Information Sciences 514 (2020) 118–130

Contents lists available at ScienceDirect

Information Sciences

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

A privacy-preserving data aggregation scheme for dynamic groups in fog computing

Xiaodong Shen[a], Liehuang Zhu[a], Chang Xu[a,*], Kashif Sharif[a], Rongxing Lu[b]

[a] Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[b] Faculty of Computer Science, University of New Brunswick USA

A R T I C L E I N F O

Article history:
Received 27 March 2019
Revised 26 November 2019
Accepted 2 December 2019
Available online 3 December 2019

Keywords:
Fog computing
Data aggregation
Dynamic groups
Privacy preservation

A B S T R A C T

Fog computing has garnered significant attention in recent years, since it can bridge the cloud and terminal devices and provide low latency, location awareness, and geo-distribution at the edge of the network. Data aggregation is a prime candidate for fog computing applications. However, most previous works about data aggregation do not focus on the fog computing. In addition, existing secure data aggregation schemes in fog computing usually do not support dynamic groups and arbitrary aggregation functions. In this paper, we construct concrete data encryption, data aggregation and data decryption algorithms, and further propose a privacy-preserving and collusion-resistant data aggregation scheme for dynamic groups in fog computing. Specifically, in the proposed protocol, the cloud server can periodically collect raw data and compute arbitrary aggregation functions on them. Even if some malicious terminal devices collude with the fog device or the cloud server, the honest terminal devices' privacy cannot be breached. The fog device can filter out false data and aggregate all terminal devices' ciphertexts to save the bandwidth. Besides, dynamic join and exit of terminal devices is achieved. Detailed security analysis shows that our scheme holds *k*-source anonymity. Our scheme is also demonstrated to be efficient via extensive experiments.

1. Introduction

The Internet of Things (IoT) is moving from far-fetched to reality, and has received wide attentions. IoT interconnects a number of ordinary physical objects and enables a new breed of applications. However, the explosive growth of IoT devices generates massive data for analytics, which requires lots of communication, computation, and storage resources. The "pay-as-you-go" cloud computing paradigm is an economic and efficient alternative, since cloud computing leverages inexpensive power to provide elastic resources. However, many applications (e.g. gaming, augmented reality, connected vehicles, etc.) require support for low latency and location awareness, which usually cannot be provided by the cloud because of the long distances. To address this challenge, fog computing [2,24] was proposed to bridge the gap between the cloud and end users. Fog computing is a typically virtualized platform that locates at the network edge and provides networking, computation, and storage services between the cloud and terminal devices. The fog is characterized by capturing low latency, location awareness, and mobility since it is in the vicinity of end nodes.

* Corresponding author.
E-mail address: xuchang@bit.edu.cn (C. Xu).

https://doi.org/10.1016/j.ins.2019.12.007
0020-0255/© 2019 Elsevier Inc. All rights reserved.

Instead of entirely replacing cloud computing, the fogs' tasks are to extend the Cloud Computing paradigm to the edge of the network and assist the cloud in processing data. In this work, we focus on the data aggregation in fog computing environments. Assuming that there is a group of terminal devices with sensing, computation, and communication capabilities in some area [25,42], we consider the following scenario: the cloud server needs to collect data from the terminal devices periodically, and use the data to get statistical results. Fog devices are deployed at the network edge to help the cloud server to collect, preprocess, and aggregate the messages. For example, when the cloud server wants to monitor the temperature data, a group of relative terminal devices periodically collect data and forward it to a fog device. After receiving the aggregated data from the fog devices, the cloud server can perform statistical analysis on them.

However, ensuring privacy and security of data aggregation in fog computing remains an open challenge. On the one hand, the data collected by the terminal devices should be protected. That is, the raw data should not be recovered by any party except for the cloud server. On the other hand, malicious terminal devices may collude with the cloud server or fog devices, therefore, collusion attacks should be resisted. Some existing literature has proposed data aggregation schemes in wireless sensor networks [12,19], vehicle sensing systems [33], smart grids [14], crowdsensing [9,10,15,23]. Some works also focus on secure data collection in different environments [22,36,43]. However, these works cannot be applied to fog computing environments. Wang et al. [31] utilized the Castagnos-Laguillaumie cryptosystem to enable data aggregation in fog computing. However, their work only supports sum aggregation. Though Lu et al. [17] proposed a data aggregation scheme in which the cloud server can compute mean and variance, the raw data cannot be recovered and dynamic groups are not considered.

Aiming at the above challenges, we propose a novel privacy-preserving data aggregation scheme in fog computing environments. After each terminal device finishes sensing data, it uses the timestamp of current period as the public parameter of a pseudo-random function (PRF) to generate a set of pseudo-random bit strings to encrypt data. When all the ciphertexts are collected by fog devices, the fog devices preprocess and aggregate them and send the aggregation results to the cloud server. Finally, the cloud server recovers all the raw data and computes arbitrary functions on the data. Our contributions can be summarized as follows:

- We construct concrete data encryption, data aggregation and data decryption algorithms, and further propose a novel data aggregation protocol to help the cloud server to collect all the raw data. Thus, the cloud server can compute any statistical functions on the data. Besides, all the raw data can only be obtained by the cloud server. If there are false data found, the cloud server can trace and revoke the malicious terminal devices with the help of the trusted authority.
- Our scheme can support dynamic join and exit of terminal devices. Specifically, if join or exit events happen, terminal devices do not require to improve their key materials. In addition, the communications among terminal devices are not needed throughout the execution of the protocol.
- Privacy preservation is achieved in our scheme. That is, the cloud server cannot link the data with their owners. The terminal devices' real identities are protected by exploiting pseudonyms. Even if some malicious terminal devices collude with the fog device or the cloud server, they still cannot breach any honest terminal device's privacy. Besides, our scheme also holds data integrity and authentication.
- Security analysis illustrates that our scheme captures k-source anonymity. We not only show the communication cost and computation cost through theoretical analysis, but also demonstrate the efficiency by extensive experiments.

The reminder of this paper is organized as follows. In Section 2, we provide the system model, security model, and design goals. We propose our novel data aggregation protocol in Section 3, give the security analysis in Section 4, show the experiment results in Section 5, and discuss the related work in Section 6. Finally, we conclude our work in Section 7.

2. Models and design goals

In this section, we formulate the system model, the security model and identify the design goals.

2.1. System model

The system comprises of four kinds of entities: a cloud server, fog devices, terminal devices, and a trusted authority. The data sensed by terminal devices is first sent to the fog devices, then processed and relayed to the cloud server. Assume each sensing data is represented by a bit string. The communication channels between terminal devices and the fog devices are cellular networks, WiFi, or other available networks. Note that only one-way communication channels are required from terminal devices to the fog devices and the fog devices to the cloud server. The system model is shown in Fig. 1.

- Trusted Authority (TA): TA is responsible for bootstrapping the system, generating keys, and distributing keys to the cloud server, the fog device, and all the terminal devices. When the system is initialized, TA will be offline unless a terminal device joins or leaves the system or the cloud server requires to trace malicious terminal devices.
- Cloud Server (CS): CS owns large storage space and strong computation ability. CS's duty is to define and release tasks to all the terminal devices, including the time periods, locations, data types, and so on. After receiving the messages from the fog, CS processes the aggregated messages to obtain raw data. Then, CS computes statistical functions based on the raw data, and provides the results to customers. When CS discovers invalid data, it can collaborate with TA to identify the malicious terminal devices.

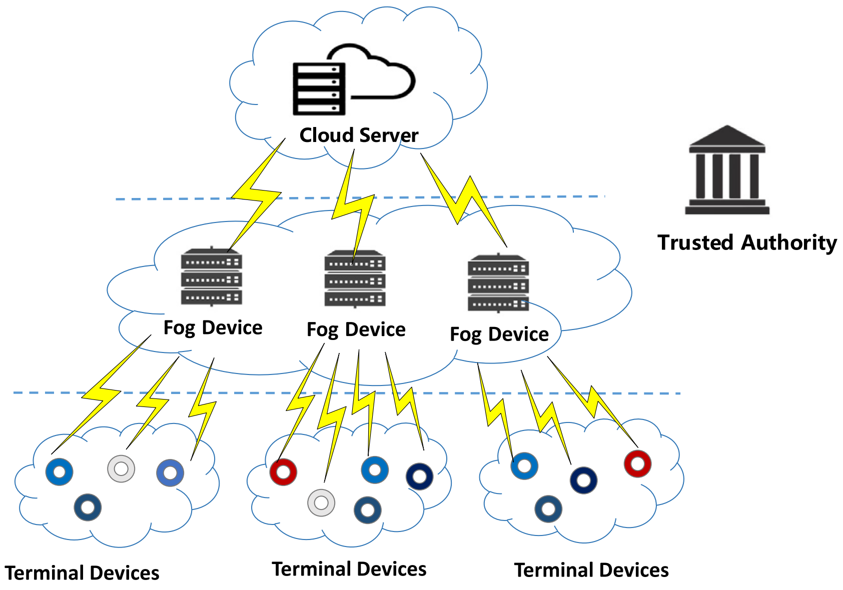

Fig. 1. System model.

- Fog Device (FD): FDs resemble lightweight servers that are deployed at the network edge. Each FD is equipped with storage and computation capability and can communicate with CS and terminal devices directly. In addition, each FD receives messages from all the relevant terminal devices, aggregates and transmits the processed messages to CS.
- Terminal Device (TD): The TDs are smartphones, pads, vehicles, and other kinds of terminals with sensing, computation, and communication capabilities. Assuming that there are N TDs in the system, $\mathcal{TD} = \{TD_1, TD_2, \ldots, TD_N\}$. The TDs periodically perform the sensing tasks according to the requirements of CS, then TDs transfer the messages to the FDs.

2.2. Security model

Similar to the schemes in [8,17,31,33], in our security model, we consider that TA is fully trusted and cannot be compromised. TA distributes keys to CS, FDs, and TDs through secure channels. All the TDs honestly follow the protocol specification. However, the TDs may try to directly disclose and get others' sensing data instead of sensing data by themselves. The reason is that sensing data will consume their batteries and computation resource. Additionally, malicious TDs may deliberately upload invalid data to CS to lead to inaccurate aggregation results. FD is willing to recover all the sensing data and tries to link the raw data with TDs. CS can obtain all the processed data from FDs and tries to link the raw data with their owners. We assume that a part of malicious TDs may collude with FDs or CS. However, similar to the model in [17], CS and FDs will not collude with each other.

Here, we introduce the definition of k-source anonymous, since our protocol can achieve k-source anonymity. A user's privacy is defined based on the anonymity level of the data. Specifically, if the best an adversary can learn is that the source of its data is one of k users ($k \in \{1, \ldots, n\}$), we say this user's data has k-source anonymity or the user has a privacy level of k. Naturally, the larger k's value is, the better the corresponding privacy is.

k-source anonymous: A data aggregation process or a protocol is k-source anonymous, if for any group U, with any two users u_i and u_j in it, $P(..., u_i(m_i), ..., u_j(m_j), ...) \overset{c}{\equiv} P(..., u_i(m_j), ..., u_j(m_i), ...)$ is satisfied, where k denotes the number of users, $\{m_1, m_2, ..., m_k\} \in \{M\}^k$ denotes the data aggregation sample, M denotes the message space, $P(..., u_i(m_i), ...)$ denotes the data aggregator's view when running the protocol with $x_i(x_i \in \{m_1, m_2, ..., m_k\})$ as u_i's input ($i = 1, 2, \ldots, k$), and $\overset{c}{\equiv}$ denotes computational indistinguishability of two random variable ensembles.

2.3. Design goals

Under the proposed system model and security model, the following design goals should be captured.

- *Privacy preservation:* TDs' privacy should be protected. Assume N TDs send sensing data to a fog device, and the fog device sends the processed data to the cloud. On the one hand, though FDs can know the relationships between all TDs' pseudonyms and their ciphertexts, they cannot decrypt the ciphertexts. On the other hand, although CS can obtain all the pseudonyms and the raw data, the probability that CS can link a pseudonym with its data should be no more than $1/N$. Any honest TD's data can only be decrypted by CS and cannot be obtained by FD and other TDs.

- *Efficiency and accuracy:* TDs, FDs, and CS should execute the protocol efficiently, so that the proposed scheme can be applied in practical scenarios. CS can recover all the data and compute any aggregation function based on them. When invalid data is discovered, TA can trace the malicious TDs.
- *Scalability:* Dynamic join and exit of TDs should be considered. That is, if join or exit events happen, the TDs in the system need not improve their keys. Besides, the system should be easily extended to larger areas by adding FDs and TDs. Moreover, the revoked TDs should not be able to disclose any valuable information.
- *Collusion resistance:* Even if malicious TDs collude with FD or CS, the malicious parties cannot recover honest TDs' secret keys and breach their privacy.

3. Proposed privacy-Preserving data aggregation scheme

In this section, we present the privacy-preserving and collusion-resistant data aggregation scheme for dynamic groups.

3.1. Overview

Our scheme consists of five parts: **System Initialization, Terminal Device Data Collection, Fog Device Data Verification and Aggregation, Cloud Server Data Decryption**, and **Join and Exit of Terminal Devices**.

The **System Initialization** algorithm initializes the system and generates key materials for each party. The sensing data is encrypted and signed in the **Terminal Device Data Collection** phase. To show the scheme clearly, we consider there exists only one FD in the system. FD verifies and aggregates all the messages in the **Fog Device Data Verification and Aggregation** phase. Finally, CS recovers all the raw data in the **Cloud Server Data Decryption** phase. In addition, we describe how to deal with join and exit of TDs in the **Join and Exit of Terminal Devices** algorithm. The notations used in this paper are listed in Table 1. The main steps of our scheme are shown in Fig. 2.

3.2. System initialization

This phase consists of *Parameter Generation* and *Key Generation* algorithms. The *Parameter Generation* algorithm outputs system parameters and the *Key Generation* algorithm generates keys for TDs, FD, and CS.

Parameter Generation: Given the security parameter λ, TA generates two cyclic groups G_1 and G_2 of the prime order p, where $|p| = \lambda$. The generators g_1 and g_2 are chosen from G_1 and G_2, respectively. e is a bilinear map e: $G_1 \times G_2 \to G_T$ with the properties of bilinearity, non-degeneracy, and computability, where $|G_1| = |G_2| = |G_T|$. TA also chooses one full-domain hash function H: $\{0, 1\}^* \to G_1$ and one PRF $f_s(x)$ indexed by the key s:

$$f_s(x) : \mathcal{H}_{l, l_t + \lceil log_2 N_{max} \rceil, l} = \{f_s : \{0,1\}^{l_t + \lceil log_2 N_{max} \rceil} \to \{0,1\}^l\}_{s \in \{0,1\}^l},$$

where l and l_t are the size of sensing data and the size of timestamp t, respectively. $N_{\max}$ is the upper bound of N. Note that $f_s(x)$ is in the PRF family $\mathcal{H}$ indexed by s, where s is set to the same size as that of the message here. We emphasize that the size of the message and s can be different and chosen according to the security requirements.

The system parameters are (λ, p, G_1, G_2, g_1, g_2, G_T, e, H, $f_s(x)$).

Key Generation: Given the system parameters, TA generates keys for TDs, FD, and CS as follows:

- TD: For each TD_i, $i \in [1, N]$, generate its pseudonym PID_i and corresponding signing key $pri\_k_i = s_i$, where s_i is randomly picked from $\mathbb{Z}_p$. The verification key pub_k_i is computed as $pub\_k_i = g_2^{s_i}$. The Verification key list is shown in Table 2. TD_i's secret key that is used for encryption is randomly chosen as $sk_i =< skf_i, skc_i >$, where $|skf_i| = |skc_i| = \lambda$.

Table 1
Notations.

Notations	Descriptions
λ	Security parameter
N	Number of TDs in the protocol
$N_{\max}$	Upper bound of N
t	Timestamp of each time period
TD_i	ith terminal device
PID_i	TD_i's pseudonym
sk_i	TD_i's secret key, $sk_i =< skf_i, skc_i >$
m_i	TD_i's raw data
l	Length of raw data
l_t	Length of timestamp t
ID_{FD}	FD's identity
sk_{FD}	FD's secret key, $sk_{FD} = \{skf_1, skf_2, \ldots, skf_N\}$
sk_{CS}	CS's secret key, $sk_{CS} = \{skc_1, skc_2, \ldots, skc_N\}$
H	A full-domain hash function, H: $\{0, 1\}^* \to G_1$
$f_s(x)$	A function indexed by s in a pseudo-random function family, $\mathcal{H}_{l, l_t + \lceil log_2 N_{max} \rceil, l} = \{f_s : \{0,1\}^{l_t + \lceil log_2 N_{max} \rceil} \to \{0,1\}^l\}_{s \in \{0,1\}^l}$
C_i	The ciphertext generated by TD_i

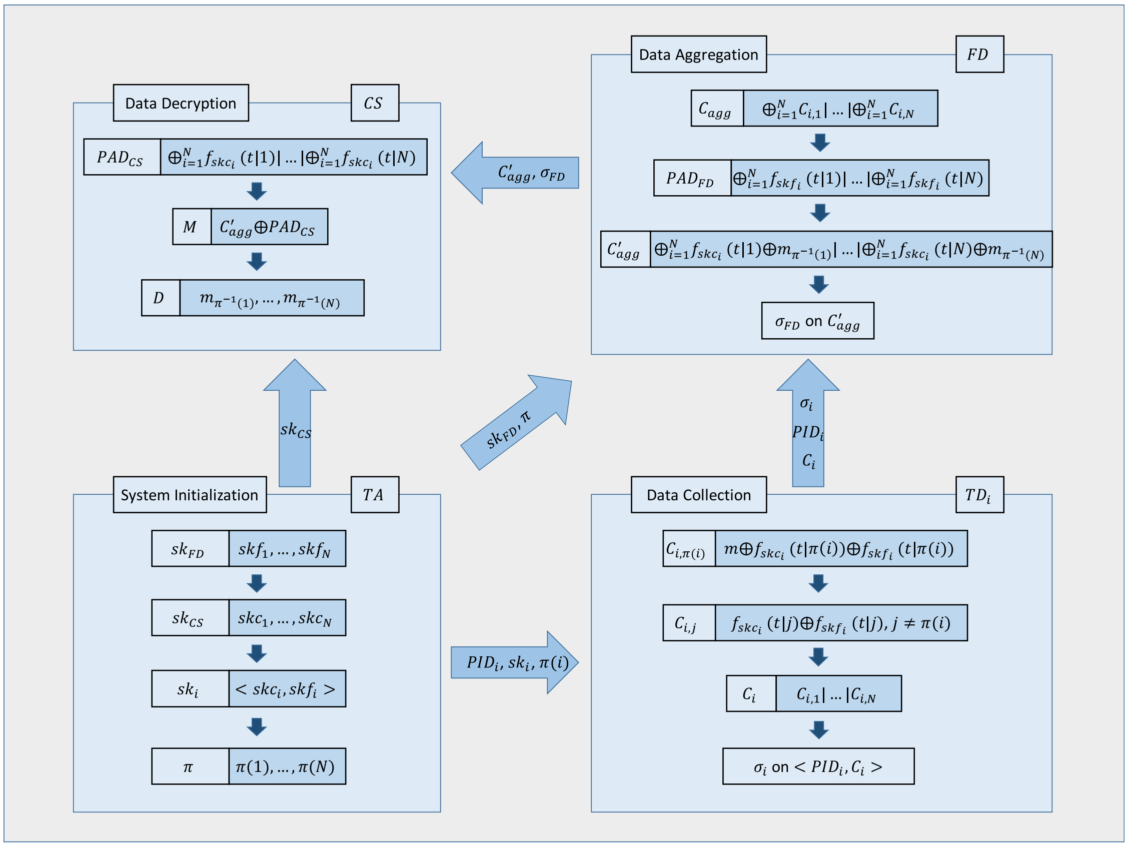
Data Decryption
CS
PAD_{CS} | $\oplus_{i=1}^{N} f_{skc_i}(t|1)|\ldots|\oplus_{i=1}^{N} f_{skc_i}(t|N)$
M | $C'_{agg} \oplus PAD_{CS}$
D | $m_{\pi^{-1}(1)}, \ldots, m_{\pi^{-1}(N)}$
C'_{agg}, σ_{FD}
Data Aggregation
FD
C_{agg} | $\oplus_{i=1}^{N} C_{i,1}|\ldots|\oplus_{i=1}^{N} C_{i,N}$
PAD_{FD} | $\oplus_{i=1}^{N} f_{skf_i}(t|1)|\ldots|\oplus_{i=1}^{N} f_{skf_i}(t|N)$
C'_{agg} | $\oplus_{i=1}^{N} f_{skc_i}(t|1) \oplus m_{\pi^{-1}(1)}|\ldots|\oplus_{i=1}^{N} f_{skc_i}(t|N) \oplus m_{\pi^{-1}(N)}$
σ_{FD} on C'_{agg}
sk_{CS}
sk_{FD}, π
σ_i PID_i C_i
System Initialization
TA
sk_{FD} | $skf_1, \ldots, skf_N$
sk_{CS} | $skc_1, \ldots, skc_N$
sk_i | $< skc_i, skf_i >$
π | $\pi(1), \ldots, \pi(N)$
$PID_i, sk_i, \pi(i)$
Data Collection
TD_i
$C_{i,\pi(i)}$ | $m \oplus f_{skc_i}(t|\pi(i)) \oplus f_{skf_i}(t|\pi(i))$
$C_{i,j}$ | $f_{skc_i}(t|j) \oplus f_{skf_i}(t|j), j \neq \pi(i)$
C_i | $C_{i,1}|\ldots|C_{i,N}$
σ_i on $< PID_i, C_i >$

Fig. 2. Work flow of the scheme.

Table 2
Verification key list.

PIDs	Verification keys
PID_1	$g_2^{s_1}$
PID_2	$g_2^{s_2}$
PID_3	$g_2^{s_3}$
...	...
PID_N	$g_2^{s_N}$

- FD: FD's secret key is $sk_{FD} = \{skf_1, skf_2, \ldots, skf_N\}$. TA also generates the identity ID_{FD} for FD, the signing key $pri\_k_{FD} = s_0 \overset{R}{\leftarrow} \mathbb{Z}_p$, and the verification key pub_k_{FD} where $pub\_k_{FD} = g_2^{s_0}$.
- CS: CS's secret key is $sk_{CS} = \{skc_1, skc_2, \ldots, skc_N\}$.

In addition, CS publishes the rule regarding the value of t in each period, e.g., t can be set to the timestamp of the beginning of each period, where each period can be 30 minutes. This means that each TD performs the task every half an hour. TA generates a permutation of $\{1, 2, \ldots, N\}$ as $\pi = \{\pi(1), \pi(2), \ldots, \pi(N)\}$ and assigns each $\pi(i)$ to TD_i. To state clearly in the following phases, if $\pi(i) = j$, then $i = \pi^{-1}(j)$, which means that j is assigned to $TD_{\pi^{-1}(j)}$. Note that π should be changed after random periods.

3.3. Terminal device data collection

After TD_i collects the sensing data m_i, where the length of m_i is l bits, TD_i executes the *Data Encryption* algorithm to encrypt m_i and the *Message Signing* algorithm to sign the ciphertext.

Data Encryption: Given TD_i's data m_i and secret key sk_i, m_i is encrypted as follows:

- Step 1: Compute

$$c_{i,\pi(i)} = m_i \oplus f_{skc_i}(t|\pi(i)) \oplus f_{skf_i}(t|\pi(i)).$$

- Step 2: For $j = 1, 2, \dots, N$ and $j \neq \pi(i)$, compute

$$c_{i,j} = f_{skc_i}(t|j) \oplus f_{skf_i}(t|j).$$

- Step 3: Concatenate $c_{i,j}$ sequentially to generate the ciphertext $C_i = c_{i,1}|c_{i,2}|\dots|c_{i,N}$, where the size of C_i is $N*l$ bits.

Message Signing: Given TD_i's ciphertext C_i and the signing key $pri\_k_i = s_i$, the signature on C_i is computed as $\sigma_i = H^{s_i}(PID_i|C_i)$ according to the aggregate signature algorithm in [1].

Finally, TD_i sends $M_i =< PID_i, C_i, \sigma_i >$ to FD.

3.4. Fog device data verification and aggregation

After FD collects $\{M_i\}_{i=1,2,\dots,N}$ from all the TDs, it executes the *Message Verification* algorithm to check the validity of all signatures. Following this, *Data Aggregation* algorithm is initiated to process and aggregate all the ciphertexts.

Message Verification: Input $\{M_i\}_{i=1,2,\dots,N}$ and each TD's verification key, FD runs the following steps to verify all signatures.

- Step 1: Compute $\sigma_{agg} = \prod_{i=1}^{N} \sigma_i$.
- Step 2: For $i = 1, 2, \dots, N$, compute $H_i = H(PID_i|C_i)$.
- Step 3: Check $e(\sigma_{agg}, g_2) \overset{?}{=} \prod_{i=1}^{N} e(H_i, pub\_k_i)$, output "*Valid*" if the equation holds, otherwise output "*Invalid*".

If the *Message Verification* algorithm outputs "*Valid*", FD continues to execute the *Data Aggregation* algorithm, otherwise it finds out the invalid data and requests a retransmission.

Data Aggregation: Input $\{C_i\}_{i=1,2,\dots,N}$, FD's secret key sk_{FD}, and the signing key pri_k_{FD}, FD executes the following steps:

- Step 1: Aggregate all ciphertexts as follows:

$$\begin{aligned} C_{agg} &= C_1 \oplus C_2 \oplus \dots \oplus C_N \\ &= \oplus_{i=1}^{N} c_{i,1} | \oplus_{i=1}^{N} c_{i,2} | \dots | \oplus_{i=1}^{N} c_{i,N}. \end{aligned}$$

  where $\oplus_{i=1}^{N} x_i$ is defined as $\oplus_{i=1}^{N} x_i = x_1 \oplus x_2 \oplus \dots \oplus x_N$.
- Step 2: Use sk_{FD} to calculate a one-time pad

$$PAD_{FD} = \oplus_{i=1}^{N} f_{skf_i}(t|1) | \oplus_{i=1}^{N} f_{skf_i}(t|2) | \dots | \oplus_{i=1}^{N} f_{skf_i}(t|N).$$

- Step 3: Process the aggregated ciphertexts with PAD_{FD} and output the final result as:

$$\begin{aligned} C'_{agg} &= C_{agg} \oplus PAD_{FD} \\ &= (\oplus_{i=1}^{N} f_{skc_i}(t|1) \oplus m_{\pi^{-1}(1)}) | (\oplus_{i=1}^{N} f_{skc_i}(t|2) \oplus \\ & m_{\pi^{-1}(2)}) | \dots \quad | (\oplus_{i=1}^{N} f_{skc_i}(t|N) \oplus m_{\pi^{-1}(N)}). \end{aligned}$$

- Step 4: Compute the signature on C'_{agg} as:

$$\sigma_{FD} = H^{s_0}(ID_{FD}|C'_{agg}).$$

Finally, FD sends $M_{FD} =< ID_{FD}, C'_{agg}, \sigma_{FD} >$ to CS.

3.5. Cloud server data decryption

After CS receives M_{FD} from FD, it executes the *Data Decryption* algorithm to verify the signature and recover all TDs' sensing data.

Data Decryption: Given M_{FD}, FD's verification key pub_k_{FD}, and CS's secret key sk_{CS}, CS outputs all TDs' data by running the following steps:

- Step 1: Check if the following equation holds

$$e(\sigma_{FD}, g_2) \overset{?}{=} e(H(ID_{FD}|C'_{agg}), pub\_k_{FD}).$$

  If so, execute Step 2; Otherwise, output "*Invalid*".
- Step 2: Use sk_{CS} to calculate a one-time pad.

$$PAD_{CS} = \oplus_{i=1}^{N} f_{skc_i}(t|1) | \oplus_{i=1}^{N} f_{skc_i}(t|2) | \dots | \oplus_{i=1}^{N} f_{skc_i}(t|N).$$

- Step 3: Decrypt C'_{agg} with PAD_{CS}:

$$M = C'_{agg} \oplus PAD_{CS} = m_{\pi^{-1}(1)} | m_{\pi^{-1}(2)} | \dots | m_{\pi^{-1}(N)}.$$

- Step 4: Divide M into N l-bit strings and output all data as:

$$\mathcal{D} = \{m_{\pi^{-1}(1)}, m_{\pi^{-1}(2)}, \dots, m_{\pi^{-1}(N)}\}.$$

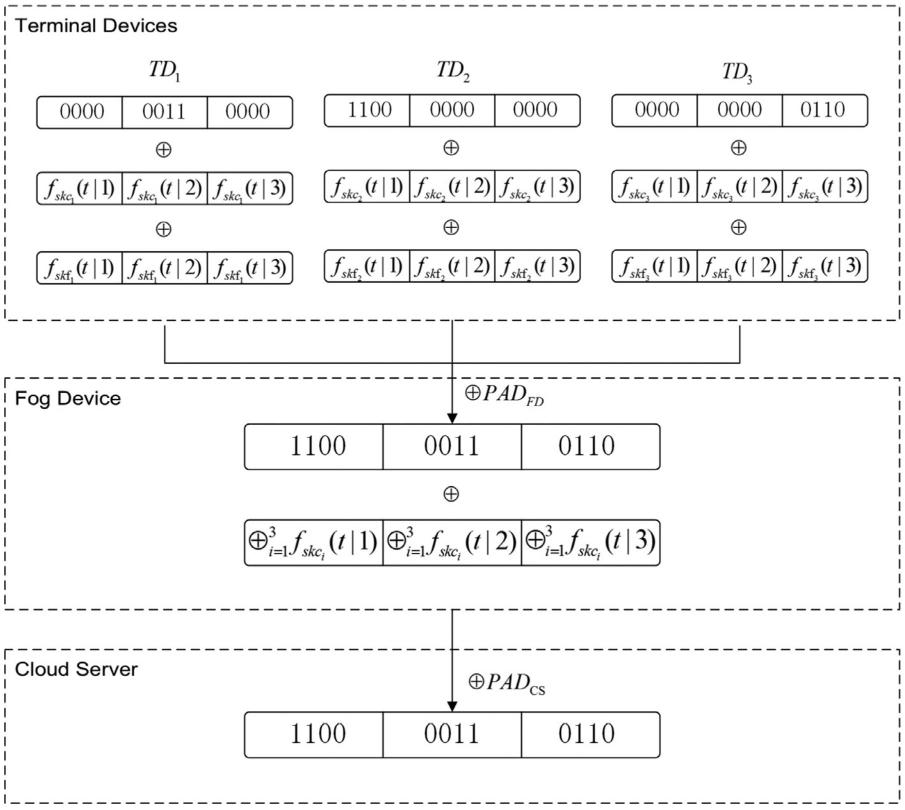
Terminal Devices
TD_1
0000 0011 0000
$f_{skc_1}(t|1)$ $f_{skc_1}(t|2)$ $f_{skc_1}(t|3)$
$f_{skf_1}(t|1)$ $f_{skf_1}(t|2)$ $f_{skf_1}(t|3)$
TD_2
1100 0000 0000
$f_{skc_2}(t|1)$ $f_{skc_2}(t|2)$ $f_{skc_2}(t|3)$
$f_{skf_2}(t|1)$ $f_{skf_2}(t|2)$ $f_{skf_2}(t|3)$
TD_3
0000 0000 0110
$f_{skc_3}(t|1)$ $f_{skc_3}(t|2)$ $f_{skc_3}(t|3)$
$f_{skf_3}(t|1)$ $f_{skf_3}(t|2)$ $f_{skf_3}(t|3)$
Fog Device
$\oplus PAD_{FD}$
1100 0011 0110
$\oplus_{i=1}^{3} f_{skc_i}(t|1)$ $\oplus_{i=1}^{3} f_{skc_i}(t|2)$ $\oplus_{i=1}^{3} f_{skc_i}(t|3)$
Cloud Server
$\oplus PAD_{CS}$
1100 0011 0110

Fig. 3. An example of our protocol.

If CS finds false data in $\mathcal{D}$, e.g., $m_{\pi^{-1}(j)}$, $j \in [1, N]$, CS requires TA to be involved and it sends $\pi^{-1}(j)$ to TA. TA can find out the malicious TD TD_i if $\pi(i) = j$ according to the permutation π.

Here we give a brief example to illustrate our protocol in Fig. 3. Assuming that there are three TDs in the system and their data are $m_1 = 0011$, $m_2 = 1100$, and $m_3 = 0110$, respectively. The permutation generated by TA is $\pi = \{2, 1, 3\}$. Each TD encrypts data with its secret key according to the permutation. After all the ciphertexts are aggregated and processed by FD, CS can recover all the raw data.

3.6. Join and exit of terminal devices

When TD_i is revoked, TA sends skf_i and skc_i to FD and CS respectively, and notifies them to revoke the corresponding secret key. If a new TD TD_i joins the system, TA generates the secret key $sk_i =< skf_i, skc_i >$ and the verification/signing key $< pub\_k_i, pri\_k_i >$ for it and assigns skf_i and skc_i to FD and CS, respectively. The permutation π is also updated accordingly.

Remark 1. We can observe that if join or exit events happen, current TDs do not need to improve their key materials. Therefore, the scheme supports dynamic groups.

4. Security analysis

In this section, we prove that our protocol can achieve the security properties described in Section 2 under the aforementioned security model. Specifically, we will elaborate that both FD and CS cannot break the k-source anonymity. According to the definition of k-source anonymous, we will prove that if any two TDs' data are interchanged in the same period, FD and CS cannot efficiently tell the difference.

Given a group of TDs $\mathcal{TD} = \{TD_1, TD_2, \ldots, TD_N\}$ and their sensing data $\mathcal{M} = \{m_1, m_2, \ldots, m_N\} \in \{M\}^N$ where $M = \{0, 1\}^l$, each TD TD_i executes the **Terminal Device Data Collection** algorithm to encrypt m_i and generate the ciphertext C_i. FD collects $\{C_1, C_2, \ldots, C_N\}$ and executes the **Fog Device Data Verification and Aggregation** algorithm to aggregate them into C'_{agg}. CS then receives C'_{agg} from FD.

For FD, although it can collect $\{C_1, C_2, \ldots, C_N\}$ from all TDs, it does not have the ability to recover any TD's raw data because of lack of CS's decryption keys. Therefore, FD cannot break k-source anonymity.

For CS, it can recover all TDs' data. Therefore, according to the definition of k-source anonymous, we will prove that if any two TDs' data are interchanged in the same period, CS cannot efficiently tell the difference. Besides, all the knowledge CS can learn is the aggregated ciphertext:

$$\begin{aligned} V &= C'_{agg} \\ &= (\oplus_{i=1}^{N} f_{skc_i}(t|1) \oplus m_{\pi^{-1}(1)})|(\oplus_{i=1}^{N} f_{skc_i}(t|2) \oplus m_{\pi^{-1}(2)})| \\ &\quad \ldots |(\oplus_{i=1}^{N} f_{skc_i}(t|N) \oplus m_{\pi^{-1}(N)}). \end{aligned}$$

Let $c_{0,j} = \oplus_{i=1}^{N} f_{skc_i}(t|j) \oplus m_{\pi^{-1}(j)}$ where $j \in [1, N]$, we have $V = c_{0,1}|c_{0,2}|\ldots|c_{0,N}$.

Let any two TDs TD_i and TD_j switch their data m_i and m_j where $1 \le i \le j \le N$, all the data are changed to $\mathcal{M}' = \{m_1, \ldots, m_{i-1}, m_j, m_{i+1}, \ldots, m_{j-1}, m_i, m_{j+1}, \ldots, m_N\}$. Then CS's knowledge is

$$V' = c_{0,1}|\ldots|c_{0,i-1}|c_{0,j}|c_{0,i+1}|\ldots|c_{0,j-1}|c_{0,i}|c_{0,j+1}|\ldots|c_{0,N}.$$

Therefore, to prove that our protocol captures k-source anonymity, it is equivalent to prove $V \overset{c}{\equiv} V'$ holds for any i, j where $1 \le i \le j \le N$, and $\mathcal{M} \in \{M\}^N$.

Here, we construct a simulator S to run the same protocol. First, S generates a pseudo-random permutation of $\{1, 2, \ldots, N\}$ as $\pi' = \{\pi'(1), \pi'(2), \ldots, \pi'(N)\}$. Then S permutes the original dataset $\mathcal{M}$ to be:

$$\mathcal{M}'' = \{m_{\pi'(1)}, m_{\pi'(2)}, \ldots, m_{\pi'(N)}\}.$$

Finally, S executes the protocol and outputs the aggregated ciphertext:

$$V'' = c_{0,\pi'(1)}|c_{0,\pi'(2)}|\ldots|c_{0,\pi'(N)}.$$

Because $\mathcal{M}$ and its pseudo-random permutation $\mathcal{M}''$ cannot be distinguished in polynomial time, V and V'' are computationally indistinguishable. Therefore, $V \overset{c}{\equiv} V''$ holds. Similarly, we have $V' \overset{c}{\equiv} V''$. Otherwise, $\mathcal{M}'$ and its pseudo-random permutation $\mathcal{M}''$ can be distinguished in polynomial time. Therefore, we have $V \overset{c}{\equiv} V'$. Thus, our protocol is k-source anonymous. □

In Section 3.6 of [11], the authors proved that their scheme is CPA-secure. In their scheme, the plaintext m is encrypted as $c = < r, f_{k_1}(r) \oplus m >$, where f is a pseudo-random function, and r is a random value. The proof follows a general paradigm with pseudorandom functions. First, the authors analyzed the security of the scheme in an idealized world where a truly random function is used in place of f, and showed that the scheme is secure. Then, they claimed that if the scheme is insecure when f was used, it would imply the possibility of distinguishing f from a truly random function.

Our scheme is also constructed based on pseudo-random functions, and m is encrypted as $c = m \oplus f_{k_1}(r) \oplus f_{k_2}(r)$ where $(k_1 \neq k_2)$. For our scheme, in an idealized world where a truly random function is used in place of f, our scheme is CPA secure. Otherwise, the adversary can distinguish $(f_{k_1}(r) \oplus f_{k_2}(r))$ from a truly random function. The detailed proof is similar to that in [11], so it is omitted here.

5. Performance evaluation

In this section, we provide the theoretical analysis, and also implement the system to evaluate the performance of CS, FD, and TD in terms of the communication cost and computation cost.

5.1. Implementation and experimental settings

The protocol is implemented on a test machine with Intel Core i7-4790 3.60GHz CPU, 8GB memory, and Windows 10 platform. We compile the program in Visual Studio 2013 and use the MIRACL library to implement the cryptographic functions.

The main security parameters in our protocol are the group size in the bilinear map and the hash functions. Here, we set the order of cyclic groups G_1 and G_2 as $|p| = 160$ bits and use HMAC-SHA512 as the PRF $f_s(x)$. Because HMAC-SHA512 always outputs a 512-bit block, it does not fit the setting of data length when the data length $l \neq 512$. Therefore, we adopt the same construction as that of [3]. When $l < 512$, we truncate the output of HMAC-SHA512 into several l-bit substrings and take XOR on these substrings as the final output. If $l > 512$, we use HMAC-SHA512 to generate several 512-bit blocks, and one shorter block to satisfy the condition of $l < 512$, and concatenate them into one string.

We take a uniform sample from $[0, 2^l - 1]$ as TDs' sensing data. In addition, we execute the proposed scheme 10 times and calculate the average running time. The PRF $f_s(x)$ is utilized to encrypt data at the TD side and decrypt data at the CS and FD side. We emphasize that $f_s(x)$ can be precomputed offline. Therefore, we only consider $f_s(x)$ in theoretical analysis.

5.2. TD's cost

In our protocol, let N denote the number of TDs and l represent the length of data. TD_i collects the l-bit data m_i, generates an Nl-bit ciphertext C_i on m_i, and computes the signature σ_i on C_i. To encrypt m_i, TD_i needs to perform $2N$ PRFs and $N + 1$

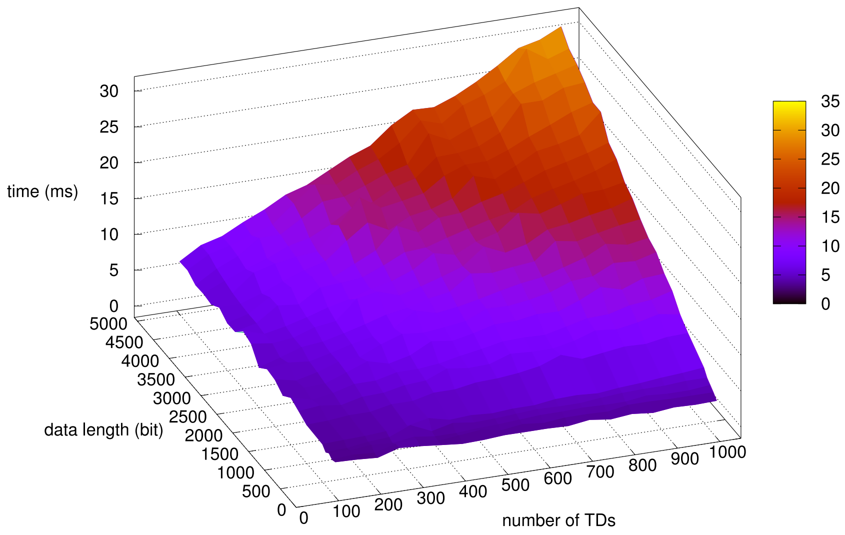

Fig. 4. Computational cost at the TD side.

XOR operations. One hash function and one exponentiation operation are also executed to generate the signature. Therefore, TD_i's theoretical computational cost is one hash function, $2N$ PRFs, one exponentiation, and $N+1$ XOR operations, where the XOR operations are performed on l-bit block. The communication cost is $Nl + |PID_i| + |\sigma_i|$ bits.

The computational cost of TD is shown in Fig. 4. Because $f_s(x)$ can be computed in advance, we do not consider the time spent in computing $2N$ PRFs here. In the experiment, the number of TDs N varies from 100 to 1000 and the length of data l varies from 100 bits to 5000 bits. From Fig. 4, we can see that the computation time increases linearly with l and N. The computational cost is 3.3 ms when $l = 100$ and $N = 100$ and rises to 6 ms when $l = 1000$ and $N = 500$. If N is fixed at 500, TD's computation time varies from 3.4 ms to 16.7 ms when l increases from 100 bits to 5000 bits. If we set l as 3000 bits, the computational cost is no more than 20 ms when the number of TDs increases to 1000. Even though l and N reach 5000 and 1000 respectively, TD's computation time is only 30.3 ms.

5.3. FD's cost

At the FD side, FD verifies all the TDs' signatures, aggregates the ciphertexts, and generates its own signature. FD executes $2(N-1)$ multiplications, N hash functions, and $N+1$ pair operations to verify all the signatures. In order to aggregate the ciphertexts, FD needs to perform N XOR operations on Nl-bit data, $N(N-1)$ XOR operations on l-bit data, and N^2 PRFs. In addition, FD performs one hash function and one exponentiation operation to compute the signature. Therefore, FD's theoretical computational cost is $N+1$ hash functions, N^2 PRFs, $N+1$ pair operations, $2(N-1)$ multiplications, one exponentiation operation, N XOR operations on Nl-bit data, and $N(N-1)$ XOR operations on l-bit data. The communication cost is $Nl + |\sigma_{FD}|$ bits.

Fig. 5 shows the computation time varying with the data length and the number of TDs. It takes much more time for FD to perform the protocol compared with TDs. If $N = 500$, the computation time increases from 3.286 s to 9.708 s when l varies from 500 to 5000. When the length of data l is 1000 bits, the computational cost rises from 0.671 s to 11.384 s when N increases from 100 to 1000. However, most computation time at the FD side is spent in verifying TDs' signatures. Specifically, the computation time and verification time are 4.525 s and 4.446 s when $l = 1000$ and $N = 500$. If we set $l = 5000$ and $N = 1000$, although FD consumes 32.511 s, it takes 30.895 s to verify all the signatures.

5.4. CS's cost

CS verifies FD's signature and decrypts the ciphertext to recover all the raw data. CS executes one hash function and one pair operation to verify the signature. Besides, CS performs N^2 PRFs, one XOR operation on Nl-bit data, and $N(N-1)$ XOR operations on l-bit data. Therefore, the theoretical computational cost of CS is one hash function, N^2 PRFs, one pair operation, one XOR operation on Nl-bit data, and $N(N-1)$ XOR operations on l-bit data.

We show CS's cost in Fig. 6. The computation time is proportional to the data length and the number of TDs. If $l = 100$ and $N = 100$, the computation time is 12.1 ms. If the number of TDs is fixed at 500, CS's computation time varies from 11.5 ms to 25.5 ms when l increases from 100 bits to 5000 bits. If we set the length of data as 1000 bits, the cost rises from 13.2 ms to 16.8 ms when the number of TDs increases from 100 to 1000. Moreover, it only takes 39.4 ms for CS in the **Cloud Server Data Decryption** phase when l and N increase to 5000 and 1000, respectively.

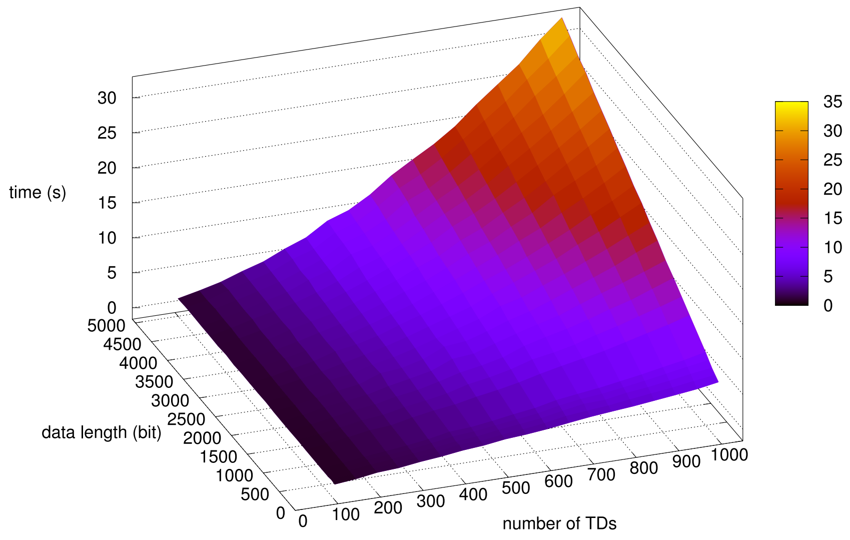

Fig. 5. Computational cost at the FD side.

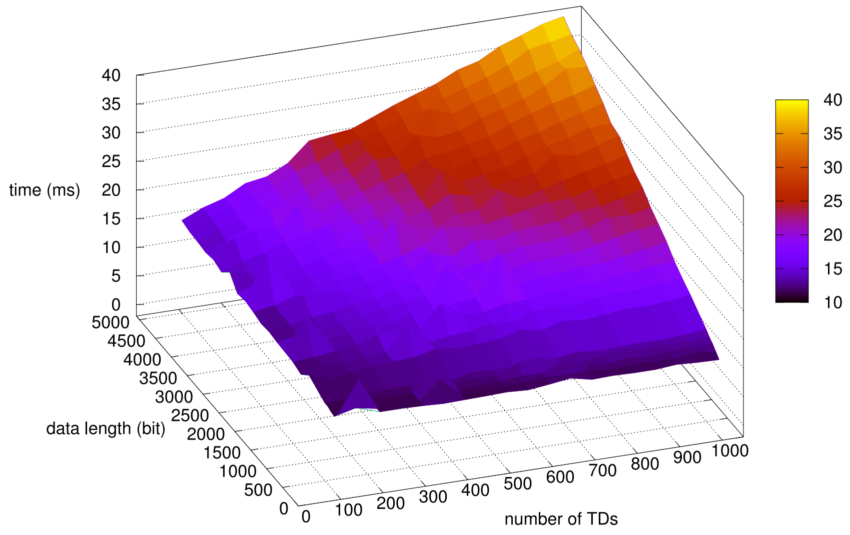

Fig. 6. Computational cost at the CS side.

6. Related works

In this section, we will review the relevant works about fog computing and the data aggregation which achieve different goals.

Bonomi et al. [2] introduced the concept of fog computing. They defined the fog's characteristics and gave some key applications of fog computing. The security issues and applications in fog computing have been discussed in [29,30,37,39]. Yi et al. [40] and Roman et al. [24] also introduced the challenges in security and privacy for fog computing. Besides, Mukherjee et al. [18] gave an overview of existing research on security and privacy in fog computing and provided the future research directions. Xu et al. [35] leveraged the fog node to deliver messages in a reliable manner to achieve data analytics in IoT. Wang et al. [31] proposed an anonymous and secure aggregation scheme based on the Castagnos-Laguillaumie cryptosystem and short signatures in fog computing environments. The terminal devices use the cloud's public key to encrypt data. After all the ciphertexts are aggregated by the fog device, the sum aggregation result can be recovered by the cloud. However, other functions cannot be computed. Lu et al. [17] proposed a novel lightweight data aggregation scheme. The Chinese Remainder Theorem and homomorphic Paillier encryption are utilized to aggregate data of hybrid IoT devices. Source authentication is also achieved at the network edge to filter false data in advance by employing one-way hash chain. Unfortunately, only some particular aggregation functions can be supported in their scheme. Liu et al. [16] presented secure intelligent traffic light control schemes in fog computing. Yu et al. [41] proposed a leakage-resilient functional encryption

Table 3
Comparison.

	Our scheme	The scheme in [31]	The scheme in [17]
TD side:	1 hash function, $2N$ PRFs, 1 exponentiation, and $N+1$ XOR operations on l-bit data.	2 hash functions, 2 multiplications, 1 exponentiation, 1 hash function, 2 pairing operations.	2 additions, 3 multiplications, 1 AES encryption, and 1 hash function.
FD side:	$N+1$ hash functions, N^2 PRFs, $N+1$ pair operations, $2(N-1)$ multiplications, 1 exponentiation operation, N XOR operations on Nl-bit data, and $N(N-1)$ XOR operations on l-bit data.	1 hash function, 4 pairing operations, $2(N-1)$ point additions, $2N-1$ multiplications.	N AES decryption, $3N+2$ hash functions, and N multiplications.
CS side:	1 hash function, N^2 PRFs, 1 pair operation, 1 XOR operation on Nl-bit data, and $N(N-1)$ XOR operations on l-bit data.	1 hash function, 2 bilinear pairings, 1 exponentiation, 1 inverse, 1 multiplication.	2 hash functions, $N+2$ multiplications, and N additions.

scheme. Yang et al. [38] designed a fine-grained query scheme based on k-nearest neighbors algorithm which achieves location privacy in fog environments. Wang et al. [32] employed fog nodes to generate dummy positions to protect the location information.

The following works also inspire us to construct our scheme in fog computing. Stergiou et al. [26] introduced data collection and processing applications and their basic features in Mobile Cloud Computing and IoT. The security issues of Mobile Cloud Computing and IoT are presented by Stergiou et al. [28]. Stergiou et al. [27] surveyed the security challenges of the integration of IoT and Cloud Computing, which helps us to build our security model. Plageras et al. [20] proposed solutions for collecting and managing sensors' data in a smart building, which operates in IoT environments. Psannis et al. [21] proposed an efficient algorithm to transfer high quality videos on Intelligent Cloud Computing systems.

Castelluccia et al. [3] proposed a novel additively homomorphic encryption scheme to implement additive aggregation operations in wireless sensor networks [4–7]. Each node uses a pseudo-random function to generate encryption keys in each session. In Castelluccia et al.'s scheme, the sink owns all nodes' secret keys and can use them to generate decryption keys. Unfortunately, the sink is assumed to be honest in their protocol.

Afterwards, Li et al. [13] presented a novel key system to guarantee the privacy of all users' data even if the aggregator colludes with some users. They adopted the homomorphic encryption scheme in [3] as a building block of their protocol. In their construction, each user is assigned to several secret keys. The number of keys is set according to the security requirements and the number of users in the group. Although the aggregator owns all the secret keys, it is hard to link each user with its secret keys if the total number of keys is large enough. In addition, their scheme further reduces the aggregator's computation overhead. However, the aggregator can only compute the sum function and min function, and the secure properties cannot be formally proven.

In order to support arbitrary aggregation functions in mobile phone sensing environments, Zhang et al. [44] proposed a privacy-preserving data aggregation scheme. Their scheme is designed based on the bitwise XOR homomorphic cipher system. Zhang et al. proved that the protocol holds k-source anonymity. However, the collusion attack is not considered in their scheme.

Xu et al. [34] proposed a novel data aggregation protocol in crowdsensing system to solve the problems which exist in [44]. Xu et al.'s scheme not only allows the aggregator to compute arbitrary aggregation functions while preserving users' privacy and resisting collusion attacks, but also safeguards the cloud's benefits by preventing other parties from decrypting the ciphertexts. Unfortunately, dynamic groups are not supported in their protocol.

Different from the above works, we propose a novel privacy-preserving data aggregation protocol in fog computing to allow the cloud server to compute arbitrary statistical functions and support dynamic join and exit of terminal devices. Our scheme is compared with the schemes in [31] and [17] in Table 3. In our scheme, though pair operations, exponentiations and multiplications are explored, these operations are only used to generate or verify the signatures. Moreover, any secure aggregate signature scheme can be utilized in our scheme. In [31], only the sum of the sensing data can be recovered by the cloud. In [17], only the mean and variance of the sensing data will be computed. Comparatively, in our scheme, CS can collect all the raw data and compute any statistical function on them. Besides, our scheme can resist collision attacks and support dynamic groups.

7. Conclusions

In this paper, we propose a novel privacy-preserving data aggregation protocol in fog computing. The proposed protocol allows CS to recover all the raw data and compute arbitrary aggregation functions while preserving the privacy of TDs. Even if some malicious TDs collude with FD or CS, the honest TDs' privacy cannot be breached. FD is leveraged to filter out the invalid signatures and aggregate ciphertexts to save the bandwidth. The aggregate signature is also employed to achieve data integrity and authentication. In addition, our protocol supports dynamic groups. The security analysis shows that our protocol holds k-source anonymity. Finally, we demonstrate the proposed scheme's efficiency via extensive simulations. In future, we will try to achieve privacy-preserving data aggregation without the assistance of the trusted parties. Besides, we will work on designing more efficient and privacy-preserving data aggregation schemes which support complex data types.

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.

CRediT authorship contribution statement

Xiaodong Shen: Investigation, Writing - original draft. **Liehuang Zhu:** Writing - review & editing. **Chang Xu:** Formal analysis, Writing - review & editing. **Kashif Sharif:** Writing - review & editing. **Rongxing Lu:** Formal analysis, Writing - review & editing.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (grant nos. 61972037, 61402037, U1804263).

References

[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps. in: Advances in cryptology - EUROCRYPT 2003, in: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, 2003, pp. 416–432. May 4-8, 2003, Proceedings.
[2] F. Bonomi, R.A. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC@SIGCOMM 2012, Helsinki, Finland, 2012, pp. 13–16. August 17, 2012.
[3] C. Castelluccia, A.C. Chan, E. Mykletun, G. Tsudik, Efficient and provably secure aggregation of encrypted data in wireless sensor networks, TOSN 5 (3) (2009). 20:1–20:36.
[4] X. Du, H.H. Chen, Security in wireless sensor networks, IEEE Wireless Commun. 15 (4) (2008) 60–66.
[5] X. Du, M. Guizani, Y. Xiao, H. Chen, Transactions papers a routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks, IEEE Trans. Wirel. Commun. 8 (3) (2009) 1223–1229.
[6] X. Du, Y. Xiao, H. Chen, Q. Wu, Secure cell relay routing protocol for sensor networks, Wirel. Commun. Mobile Comput. 6 (2006) 375–391.
[7] X. Du, Y. Xiao, M. Guizani, H. Chen, An effective key management scheme for heterogeneous sensor networks, Ad Hoc Netw. 5 (1) (2007) 24–34.
[8] H. Hu, R. Lu, Z. Zhang, J. Shao, REPLACE: a reliable trust-based platoon service recommendation scheme in VANET, IEEE Trans. Veh. Technol. 66 (2) (2017) 1786–1797.
[9] M. Joye, B. Libert, A Scalable Scheme for Privacy-preserving Aggregation of Time-series Data, in: Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1–5, 2013, Revised Selected Papers, 2013, pp. 111–125.
[10] T. Jung, X. Li, M. Wan, Collusion-tolerable privacy-preserving sum and product calculation without secure channel, IEEE Trans. Dependable Sec. Comput. 12 (1) (2015) 45–57.
[11] J. Katz, Y. Lindell, Introduction to Modern Cryptography, Chapman and Hall/CRC Press, 2007.
[12] C. Li, Y. Liu, SRDA: smart reputation-based data aggregation protocol for wireless sensor network, IJDSN 11 (2015). 105364:1–105364:10.
[13] Q. Li, G. Cao, T.F.L. Porta, Efficient and privacy-aware data aggregation in mobile sensing, IEEE Trans. Dependable Sec. Comput. 11 (2) (2014) 115–129.
[14] S. Li, K. Xue, Q. Yang, P. Hong, PPMA: privacy-preserving multisubset data aggregation in smart grid, IEEE Trans. Ind. Inform. 14 (2) (2018) 462–471.
[15] Y. Li, S. Liu, J. Wang, M. Liu, Collusion-tolerable and efficient privacy-preserving time-series data aggregation protocol, Int. J. Distrib. Sens. Netw. 12 (7) (2016) 1341606.
[16] J. Liu, J. Li, L. Zhang, F. Dai, Y. Zhang, X. Meng, J. Shen, Secure intelligent traffic light control using fog computing, Future Gener. Comp. Syst. 78 (2018) 817–824.
[17] R. Lu, K. Heung, A.H. Lashkari, A.A. Ghorbani, A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot, IEEE Access 5 (2017) 3302–3312.
[18] M. Mukherjee, R. Matam, L. Shu, L.A. Maglaras, M.A. Ferrag, N. Choudhury, V. Kumar, Security and privacy in fog computing: challenges, IEEE Access 5 (2017) 19293–19304.
[19] C.R. Perez-Toro, R.K. Panta, S. Bagchi, RDAS: reputation-based resilient data aggregation in sensor network, in: Proceedings of the Seventh Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2010, June 21–25, 2010, Boston, Massachusetts, USA, 2010, pp. 430–438.
[20] A.P. Plageras, K.E. Psannis, C. Stergiou, H. Wang, B.B. Gupta, Efficient iot-based sensor BIG data collection-processing and analysis in smart buildings, Future Gener. Comp. Syst. 82 (2018) 349–357.
[21] K.E. Psannis, C. Stergiou, B.B. Gupta, Advanced media-based smart big data on intelligent cloud systems, IEEE Trans. Sustainable Comput. 4 (1) (2018) 77–87.
[22] F. Qiu, F. Wu, G. Chen, SLICER: A slicing-based k-anonymous privacy preserving scheme for participatory sensing, in: IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems, MASS 2013, Hangzhou, China, 2013, pp. 113–121. October 14-16, 2013.
[23] F. Qiu, F. Wu, G. Chen, Privacy and quality preserving multimedia data aggregation for participatory sensing systems, IEEE Trans. Mob. Comput. 14 (6) (2015) 1287–1300.
[24] R. Roman, J. López, M. Mambo, Mobile edge computing, fog et&nbsp;al.: a survey and analysis of security threats and challenges, Future Gener. Comp. Syst. 78 (2018) 680–698.
[25] L. Shu, Y. Chen, Z. Huo, N. Bergmann, L. Wang, When mobile crowd sensing meets traditional industry, IEEE Access 5 (2017) 15300–15307.
[26] C. Stergiou, K.E. Psannis, Recent advances delivered by mobile cloud computing and internet of things for big data applications: a survey, Int. J. Netw. Manage. 27 (3) (2017).
[27] C. Stergiou, K.E. Psannis, B.B. Gupta, Y. Ishibashi, Security, privacy & efficiency of sustainable cloud computing for big data & iot, Sustain. Comput.: Inf. Syst. 19 (2018) 174–184.
[28] C. Stergiou, K.E. Psannis, B. Kim, B.B. Gupta, Secure integration of iot and cloud computing, Future Gener. Comp. Syst. 78 (2018) 964–975.
[29] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, in: Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, 2014, pp. 1–8. September 7-10, 2014.
[30] I. Stojmenovic, S. Wen, X. Huang, H. Luan, An overview of fog computing and its security issues, Concurr. Comput.: Pract.Exp. 28 (10) (2016) 2991–3005.

[31] H. Wang, Z. Wang, J. Domingo-Ferrer, Anonymous and secure aggregation scheme in fog-based public cloud computing, Future Gener. Comp. Syst. 78 (2018) 712–719.
[32] T. Wang, J. Zeng, M.Z.A. Bhuiyan, H. Tian, Y. Cai, Y. Chen, B. Zhong, Trajectory privacy preservation based on a fog structure for cloud location services, IEEE Access 5 (2017) 7692–7701.
[33] C. Xu, R. Lu, H. Wang, L. Zhu, C. Huang, PAVS: A new privacy-preserving data aggregation scheme for vehicle sensing systems, Sensors 17 (3) (2017) 500.
[34] C. Xu, X. Shen, L. Zhu, Y. Zhang, A collusion-resistant and privacy-preserving data aggregation protocol in crowdsensing system, Mobile Information Systems 2017 (2017). 3715253:1–3715253:11.
[35] Y. Xu, M. Veeramani, S. Radhakrishnan, Towards sdn-based fog computing: MQTT broker virtualization for effective and reliable delivery, in: 8th International Conference on Communication Systems and Networks, COMSNETS 2016, Bangalore, India, 2016, pp. 1–6. January 5-10, 2016.
[36] K. Xue, W. Chen, W. Li, J. Hong, P. Hong, Combining data owner-side and cloud-side access control for encrypted cloud storage, IEEE Trans. Inf. Forens. Secur. 13 (8) (2018) 2062–2074.
[37] K. Xue, J. Hong, Y. Ma, D.S.L. Wei, P. Hong, N. Yu, Fog-aided verifiable privacy preserving access control for latency-sensitive data sharing in vehicular cloud computing, IEEE Netw. 32 (3) (2018) 7–13.
[38] X. Yang, F. Yin, X. Tang, A fine-grained and privacy-preserving query scheme for fog computing-enhanced location-based service, Sensors 17 (7) (2017) 1611.
[39] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata@MobiHoc 2015, Hangzhou, China, 2015, pp. 37–42. June 21, 2015.
[40] S. Yi, Z. Qin, Q. Li, Security and privacy issues of fog computing: A survey, in: Wireless Algorithms, Systems, and Applications - 10th International Conference, WASA 2015, Qufu, China, 2015, pp. 685–695. August 10-12, 2015, Proceedings.
[41] Z. Yu, M.H. Au, Q. Xu, R. Yang, J. Han, Towards leakage-resilient fine-grained access control in fog computing, Future Gener. Comp. Syst. 78 (2018) 763–777.
[42] C. Zhang, L. Zhu, C. Xu, X. Liu, K. Sharif, Reliable and privacy-preserving truth discovery for mobile crowdsensing systems, IEEE Trans. Depend. Secure Comput. (2019).
[43] C. Zhang, L. Zhu, C. Xu, K. Sharif, X. Du, M. Guizani, LPTD: Achieving lightweight and privacy-preserving truth discovery in ciot, Future Gener. Comp. Syst. 90 (2019) 175–184.
[44] Y. Zhang, Q. Chen, S. Zhong, Privacy-preserving data aggregation in mobile phone sensing, IEEE Trans. Inf. Forens. Secur. 11 (5) (2016) 980–992.