zhang, Q. Chen, S. Zhong, Efficient and privacy-preserving min and kth min computations in mobile sensing systems, IEEE Trans. Dependable Sec. Comput. 14 (1) (2017) 9–21.
[3] A. Saleem, A. Khan, S.U.R. Malik, H. Pervaiz, H. Malik, M. Alam, A. Jindal, FESDA: fog-enabled secure data aggregation in smart grid IoT network, IEEE Internet Things J. 7 (7) (2020) 6132–6142.
[4] C. Guo, P. Tian, K.R. Choo, Enabling privacy-assured fog-based data aggregation in E-healthcare systems, IEEE Trans. Ind. Inf. 17 (3) (2021) 1948–1957.
[5] S. Zhao, F. Li, H. Li, R. Lu, S. Ren, H. Bao, J. Lin, S. Han, Smart and practical privacy-preserving data aggregation for fog-based smart grids, IEEE Trans. Inf. Forensics Secur. 16 (2021) 521–536.
[6] M.M. Groat, W. He, S. Forrest, KIPDA: k-indistinguishable privacy-preserving data aggregation in wireless sensor networks, in: Proc. of INFOCOM, IEEE Computer Society, 2011.
[7] S. Roy, M. Conti, S. Setia, S. Jajodia, Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact, IEEE Trans. Inf. Forensics Secur. 9 (4) (2014) 681–694.

[8] L. Zhou, C. Ge, S. Hu, C. Su, Energy-efficient and privacy-preserving data aggregation algorithm for wireless sensor networks, IEEE Internet Things J. 7 (5) (2020) 3948–3957.
[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proc. of CCS, ACM, 2017.
[10] P. Mohassel, Y. Zhang, SecureML: A system for scalable privacy-preserving machine learning, in: Proc. of S&P, IEEE Computer Society, 2017.
[11] T. Jung, X. Mao, X. Li, S. Tang, W. Gong, L. Zhang, Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation, in: Proc. of INFOCOM, IEEE Computer Society, 2013.
[12] X. Gong, Q. Hua, L. Qian, D. Yu, H. Jin, Communication-efficient and privacy-preserving data aggregation without trusted authority, in: Proc. of INFOCOM, IEEE Computer Society, 2018.
[13] K. Mandal, G. Gong, PrivFL: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks, in: Proc. of CCSW, ACM, 2019.
[14] W. Abramson, A.J. Hall, P. Papadopoulos, N. Pitropakis, W.J. Buchanan, A distributed trust framework for privacy-preserving machine learning, in: Proc. of TrustBus, Springer, 2020.
[15] M.A.P. Chamikara, P. Bertók, I. Khalil, D. Liu, S. Camtepe, Privacy preserving distributed machine learning with federated learning, Comput. Commun. 171 (2021) 112–125.
[16] B. Hie, H. Cho, B. Berger, Realizing private and practical pharmacological collaboration, Science 362 (6412) (2018) 347–350.
[17] D. Lia, M. Togan, Privacy-preserving machine learning using federated learning and secure aggregation, in: Proc. of ECAI, IEEE, 2020.
[18] R. Patel, P. Wolfe, R. Munafo, M. Varia, M.C. Herbordt, Arithmetic and Boolean secret sharing MPC on FPGAs in the data center, in: Proc. of HPEC, IEEE, 2020.
[19] P.-F. Wolfe, R. Patel, R. Munafo, M. Varia, M. Herbordt, Secret sharing MPC on FPGAs in the datacenter, in: Proc. of FPL, IEEE, 2020.
[20] J. Konecný, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, 2016, CoRR. URL http://arxiv.org/abs/1610.05492.
[21] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 12:1–12:19.
[22] Z. Li, V. Sharma, S.P. Mohanty, Preserving data privacy via federated learning: Challenges and solutions, IEEE Consum. Electron. Mag. 9 (3) (2020) 8–16.
[23] A.C. Yao, Protocols for secure computations (extended abstract), in: Proc. of FOCS, IEEE Computer Society, 1982.
[24] A.C. Yao, How to generate and exchange secrets (extended abstract), in: Proc. of FOCS, IEEE Computer Society, 1986.
[25] B. Barak, A. Sahai, How to play almost any mental game over the net - concurrent composition via super-polynomial simulation, in: Proc. of FOCS, IEEE Computer Society, 2005.
[26] Y. Lindell, B. Pinkas, N.P. Smart, A. Yanai, Efficient constant round multi-party computation combining BMR and SPDZ, in: Proc. of CRYPTO, Springer, 2015.
[27] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proc. of STOC, ACM, 2009.
[28] C. Gentry, S. Halevi, Implementing gentry's fully-homomorphic encryption scheme, in: Proc. of EUROCRYPT, Springer, 2011.
[29] A. Peter, Securely Outsourcing Computations using Homomorphic Encryption (Ph.D. thesis), Darmstadt University of Technology, Germany, 2013.
[30] Q. Wang, D. Zhou, Y. Li, Secure outsourced calculations with homomorphic encryption, 2018, CoRR. arXiv:1812.00599.
[31] X. Liu, R.H. Deng, P. Wu, Y. Yang, Lightning-fast and privacy-preserving outsourced computation in the cloud, Cybersecurity 3 (1) (2020) 17.
[32] E.G. Rieffel, J.T. Biehl, B. van Melle, A.J. Lee, Secured histories for presence systems, in: Proc. of CTS, IEEE, 2011.
[33] Q. Li, G. Cao, Efficient and privacy-preserving data aggregation in mobile sensing, in: Proc. of ICNP, IEEE Computer Society, 2012.
[34] L. Burkhalter, A. Hithnawi, A. Viand, H. Shafagh, S. Ratnasamy, TimeCrypt: Encrypted data stream processing at scale with cryptographic access control, in: Proc. of NSDI, USENIX Association, 2020.
[35] C. Castelluccia, A.C. Chan, E. Mykletun, G. Tsudik, Efficient and provably secure aggregation of encrypted data in wireless sensor networks, ACM Trans. Sens. Netw. 5 (3) (2009) 20:1–20:36.
[36] Q. Li, G. Cao, Providing efficient privacy-aware incentives for mobile sensing, in: Proc. ICDCS, IEEE Computer Society, 2014.
[37] J. Katz, Y. Lindell, Introduction to Modern Cryptography, third ed., CRC Press, 2021.
[38] R. P. Protection, RFC 5246, Internet Engineering Task Force.
[39] Intel software guard extensions (intel SGX), 2021, https://software.intel.com/en-us/sgx (accessed 28 August 2021).
[40] A. Consortium, ARM trustzone, 2021, https://www.arm.com/products/security-on-arm/trustzone (accessed 28 August 2021).
[41] Advanced microchip devices, 2021, https://developer.amd.com/sev/ (accessed 28 August 2021).
[42] G. Kaptchuk, M. Green, I. Miers, Giving state to the stateless: Augmenting trustworthy computation with ledgers, in: Proc. of NDSS, The Internet Society, 2019.
[43] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N.M. Johnson, A. Juels, A. Miller, D. Song, Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts, in: Proc. of EuroS&P, IEEE, 2019.
[44] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, M. Naor, Our data, ourselves: Privacy via distributed noise generation, in: Proc. of EUROCRYPT, Springer, 2006.
[45] V. Rastogi, S. Nath, Differentially private aggregation of distributed time-series with transformation and encryption, in: Proc. of SIGMOD, ACM, 2010.
[46] E. Shi, T.H. Chan, E.G. Rieffel, R. Chow, D. Song, Privacy-preserving aggregation of time-series data, in: Proc. of NDSS, The Internet Society, 2011.
[47] T.H. Chan, E. Shi, D. Song, Privacy-preserving stream aggregation with fault tolerance, in: Proc. of FC, Vol. 7397, Springer, 2012, pp. 200–214.
[48] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, in: Proc. of TCC, Springer, 2011.

Mei Wang received the M.S. degree in cryptography from Xidian University, Xi'an, China, in 2016. She is currently working toward the Ph.D. degree in cyberscience, Wuhan University, Wuhan, China. She was a visiting Ph.D students with Lancaster University from Sep 2018 to Sep 2019. Her research interests include applied cryptography, cloud computing, mobile computing, and privacy protection.

Kun He received the Ph.D. degree in computer science from Wuhan University, Wuhan. He has published research papers in the FUTURE GENERATION COMPUTER SYSTEM, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE IOT, the USENIX security and INFOCOM. His research interests include cryptography, network security, mobile computing, and cloud computing.

Jing Chen received the PhD degree in computer science from the Huazhong University of Science and Technology, Wuhan. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, Wuhan. He has published over 90 research papers in many international journals and conferences, such as the FUTURE GENERATION COMPUTER SYSTEM, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE IOT, the USENIX Security and INFOCOM. His research interests are in the areas of network security and cloud security.

Ruiying Du received the B.S., M.S., and Ph.D. degrees in computer science from Wuhan University, Wuhan, China, in 1987, 1994, and 2008, respectively. She is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. She has published over 80 research papers in many international journals and conferences, such as the FUTURE GENERATION COMPUTER SYSTEM, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE IoT, the USENIX Security, INFOCOM, SECON, and TrustCom. Her research interests include network security, wireless network, and mobile computing.

Bingsheng Zhang is a ZJU100 Professor at Zhejiang University. Before that, he was the programme director of Msc. Cyber security and security group leader at Lancaster University from 2015 to 2019. He has published in peer-reviewed security/cryptography conferences and journals, such as EUROCRYPT, ASIACRYPT, ACM CCS, PKC, ICDCS, FC, PODC, IEEE Security & Privacy, etc. In the recent years, his research efforts focus on secure computing, verifiable electronic voting and blockchain security. He is currently running several projects with EPSRC and PETRAS grants on blockchain security and e-voting security.

Zengpeng Li is currently a faculty member of Shandong University (SDU), Qingdao Campus, China. Before joining SDU, he was a postdoctoral research fellow at the Singapore University of Technology and Design (SUTD) and a postdoctoral research associate at Lancaster University. He received his Ph.D. degree from Harbin Engineering University (HEU), China. His primary research interests are in cryptographic protocol and secure distributed computing.