This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



# Multi-Access Filtering for Privacy-preserving Fog Computing

Keke Gai, *Member, IEEE,* Liehuang Zhu, *Member, IEEE,* Meikang Qiu, *Senior Member, IEEE,* Kai Xu, Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—The interest in fog computing is growing, including in traditionally conservative and sensitive areas such as military and governments. This is partly driven by the interconnectivity of our society, and advances in technologies such as Internet-of-Things (IoT). However, protecting against privacy leakage is one of several key considerations in fog computing deployment. Therefore, in this paper, we present a privacy-preserving multi-layer access filtering model, designed for a fog computing environment; hence, coined fog-based access filter (FAF). FAF comprises three key algorithms, namely: access filter initialization algorithm, optimal privacy-energy-time algorithm, and tuple reduction algorithm. Also, a hierarchical classification is used to distinguish the protection objectives. Findings from our experimental evaluation demonstrate that FAF allows one to achieve an optimal balance between privacy protection and computational costs.**

***Index Terms*—Privacy-preserving, access filter, network fusion, fog computing, dynamic programming, optimal scheduling**

## I. Introduction

In our contemporary society, mobile devices / apps are widely used to deliver a broad range of services, partly due to the underpinning technologies (e.g., embedded sensors) [1]–[4]. There are, however, known risks such as privacy leakage. For instance, a location-oriented mobile app has access to a user's geographical data and such information can be abused to stalk a user [5], [6]. Seeking to achieve optimal quality of user experience, most current access control mechanisms utilize a coarse-grained setting, which is configured by the underlying operating system. However, one of the many factors contributing to data over-collection (e.g., due to lack of or weak enforcement) is coarse-grained access control mechanism [7], [8]. Hence, there have been interest in deploying fine-grained access control mechanism [9], [10]. There are, however, limitations associated with fine-grained access control deployment in practice [11], [12]. For example, such an approach shifts the access control decision to users and most users, particularly elderly and non-technically inclined users, are not in a position to determine which applications or services should have access to what permission(s), etc. In addition, frequent permission requests also dilute or weaken the user's quality of experience.

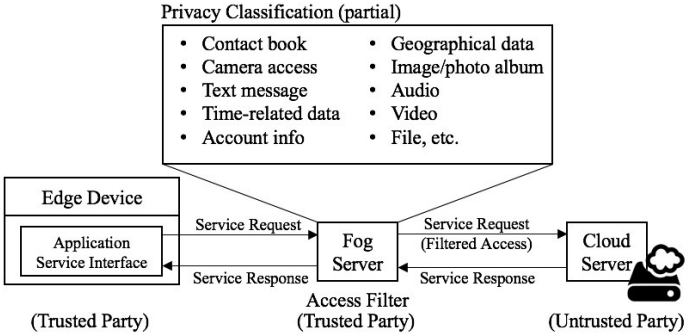


Fig. 1: Architecture of the proposed FAF model.

Seeking to strike a balance between offering customized access control capability and having a high quality of experience is non trivial. A user, for example, cannot entirely outsource the fine-grained access control capability to a potentially untrusted cloud service provider. Thus, an alternative approach is to rely on a trusted third-party. Therefore, in this paper we consider fog nodes as some trusted third-party, and such fog nodes can be a computationally capable device that is closer to the user. Examples of fog nodes include an intelligent home assistant device such as Amazon Echo and Google Home, or even a smart vehicle such as Tesla. In other words, these fog nodes will be utilized to facilitate the customization of user's access control mechanism for user-selected apps.

Specifically, extending an earlier prior fog-based paradigm [13], we present a *fog-based access filter* (FAF) model. We utilize fog computing for both computation and storage. Key differences between the FAF model in this paper and the earlier work are summarized as follows:

1) The FAF model in this paper comprises two new key algorithms, namely: access filter initialization algorithm and tuple reduction algorithm, and extends the optimal privacy-energy-time algorithm from the earlier work.
2) Unlike the earlier work, the FAF model in this paper offers a multi-layer customized access control capability.
3) To enhance privacy classification, we use weighting access in the FAF model in this paper.

K. Gai (first author) is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, 100081 (email: gaikeke@bit.edu.cn).

L. Zhu (Corresponding Author) is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, 100081 (email: liehuangz@bit.edu.cn).

M. Qiu (Co-corresponding Author) is with the Department of Electrical Engineering, Columbia University, New York, NY, USA, 10027, qiumeikang@yahoo.com.

K. Xu is with School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, 100081, 3220180758@bit.edu.cn.

K.K.R. Choo (email: raymond.choo@fulbrightmail.org) is with the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA.

This work is supported by Beijing Institute of Technology Research Fund Program for Young Scholars (Dr. Keke Gai).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



As shown in Fig. 1 , participants are categorized into *Trusted Party* (TP) and *Untrusted Party* (UP) and we consider fog servers and edge devices as TPs and the cloud service providers as UP. A TP fog server offers the function of filtering unnecessary access authorizations; hence, the fog contributor acts in a deterministic role. The main advantage of deploying this setting is to minimize wastage of computing resources on the edge, as well as establishing a programmable access control environment. The optimizing filter running on the fog server also provides a dynamic working mode, so that data over-collection can be minimized.

The key features / contributions of this paper are summarized as follows:

1) The proposed FAF model allows one to utilizes fog servers to filter access control authorizations, in order to provide fine-grained access service for applications (apps) and services. Since the fog server layer is considered a TP, cloud service providers do not to share access demands with the fog servers.
2) In our approach, access control is "translated" into a scheduling problem, in the sense that access authorization is associated with privacy weights and estimated computing costs. Hence, we are able to construct an essential deterministic access strategy, based on the available computing resources and service requirements.
3) In other words, we address an NP-complete problem that considers multi-dimensional constraints, which include privacy weight, latency, and energy cost. The core algorithm supporting the proposed approach is dynamic programming, which is designed to maximize the sum of the privacy weight while achieving the requirement / satisfying the required condition relating to the service.

The remainder of this paper is organized as follows. The next Section II discusses related literature. In Sections III and IV, we describe the proposed model and the three key algorithms. In Section V, we describe a potential use case. We then describe our experimental setup and findings in Section VI, before concluding the paper in Section VII.

## II. Related Work

As the proverbial saying goes "Security is as strong as the weakest link", and in our approach the fog nodes could be one of the weakest link, since an attacker could seek to directly compromise the fog node, say an Amazon Echo or a smart TV in the user's home [14]. For example, Su et al. [15] studied the potential of edge nodes been compromised and designed a secure content caching scheme. Specifically, in their work the authors developed a disaster backup mechanism using trusted nodes, as a caching-based solution requires storage space. They used an auction-style to allocate servers in order to optimize resource distributions on both edge and cloud nodes. Such an approach, however, requires significant caching storage and results in resource wastage due to replication.

### A. Fog-enabled Privacy-preserving Explorations

Fog-enabled privacy-preserving approaches can be broadly discussed in five broad categories, as follows.

First, there are approaches based on differential privacy. The latter is a typical privacy-preserving method that generally adds noises which obey a certain distribution to screen / mask the original data. Noise construction methods vary between use cases and scenarios. Yang et al. [16], for example, argued that it is possible to apply differential privacy methods to construct multifunctional data aggregation in fog computing. Such an approach use fog computing computing resources to support a broad range of statistical aggregation functions, where the fog server is tasked with deployment of training learning models to generate query sets. Piao et al. [17] also applied differential privacy techniques in data publishing, where fog servers are used to introduce differential noises. We observed that a common approach of implementing differential privacy in fog computing is to utilize fog computing computing resources (e.g., fog nodes' data processing capability). However, fog nodes can also be utilized to perform other activities. For example, Lyu et al. [18] focused on enhancing energy aggregation performance in a fog-based smart grid scenario, while seeking to achieve privacy protection. In this approach, Gaussian noise is introduced to achieve differential privacy, and the fog nodes encrypt the aggregated data.

Second, to achieve differential privacy one could also introduce pseudonyms to mask sensitive information, such as in the approach of Kang et al. [19]. However, a key limitation in the approach of Kang et al. [19] (and also other similar approaches) is that the performance of hiding features will likely degrade as the number of participants (e.g., vehicles in the context of Kang et al.'s approach [19]) decreases significantly. While introducing pseudonyms to mask sensitive information is easy to deploy, its utility can be limited as illustrated in the preceding example (e.g., vehicular ad-hoc network - VANET) or in other situations. Wang et al. [20] used a similar method to achieve information screening on a cognitive network, which added noises on the demand estimation queries in order to hide the demand differential privacy of sampled secondary users.

A third way of achieving differential privacy is to use cryptographic schemes, such as attribute-based schemes [21], [22]. Cryptographic schemes used in such an approach have to be sufficiently lightweight and robust, in order to deal with the dynamic nature of fog and other nodes and devices in the heterogeneous fog computing environment. Li et al. [23] proposed a privacy-preserving mining on vertically partitioned database that was based on association rule mining. The scenario assumed that data owners needed to learn association rules/ frequent itemsets without leaking raw data as much as possible. The proposed solution was a specific-purpose homomorphic encryption scheme. Such approach also encountered the efficiency issue and adoptability was challenged when the complicated task was processed at fog.

A fourth way is to use fog computing to provide differential privacy, for example during data gathering and data filtering [24]–[26]. For instance, Wei et al. [27] explained that fog computing can be utilized to provide an additional "firewall" for vehicles in a VANET setting. Yang et al. [28] attempted to solve the location-based privacy leakage problem by using fog nodes. The solution constructed a secure positioning protocol to achieve bounded retrievals. Du et al. [29], [30] investigated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



the method that uses a differential privacy-based query model for supporting data center by establishing a sustainable fog computing. This study used a query model to obtain the structure information in order to map the data center infrastructure. Both curious verifiers and passive outside attackers could be against as they were not within the bounded region. Computation overhead was also considered in this study. Nonetheless, there were two disadvantages in this protocol-based solution. The first disadvantage was that this solution was not applicable for multi-dimensional cases. The other was lacking proper method of location verification services.

The last successful attempt is to combine blockchain techniques with fog computing [31]. Their approach was designed to achieve privacy-preserving carpooling in the context of vehicular fog computing. The mechanism perfectly utilized the advantage of blockchain technique in privacy protections as well as the merit of fog computing in reducing communication overhead. Nevertheless, blockchain technique's virtue also brought its potential drawback, which might result in having adversarial nodes in the system. For example, a classic 51% attack could be easily launched when the number of vehicles was at a low level in the network.

### B. Fog-enabled Access Control Mechanisms

Access control mechanisms allow and restrict users from accessing the system, which is also especially important to ensure security [32]. The access control ensures the normal access of effective users, prevents unauthorized users from attacking, and solves the security problems caused by effective user fault operations. In fog computing, access control is an important tool to maintain users' privacy and ensure systems' security, which is a challenging task for meeting various systems' demands. In traditional access control models, users store data in trusted servers. The trusted servers then check whether the requesting user has access to the data. In fog computing, the model does not apply because the user and server are in different trust domains and the server is untrusted.

In order to extend the existing fog computing platform and support the secure sharing and communication of fog computing, authors in [33] proposed a strategy-based fog computing resource management method. However, this approach was only a preliminary framework and did not mention details such as how to identify users, make decisions, and protect identity and data privacy. To balance the security and functionality, many prior studies have made efforts. For example, Miao et al [34] developed a *Lightweight Fine-Grained ciphertexts Search* (LFGS) mechanism that combines *Cipher-text Policy Attribute-based Encryption* (CP-ABE) with *Searchable Encryption* (SE) in the context of fog computing. Fog nodes were mainly playing a role that participated in computation and data storage. Similarly, considering the support for outsourcing capabilities and attribute updates, a CP-ABE [35] was used to ensure data confidentiality and fine-grained data access control. It enabled data owners to define flexible access policies for data sharing. Hone et al. [36] further explored the manner of the technical combination by merging attribute-based access control with k-times anonymous authentication methods in the fog context. Similar to other fog-based solutions, this study also emphasized the latency time and cost-saving located at fog nodes.

In many attack scenarios, adversaries often targeted at those fixed attributes through mining, monitoring, or physical attacks [37], [38]; thus, applying dynamic attributes for access control could enhance the security protection capability. Li et al. [39] used dynamic attributes collected by smart devices (e.g. unlock failure, application usage, device location and proximity) to determine the risk level of the end user, and proposed a way to incorporate dynamic attributes into the ABE solution. Control algorithms to verify access rights in real time. Another study [40] proposed an efficient encryption scheme for access control and data sharing, enabling smart devices to securely share data with other devices at the edge of the cloud-assisted IoT. All of the security-oriented operations were offloaded to nearby edge servers that are within one hop of IoT devices.

Addressing the network context, Su et al. [41] investigated whether fog computing could be used for achieving end-to-end trustworthiness establishment. The finding of this work suggested that fog nodes could enhance security capability between organizations through offering security protocols. Zaghdoudi et al. [42] used a distributed hash table to build a scalable, generic and robust access control solution for ad hoc *Mobile Cloud Computing* (MCC) and fog computing, and proposed a DHT-based ad hoc MCC and fog computing access control method. The access control model applied to the creation of a mobile network in the case of the creation of a spontaneous network and responded to MCC access control requirements.

In summary, literature review given in this section indicated that the research on access control model had many unsolved issues. Our study was a further exploration based on the proofed functionality of fog computing retrieved from prior research. Based on our observation of the existing literature, most previous investigations and findings utilized fog servers as a medium who could play distinct roles, due to its data processing and location virtues. However, the medium role of fog servers generally could only offer an individual service or a single encryption method. Our work was distinct from most prior studies, as we aimed to design a scheduling mechanism for support multiple access control methods that was deployed at fog nodes. Fog computing can be utilized to minimize computing resources when a proper scheduling strategy is implemented. This partly motivates our multi-access design using optimal scheduling.

## III. Proposed FAF Model

### A. Definition

We will now describe the underlying components in our proposed FAF model, such as the fog-enabled access filtering mechanism in Fig. 2. Access filter (AF) is a key component in the model, which processes the deterministic action of filtering access authorization. AF consists of a number of core sub-components, namely: configured privacy data (CPD), essential access request (EAR), and optimization filter (OpF).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



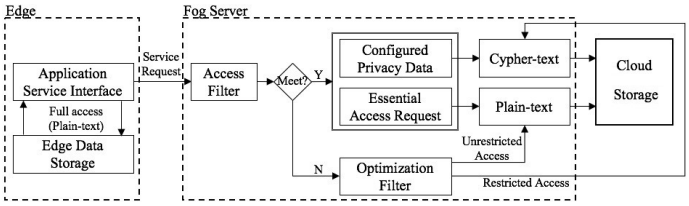


Fig. 2: Proposed fog-based access filter mechanism.

- CPD refers to data carrying sensitive information or data configured by the data owner/user. In our model, we consider such data class to be a primary group with a protection priority.
- EAR is the data access authorization, which meets the requirement of some services. Cloud service providers generally configure EARs.
- OpF, core to privacy-preserving optimization, runs as a control panel in our model. It essentially scopes down the access authorization in order to strengthen privacy protection when constraints are applied.

The principle for filtering privacy follows a content-oriented mechanism with a *Quality of Experience* (QoE) approach. The proposed approach uses our a content-oriented privacy-preserving method deriving from our prior work [43]. Moreover, our prior work [44] has explored QoE method through implementing the *Reinforcement Learning* (RL) technique. The quality here mainly refers to the satisfaction level of preserving privacy.

Application service interface normally has full or high level access to data, as a typical data access policy configured by the operating system is coarse-grained. In our model, data is transmitted to a trusted party, the fog server. This removes the need to send data directly to the untrusted cloud server. The role of the fog server is to make decisions on authorizing accesses. Any multiple encryption methods with distinct complexities can be applied on the plaintext data sent to the cloud, as long as they satisfy the conditions and requirements in the specific use case / application. In essence, the model is handled to work in a heterogeneous computing environment. Different computing environments and different encryption methods will generate varying executing time length and energy costs.

A more efficient access filtering will help us achieve higher-level of data security. In this paper, we do not focus on individual data security techniques, as we are more interested in the different security levels offered by the different encryption algorithms. It is also generally accepted that loss / leakage of encrypted data is potentially less damaging than losing / leaking unencrypted data, for obvious reason. Hence, the scope of data access can also be restricted by encrypting partial data. In the context of our model, it is the fog servers that help determine how access to data usage requests is made.

In our problem formulation, we consider privacy, time, and energy demand (e.g., electricity).

1) A customized access control with multiple encryption methods is applied to increase data protection ability, based on the available computing resources. Therefore, access customization can be determined in the multi-layer system.
2) Time cost is also another important parameter for measuring efficiency. A timing constraint is used to assess whether all tasks can be completed within a required time frame. This is an adjustable and controllable parameter in our model, which depends on practical demands.
3) Minimizing energy consumption helps achieve green computing and reduces operating costs.

Data with sensitive information are ranked by their data classes at the fog servers. This operation is generally completed by the system administrators, which can be varied due to different requirements. All access filtering work modes process encryption over data sequentially, but various work modes can have different proportion of encrypted data (as not all data need to be encrypted). Our model assumes that the ability of preserving privacy is associated with the amount of encrypted data which is a basic parameter to measure whether the ability of preserving privacy is enhanced.

Hence, our model seeks to address the *Maximizing Privacy-Level and Minimizing Energy-Cost in Heterogeneous Modes* (MPLMECHM) problem – see also Definition 3.1.

*Definition 3.1:* **Maximizing Privacy-Level and Minimizing Energy-cost in Heterogeneous Modes Problem:**
Input: data packages ($D^j$), which comprise a group of input data sorted by data classes with given privacy proportion ($Pr_i^j$) of the encrypted data in each data package and its relevant executing time ($T_i^j$) and energy consumption $E_i^j$. The working mode determines the privacy proportion of encrypted data. The time constraint ($T_c$) limits the total completion time of all operations within the period.
Output: a cryptographic strategy determining which data package shall be encrypted. a strategy for determining the privacy proportion of the encrypted data in each data package.
The main objective is to obtain the minimum value of the energy cost and the maximum value of the total privacy proportion under the given time constraint.

Let us consider the below scenario, where the input to our model is $D^j$ data package. The $j$th data package containing $i$ encrypted data is denoted by $Pr_i^j$; $T_i^j$ denotes the required executing time while $Pr_i^j$ is determined; $E_i^j$ denotes the required energy cost under the same condition. We define a 4-tuple, $\langle D, T, E, Pr \rangle$, to indicate a mode of work that includes data package and its relevant executing time, energy consumption, and privacy proportion. A set of 4-tuples is produced as a strategy for output generation.

We will give the objective function in the next section.

### B. Model Design

Similar to many prior studies, our model assumes fog nodes to be a suitable security medium to determine the authorization of the remote cloud. The choice of fog servers in such a context is because most current apps can only deploy coarse grained accessing configured by the operating system. There is limited ability to customize access due to lack of control at data owner/user. In addition, we can disable access using data encryption. In other words, the cloud server's access will

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



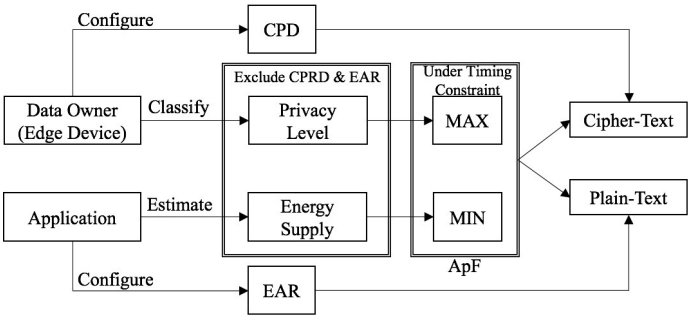


Fig. 3: The diagram of the proposed model's working flow.

be restricted to only unencrypted data, even though the cloud service provider may have "full" access.

The proposed model mainly consists of three phases. We consider edge device to be the data owner, where data are collected. Two operations are completed in this phase, including the privacy classification for producing privacy levels and the data configuration. We present a high-level diagram of the proposed model in Fig. 3. As illustrated in the figure, mobile apps at the edge layer primarily acts as the data owner, and provides privacy proportion evaluations and energy consumption estimates. We define a 2-tuple, $\langle D, Pl \rangle$, to indicate an input data package which shows the data class and its relevant privacy level. The configuration that establishes the connection between data classes and privacy levels is determined by fog nodes. The privacy level associated with the input data package is determined by the value of the privacy level, which is an approximate value based on the significant of a particular data class to the data owner.

Meanwhile, the second phase takes place at the application side, where estimates the value of the element used for strategy-making. EAR is configured by the application side and determines which data will be stored in the plain-text manner.

In third phase, data packets at each data package will be arranged on the basis of their privacy levels at the fog layer. In essence, the output strategy determines the order of encrypting data according to the value of privacy level. The approximation of the energy cost needs to be stored in advance and also provided by fog layer. We estimate the cost of energy according to data class and data size. The mathematical format of estimating the cost of energy is given in Eq. (1):

$$E^{est} = \alpha^{Pl} S. \tag{1}$$

In the above equation, $E^{est}$ represents the approximation of the energy cost, $\alpha^{Pl}$ represents the parameter that reflect the data class, and $S$ represents the size of input data package.

In order to abstract the proposed problem, we give the objective function in Eq. (2):

$$\begin{cases} \mathbf{Pr}^o = MAX[\mathbf{Pr}] \\ \mathbf{E} = MIN[\mathbf{E}] \text{ while having } \mathbf{Pr}^o \\ \mathbf{T} \leqslant T_c. \end{cases} \tag{2}$$

In the equation, $\mathbf{Pr}^o$ refers to the first-rank encrypted data proportion, $\mathbf{E}^o$ refers to the optimal energy consumption connected with the first-rank encrypted data proportion, and $\mathbf{T}$ refers to the required time connected with the above two variables. Retrieving the optimal proportion requires selecting the maximum value of proportion and the minimum value of from all possible sets which meet the condition $\mathbf{T} \leqslant T_c$.

Eq. (3) gives the mathematical expression of the state transition equation in our model.

$$\begin{cases} \mathbf{S}_1 = \mathbf{P}_1 \\ \mathbf{S}_i = \mathbf{S}_{i-1} + \mathbf{P}_i \text{ while } i > 1. \end{cases} \tag{3}$$

The equation mainly describes the iterative process in the dynamic programming algorithm of the proposed model. $\mathbf{S}_i$ refers to the optimal strategy generated by current iteration, $\mathbf{S}_{i-1}$ refers to the optimal strategy generated by last iteration, and $P_i$ refers to the local optimal tuples that should be calculated in current iteration.

$$\mathbf{Pr}^o = \prod_{j=1}^{m} \prod_{\substack{f(i)=1, \\ i=1}}^{n(j)} Pr_i^j. \tag{4}$$

$$\mathbf{E}^o = \sum_{j=1}^{m} \sum_{\substack{f(i)=1, \\ i=1}}^{n(j)} E_i^j + \overline{E} \times T^{SG}. \tag{5}$$

$$\mathbf{T} = \sum_{j=1}^{m} \sum_{\substack{f(i)=1, \\ i=1}}^{n(j)} T_i^j + T^{SG}. \tag{6}$$

Eq.s (4) to (6) are the specific calculation methods of the iterative process in Eq. (3). Specifically, $m$ refers to the maximum number of data package; $j$ refers to the index of data package; the function $n(j)$ is created to indicate the maximum number of the data packet or data contained in a certain data package; and the function $f(i)$ is created to present the current working mode of encrypting the data/ data packet, regardless of which encryption method is chosen. The calculation of the energy cost considers energy cost from both the computation workload and the strategy generation. In Eq. (5), $\overline{E} \times T^{SG}$ represents the energy cost of the strategy generation. $\overline{E}$ is the energy consumption in unit that can be obtained from calculating the average value; $T^{SG}$ is a time length that is caused by the strategy generation. Similarly, the total time $\mathbf{T}$ also involves this time period. The major reason for considering $T^{SG}$ as well as the energy cost associated with $T^{SG}$ is because the core algorithm used in our approach is a dynamic programming. Time complexity generally higher than other optimization algorithms, such as greedy, memorization, and genetic algorithms. The additional time deriving from the strategy generation needs to be considered in order to guarantee a higher-level efficiency of the whole system.

According to the actual needs, our system provides a variety of different encryption methods, so that the value of function $f(i)$ should be a natural number. The quantity of the available working mode determines the upper bound for this function. Suppose that our system includes $r$ kinds of encryption methods; therefore, we can select the value of the function $f(i)$ from the range [0, r]. The size of the r value can be adjusted

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



according to the needs of the actual application scenario. The value of $r$ will affect the time complexity of the proposed approach, because each working mode adds one dimension during the implementation of dynamic programming. Since each additional encryption method adds one dimension during the implementation of dynamic programming, the variation of $r$ value will affect the time complexity of the proposed approach.

$$\begin{cases} \Delta\mathbf{Pr} = \mathbf{Pr}^o - \mathbf{Pr}^c \\ \Delta\mathbf{E} = \mathbf{E}^o - \mathbf{E}^c. \end{cases} \tag{7}$$

Compare to other methods, our approach can be measured by calculating the difference in the proportion of the encrypted data and the total energy cost. Eq. (7) represents methods of the performance measurement through comparing our approach with others. In equations, a $\Delta\mathbf{Pr}$ refers to the proportion difference; a $\mathbf{Pr}^c$ denotes the proportion obtained from the comparing method. Similarly, a $\Delta\mathbf{E}$ denotes the energy cost difference and a $\Delta\mathbf{E}^c$ means the energy cost generated by the comparing method.

In summary, our model design highlights the functionality of fog servers in scheduling data encryption operations. A time length is configured to be a condition constraint and a dynamic programming algorithm is proposed to optimize the implementation of the available computing resource and maximize the performance in both energy-saving and data protection.

### C. Threat Environment

In this section we analyze the major threats that may exist based on the access control system proposed above. The analysis is based on the fact that existing cloud services typically over-collect data, which caused by the coarse-grained access control mechanism configured in the operating system. We consider edge devices as data owners and fog servers as the core components that limit accesses in the system. Contrary to their being considered as the trusted parties, cloud servers and applications running on edge devices are considered as untrusted parties. The cloud servers are not trusted because of the phenomenon mentioned above that cloud services over-collect data and may abuse it when data owners can not control the data. Besides, applications running on edge devices also have a potential to threat privacy by reason that they are service presentations of the cloud servers' solution.

In our model, we define the act of abusing data as any behavior that is not within the expected scope of the data owner, and behavior that occurs without authorization from the data owner. Due to the coarse-grained access control mechanism, we can assume that the untrusted parties which including both cloud servers and applications can fully access the data. Therefore, we assume that the threat for untrusted parties in the proposed system is mainly focused on the abuse of data after data from edge devices have been collected. To create a fine-grained access control mechanism at the fog server, our proposed system can limit the access scope and permissions of both cloud servers and applications.

## IV. Algorithms

In this section we describe the main algorithms used in the proposed model. Main algorithms used in proposed model are represented in this section. Section IV-A presents a preprocessing algorithm, Section IV-B presents the main algorithm designed for the proposed model and Section IV-C presents a supplementary algorithm for the main algorithm.

### A. Access Filter Initialization Algorithm

This algorithm is designed to initialize the input of the optimization considering the timing constraint. The demand of this algorithm derives from various needs from both user and service provider sides. CPD shall be encrypted first as data owners configure them with a high priority of the data protection. It means that CPD do not have the un-encryption working mode. For those EAR data, service providers need a full access so that they only have an un-encryption working mode. Other data going through OpF will be divided into a few working modes that include both encryption and un-encryption alternatives.

As there is a latency tolerance, the amount of the input data packets need to be restricted in order to guarantee the total executing time shall be shorten than the configured timing constraint. Hence, we design an algorithm named *Access Filter Initialization* (AFI) algorithm to formulate the input of the access filter.

The input of this algorithm should be given by a "mapping table" (*M-Table*), which is a collection of raw data collected, recording all possible 4-tuples. The reason we use a set of 4-tuples as algorithm inputs is that each 4-tuple $\langle D, T, E, Pr\rangle$ represents a working mode in the mapping table. The output of this algorithm is a costing table containing only the optimal 4-tuples under timing constraints. During the execution of Algorithm 1, we conduct a sorting operation on the tuples of each data packet. Suppose that the number of data packets is $m$ and the number of tuples in each data packet is $n$, we first sort all input tuples by time consumption values, and then delete or retain each tuple according to the constraint. So we can figure out that the time complexity of Algorithm 1 to solve $mn$ input tuples problem is $O\left(mn^2\right)$. The Algorithm 1 creates a new table to store the output. We assume that the space complexity of each tuple is constant $\Omega\left(1\right)$, so the space required by Algorithm 1 in the proposed model is $\Omega\left(mn\right)$. We will provide a presentation about examples of the mapping table and costing table in Section V.

### B. Optimal Time-Constrained Privacy-Energy Algorithm

In this section, a core algorithm for implementing the key processes of the proposed model is presented. The proposed algorithm is called *Optimal Time-Constrained Privacy-Energy* (OTCPE) algorithm. Inputs to this algorithm include a "costing table" *C-Table* which only contains the optimal 4-tuples and is consistent with Definition 3.1. In fact, the creation of a costing table, which is described in Algorithm 1, is usually an intermediate process in the overall algorithm. The output of this algorithm is an optimization table (*O-Table*), which consist

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



**Algorithm 1 Access Filter Initialization (AFI) Algorithm**

**Require:** M-Table
**Ensure:** C-Table

```
Initialize a C-Table; input all tuples from M-Table to C-Table; call Algorithm 3
for ∀ data packages do
    Sort the tuple by ascending the value of T_k
    for ∀ time spot T_j in M-Table do
        Input all tuples with the same data package and time spot T_j to Algorithm 3
        return the optimal tuples from Algorithm 3
    end for
end for
return C-Table
```

**Algorithm 2 Optimal Time-Constrained Privacy-Energy (OTCPE) Algorithm**

**Require:** C-Table
**Ensure:** O-Table

```
Initialize an O-Table; input all tuples from C-Table to O-Table; call Algorithm 3
for ∀ data packages do
    for ∀ time units (T_k) within the timing constraint range do
        for ∀ time spot T_j in C-Table do
            if ∀ the preceding data/ data packet is not null (considering iteration) then
                Sum up energy cost and proportion
            end if
            Input all tuples with energy cost E and proportion Pr to Algorithm 3
            return the optimal tuples from Algorithm 3
        end for
    end for
end for
return O-Table
```

of a set of 4-tuples selected from the costing table. We conduct a simple addition and multiplication on each tuple during the execution of Algorithm 2, and then delete or retain each tuple according to the constraint. Suppose that the number of data packets is $m$ and the number of tuples in each data packet is $n$, we can figure out that the time complexity of Algorithm 2 to solve $mn$ input tuples problem is $O(mn)$. The Algorithm 2 creates a new table to store the output. We assume that the space complexity of each tuple is constant $\Omega(1)$, so the space required by Algorithm 2 in the proposed model is $\Omega(mn)$.

The Algorithm 2 gives the pseudo code of the OTCPE algorithm. The key steps of the algorithm are as follows:

1) Initialize an O-Table and input all tuples from C-Table to O-Table There are two components in the structure of the O-Table. First, it has a timeline that shows all possible time units (constraint). Second, under each timing constraint in the table, each data package has a cell which stores one or a number of 2-tuples $\langle E, Pr\rangle$. We input all data packages in sequence.

**Algorithm 3 Tuple Reduction (TR) Algorithm**

**Require:** A set of tuples $\mathcal{P} = \{\langle E_k, Pr_k\rangle\}$
**Ensure:** An updated set of $\mathcal{P}_{update}$

```
Input the set P, sort the tuple by ascending the value of E
for ∀ k do
    if Pr_k ≤ Pr_{k+1} then
        Delete the tuple ⟨E_k, Pr_k⟩; update P
    else if E_k == E_{k+1} then
        Delete the tuple ⟨E_{k+1}, Pr_{k+1}⟩; update P
    end if
end for
return P_update
```

2) While inputting a data package, check to see if it has a previous cell in iterations and the available working modes. Summarize tuples if applicable. A key operation is to delete unnecessary tuples. We utilize an supplementary Algorithm 3 in the next section to make sure that each cell only retains optimal tuples after deleting non-optimal tuples.
3) Update O-Table after each cell update is completed. When all data packages are inputted and all iterative operations are not applicable, the generation of an O-Table is completed. Return the final O-Table.

### *C. Tuple Reduction (TR) Algorithm*

In this section, in order to remove unnecessary tuples in the dynamic programming implementation process and reduce the spatial complexity of the algorithm, we introduce a core supplementary algorithm called Tuple Reduction (TR) algorithm. This algorithm is used to ensure that only local optimal solutions are retained in each iteration of the dynamic programming implementation process. Therefore, inputs to the algorithm include a set of tuples; outputs of the algorithm include a set of updated tuples containing only optimal tuples. Algorithm 3 shows the pseudo codes of TR algorithm.

We consider here the computational complexity notion of time and space, which is the number of computational steps needed for the algorithm to finish its task. We assume that the space complexity of each tuple is constant $\Omega(1)$. Assuming that the number of input tuples in Algorithm 3 is $n$, we first sort all input tuples by energy consumption values, and then delete or retain each tuple according to the constraint. The Algorithm 3 does not occupy other storage space, but only operates on the basis of the original tuples' space. So we can figure out that the time complexity of Algorithm 3 to solve $n$ input tuples problem is $O(n^2)$ and the space required by Algorithm 3 in the proposed model is $\Omega(1)$.

The Algorithm 3 gives the pseudo code of the TR algorithm. The key steps of the algorithm are as follows:

1) We input all tuples in sequence. Then sort the tuple by ascending the value of $E$ to make sure all tuples are arranged in order of energy consumption from large to small. The structure of the tuple consists of energy cost and proportion $\langle E, Pr\rangle$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



2) The key operation is to delete unnecessary tuples. While inputting a tuple, check to see if the proportion $Pr_k$ is less than or equal to next proportion $Pr_{k+1}$. If it is, delete the tuple $\langle E_k, Pr_k \rangle$ and update $\mathcal{P}$; if the proportion $Pr_k$ is more than next proportion $Pr_{k+1}$ and the energy cost $E_k$ is equal to next energy cost $E_{k+1}$, delete the tuple $\langle E_k, Pr_k \rangle$ and update $\mathcal{P}$.
3) Update the set $\mathcal{P}$ after all tuples update is completed. When all tuples are inputted and all iterative operations are not applicable, the updating of the set $\mathcal{P}$ is completed. Return the final set $\mathcal{P}$.

Our previous research has explored non-optimal tuples deletion rules [45], [46]. Therefore, each cell only retains optimal tuples after deleting non-optimal tuples.

In the next section, we will introduce the main procedures of the proposed method.

## V. Potential Use Case

In this section, we present a motivational example of the proposed method. In the scenario of the given example, we assume that the application collects four data packages. The cloud server that provides all back-end services has full access to the data. Due to the service agreement, the application also has full access. Our approach is based on the fact that cloud service providers and applications are not allowed direct access to some data by data owners, especially some data with privacy. There is an urgent and observable need to create a mechanism that limits the scope of data access except the operating system.

In the next, we present the configuration for the given example. Suppose that there are four data packages, i.e., $D_1$, $D_2$, $D_3$, and $D_4$. In our example, three parameters are considered, including length of time ($T$), energy consumption ($E$), and the proportion of data encryption ($Pr$). We express the variables $T$ and $E$ in units, and $Pr$ in decimals to indicate the proportion. Four distinct working modes ($M_1$, $M_2$, $M_3$, and $M_4$) have been established, and the values of the parameters are different. In the four working modes, $M_1$ does not encrypt any data; $M_2$, $M_3$, and $M_4$ are 3 distinct encryption methods. The M-Table we mentioned in the example is presented in Table I. It illustrates that there is no change in the proportion of unencrypted data. But in other three working modes, each data package has three different sets of data, which are mainly determined by different proportion values. Outputting all optimal solutions under the constraint of time ranges [11, 30] is the goal of the motivational example.

Next, as a major part of data preprocessing, we use the costing table, which is created using the mapping table and is machine-readable, as an input to our main algorithm. An example of C-Table is shown in Table II. As can be seen from the table, each cell contains a different number of 2-tuples. The row of each 2-tuple represents the data package it belongs to, and the column represents the time condition it is in. Under a certain time condition, the stored binary group is the optimal working mode. Under a certain time condition, the stored 2-tuples all are optimal tuples. The OTCPE algorithm we proposed will be applied on C-Table.

In the final optimization table, our algorithm will only keep optimal tuples. Table III illustrates the output (O-Table). Under the given time constraints, tuples in different data packages that meet the time limit are selected in order. Then Summarize tuples. The entire dynamic programming process adds data packages in strict chronological order and implements the iteration repeatedly. When the timing constraint in the cell increases, additional tuples may be created in each step.

As shown in Table III, we can observe that all optimal tuples within the specified time constraint ranges of 11 to 30 are listed. For instance, two optimal tuples, which are $\langle 11, 0.0081 \rangle$ and $\langle 6, 0 \rangle$, output under time constraint 14. Tuple $\langle 11, 0.0081 \rangle$ indicates the total energy consumption is 11, with the accumulative total of the proportion is 0.0081. The number of output tuples may vary under different time constraints. In most cases, the number of optimal tuples increases as time constraints are relaxed. Depending on the priority criteria set by the data owner or system developer, the proposed approach can choose one of the multiple optimal tuples as the output. The next section will show some of the experimental results gathered from our evaluation.

## VI. Experiment and the Results

Our experiments mainly evaluated following aspects which we paid close attention to: energy costs, proportion of encryptions, and strategy generation time. The first element was a vital parameter for assessing performance of green computing; the proportion of the encrypted data was a benchmark for evaluating capability of preserving privacy; the strategy generation time was testing whether the proposed approach was practical from the perspective of efficiency.

Since the proposed model considered the cloud server as the untrusted party and used the fog server to filter access control authorization, we used our own simulator as a fog server to complete the evaluation, which was developed by Java. Hardware configuration was with a processor which was an Intel Core i5-7200U CPU and a 4.0 GB memory.

The encryption methods used in the proposed model all were symmetric encryption algorithm. In order to illustrate that the time of using different symmetric encryption algorithms for the same plaintext was basically the same, which is easy to understand because time-intensive algorithms have been eliminated, we have done a simple comparative experiment on some mainstream symmetric encryption algorithms which included AES, DES and 3DES. We applied three encryption algorithms mentioned above to the same plaintext respectively. We found AES cost 41.65 milliseconds, DES cost 34.33 milliseconds and 3DES cost 38.50 milliseconds. The result presented the three classic encryption algorithms took approximately the same time for the same plaintext encryption. Therefore, we could believe that the growth of the time consumption had nothing to do with the category of encryption methods and was only related to the number of encryption methods.

In order to judge whether our approach was superior to prior work addressing similar method, our evaluation compared our FAF with another optimization algorithm (DASS) [47]. The DASS algorithm was a greedy algorithm that selected two

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



TABLE I: Example of Mapping Table (M-Table)

| Data Package | $M_1$ | | | $M_2$ | | | $M_3$ | | | $M_4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | T | E | Pr | T | E | Pr | T | E | Pr | T | E | Pr |
| $D_1$ | 1 | 1 | 0 | 3 | 4 | 0.3 | 4 | 3 | 0.3 | 4 | 4 | 0.3 |
| | | | | 6 | 8 | 0.6 | 8 | 6 | 0.6 | 7 | 8 | 0.6 |
| | | | | 9 | 12 | 1 | 12 | 10 | 1 | 10 | 11 | 1 |
| $D_2$ | 1 | 2 | 0 | 3 | 5 | 0.3 | 5 | 2 | 0.3 | 4 | 5 | 0.3 |
| | | | | 7 | 8 | 0.6 | 8 | 5 | 0.6 | 7 | 9 | 0.6 |
| | | | | 10 | 11 | 1 | 11 | 9 | 1 | 12 | 10 | 1 |
| $D_3$ | 2 | 1 | 0 | 4 | 2 | 0.3 | 3 | 3 | 0.3 | 5 | 4 | 0.3 |
| | | | | 8 | 6 | 0.6 | 6 | 6 | 0.6 | 7 | 7 | 0.6 |
| | | | | 12 | 9 | 1 | 9 | 10 | 1 | 11 | 10 | 1 |
| $D_4$ | 2 | 2 | 0 | 4 | 3 | 0.3 | 5 | 4 | 0.3 | 3 | 2 | 0.3 |
| | | | | 8 | 6 | 0.6 | 7 | 6 | 0.6 | 6 | 5 | 0.6 |
| | | | | 12 | 10 | 1 | 11 | 9 | 1 | 9 | 8 | 1 |

TABLE II: Example of Costing Table (C-Table)

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | (1,0) | (1,0) | (4,0.3)<br>(1,0) | (3,0.3)<br>(1,0) | (3,0.3)<br>(1,0) | (8,0.6)<br>(3,0.3)<br>(1,0) | (8,0.6)<br>(3,0.3)<br>(1,0) | (6,0.6)<br>(3,0.3)<br>(1,0) | ... |
| $D_2$ | (2,0) | (2,0) | (5,0.3)<br>(2,0) | (5,0.3)<br>(2,0) | (2,0.3) | (2,0.3) | (8,0.6)<br>(2,0.3) | (5,0.6)<br>(2,0.3) | ... |
| $D_3$ | - | (1,0) | (3,0.3)<br>(1,0) | (2,0.3)<br>(1,0) | (2,0.3)<br>(1,0) | (6,0.6)<br>(2,0.3)<br>(1,0) | (6,0.6)<br>(2,0.3)<br>(1,0) | (6,0.6)<br>(2,0.3)<br>(1,0) | ... |
| $D_4$ | - | (2,0) | (2,0.3) | (2,0.3) | (2,0.3) | (5,0.6)<br>(2,0.3) | (5,0.6)<br>(2,0.3) | (5,0.6)<br>(2,0.3) | ... |

TABLE III: Example of Optimization Table (O-Table)

| **Time** | **Optimal Solutions** |
|---|---|
| 11 | (6, 0) |
| 12 | (14, 0.0081) (6, 0) |
| 13 | (13, 0.0081) (6, 0) |
| 14 | (11, 0.0081) (6, 0) |
| 15 | (17, 0.0162) (10, 0.0081) (6, 0) |
| 16 | (16, 0.0162) (9, 0.0081) (6, 0) |
| 17 | (14, 0.0162) (9, 0.0081) (6, 0) |
| 18 | (20, 0.0324) (13, 0.0162) (9, 0.0081) (6, 0) |
| 19 | (19, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 20 | (17, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 21 | (24, 0.0648) (23, 0.054) (16, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 22 | (23, 0.0648) (22, 0.054) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 23 | (20, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 24 | (27, 0.108) (19, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 25 | (27, 0.1296) (26, 0.108) (19, 0.0648) (18, 0.054) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 26 | (24, 0.1296) (23, 0.108) (18, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 27 | (31, 0.18) (24, 0.1296) (22, 0.108) (18, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 28 | (30, 0.216) (29, 0.18) (22, 0.1296) (18, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 29 | (27, 0.216) (22, 0.1296) (21, 0.108) (18, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |
| 30 | (35, 0.3) (27, 0.216) (26, 0.18) (22, 0.1296) (21, 0.108) (18, 0.0648) (15, 0.0324) (12, 0.0162) (9, 0.0081) (6, 0) |

priorities so that two-round scheduling was implemented. This algorithm was a representative approach since this thinking was common in practice. The advantage of this type of algorithm was that the strategy generation time was effectively short, even though the output was not optimal.

Data collections came from a variety of experimental settings. Due to page limitation, we only presented a representative set of evaluation results for exhibiting the difference between the two methods in terms of both energy cost and privacy protection. Fig. 4 and Fig. 5 presented partial results from one of our data collections. The number of input data packages were 10 in this round of evaluation. We set the value

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



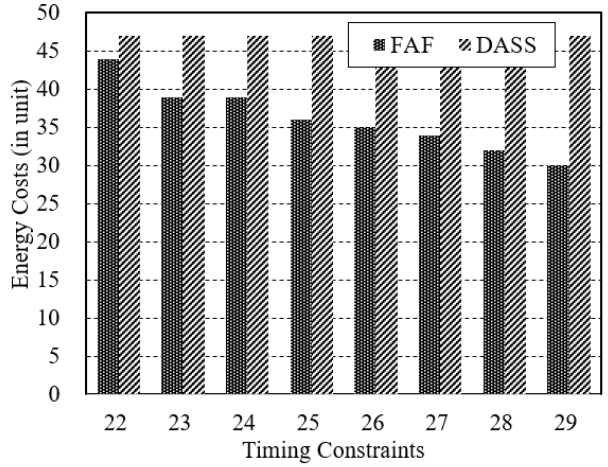


Fig. 4: Some comparison results between FAF and DASS concerning energy costs.

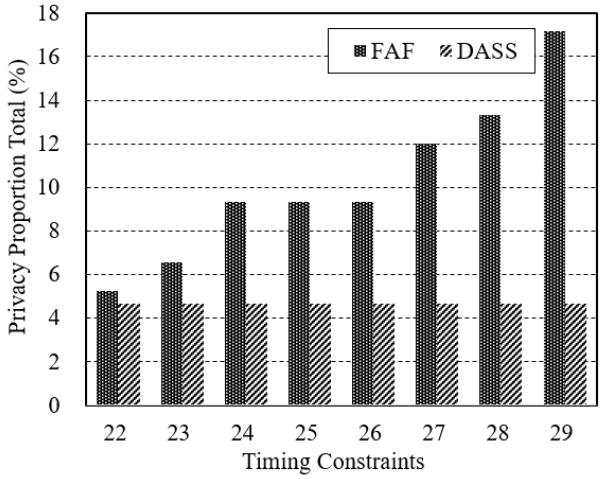


Fig. 5: Some comparison results between FAF and DASS concerning privacy proportion totals.

TABLE IV: Input Configuration

| | | # Data Package | # Encryption Method |
|---|---|---|---|
| Setting 1 | Setting 1-1 | 5 | 3 |
| | Setting 1-2 | 10 | |
| | Setting 1-3 | 15 | |
| | Setting 1-4 | 20 | |
| | Setting 1-5 | 25 | |
| Setting 2 | Setting 2-1 | 5 | 5 |
| | Setting 2-2 | 10 | |
| | Setting 2-3 | 15 | |
| | Setting 2-4 | 20 | |
| | Setting 2-5 | 25 | |
| Setting 3 | Setting 3-1 | 10 | 3 |
| | Setting 3-2 | | 4 |
| | Setting 3-3 | | 5 |
| | Setting 3-4 | | 6 |
| | Setting 3-5 | | 7 |
| Setting 4 | Setting 4-1 | 15 | 3 |
| | Setting 4-2 | | 4 |
| | Setting 4-3 | | 5 |
| | Setting 4-4 | | 6 |
| | Setting 4-5 | | 7 |

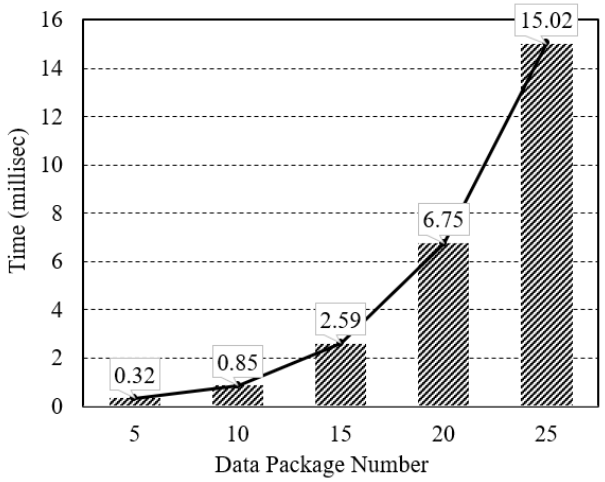


Fig. 6: Generation time collections under Setting 1.

of energy cost within a range [2, 8] and set the value of the protection proportion within a range [0.2, 1]. Fig. 4 depicted the difference between FAF and DASS for comparing energy costs under different time constraints. In the meanwhile, Fig. 5 depicted the difference between FAF and DASS for comparing the proportion of privacy protection collected from the same evaluation round compared to Fig. 4. According to results illustrated in this setting, the average privacy proportion was increased 121.3% and the average energy cost was reduced 23.1%.

In order to figure out the influence degree of both data package number and encryption method number on the length of strategy generation time, we adopted the controlling variable method to set up a total of four different experimental settings. For instance, in Table IV which presented the input configuration of different experimental settings, both Setting 1 and Setting 2 kept the number of encryption methods unchanged when the number of data packages varied from 5 to 25. On the contrary, Setting 3 and Setting 4 kept the number of data packages unchanged when the number of encryption methods varied from 3 to 7. The intention of having these setting was examining FAF's adoptability under different scenarios. We also evaluated the impact caused by the number of the data package and the number of encryption methods; therefore, two variables (# Data Package and # Encryption Method) were given in the table.

Fig. 6 and Fig. 7 presented a group of collections showing strategy generation time lengths with different data package number and same encryption method number under Setting 1 and Setting 2. Meanwhile, Fig. 8 and Fig. 9 exhibited a set of collections showing the length of strategy generation time with different encryption method number and same data package number under Setting 3 and Setting 4. The time length was counted in milliseconds, as shown in figures. From Fig. 6 and Fig. 7, we observed that the time consumption dramatically went up when the number of data packages increased and the number of encryption methods did not change, and the growth

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



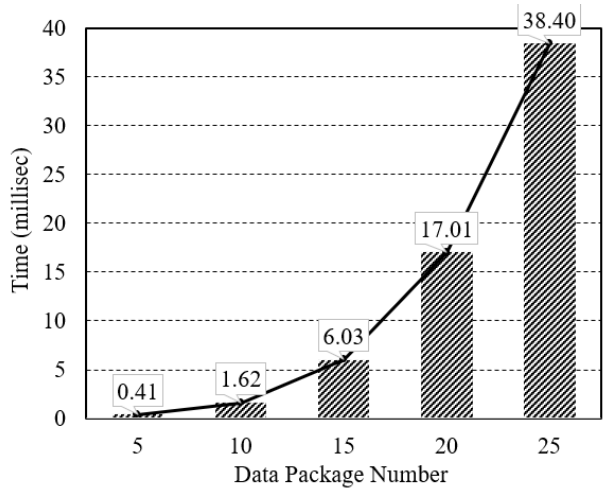


Fig. 7: Generation time collections under Setting 2.

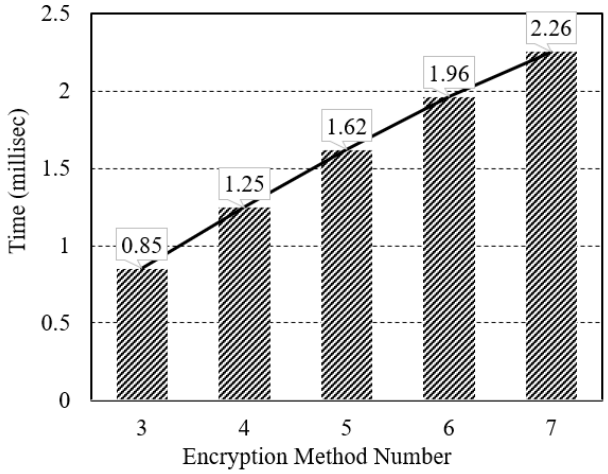


Fig. 8: Generation time collections under Setting 3.

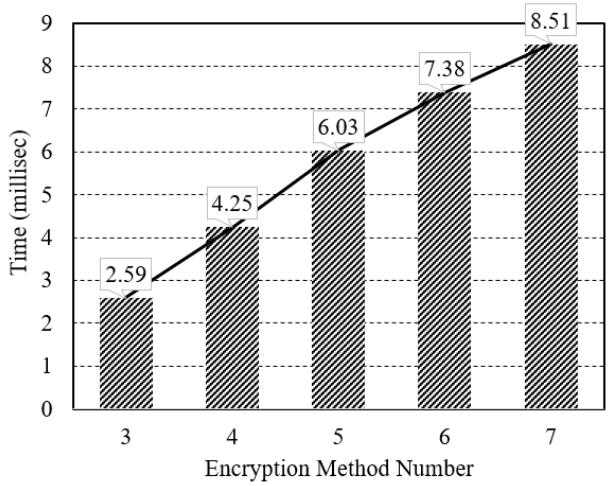


Fig. 9: Generation time collections under Setting 4.

trend was closer to exponential growth than linear growth. Fig. 8 and Fig. 9 also depicted that the increment of the number of encryption methods could result in a growth of the time consumption, but the influence extent of encryption method number was less than that of data package number.

TABLE V: Measurement Parameters Under Setting 1

| Setting 1-$i$ | $i$=1 | $i$=2 | $i$=3 | $i$=4 | $i$=5 |
|---|---|---|---|---|---|
| Ave | 0.318 | 0.851 | 2.593 | 6.748 | 15.020 |
| Min | 0.140 | 0.383 | 0.946 | 1.661 | 2.809 |
| Max | 2.265 | 5.993 | 14.023 | 22.706 | 35.505 |
| $DSM_i$ | 15.148 | 14.633 | 13.831 | 12.669 | 11.640 |
| $TGR_{i,i-1}$ | - | 2.677 | 3.048 | 2.602 | 2.226 |

TABLE VI: Measurement Parameters Under Setting 2

| Setting 2-$i$ | $i$=1 | $i$=2 | $i$=3 | $i$=4 | $i$=5 |
|---|---|---|---|---|---|
| Ave | 0.412 | 1.621 | 6.029 | 17.009 | 38.404 |
| Min | 0.263 | 0.623 | 1.734 | 4.124 | 8.977 |
| Max | 3.972 | 8.809 | 23.700 | 51.171 | 107.209 |
| $DSM_i$ | 14.089 | 13.140 | 12.669 | 11.407 | 10.943 |
| $TGR_{i,i-1}$ | - | 3.934 | 3.721 | 2.821 | 2.259 |

In order to have a better evaluation, we introduced a few parameters, as shown in Table V to Table VIII, that included average, minimum, maximum, distribution scope measurement, and time grow values. Besides common values, we proposed a *Distribution Scope Measurement* (DSM) value to assess the distribution extent. The retrieval method of DSM was shown in Eq. (8).

$$DSM_i = (Max_i - Min_i)/Mins_i. \tag{8}$$

In the equation, we assumed $DSM_i$ to be a DSM under a random setting $i$; $Max_i$ was the maximum value under the corresponding setting and $Min_i$ was the corresponding minimum value. DSM's value determined the extent of the distribution.

$$TGR_{i,j} = Ave_i/Ave_j. \tag{9}$$

Another parameter was a *Time Growth Rate* (TGR) that measured the growth rate of the average time consumption. The calculation method was shown in Eq. (9). We divided current average time consumption ($i$) by the prior setting's time consumption ($j$) to assess the growth rate; hence, the value of TGR had a positive relationship with the growth rate.

Fig. 10 presented the trend line of parameter DSM under Setting 1 to Setting 4. We observed that the increment of the input complexity which caused by either data package or encryption methods could result in a decline of the distribution. It implied that the time consumption became more concentrated as the input complexity increased. This phenomenon signified that a large number of input packages or a great amount of encryption methods could result in low performance of the proposed method.

The time consumption stably grew aligning with the increasing number of data packages and encryption methods, based on our observation on parameter TGR in Fig. 11. We observed that the average time consumption multiplied each additional five input data packages from $TGR_{2,1}$ to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2942293, IEEE Transactions on Cloud Computing



TABLE VII: Measurement Parameters Under Setting 3

| Setting 3-$i$ | $i$=1 | $i$=2 | $i$=3 | $i$=4 | $i$=5 |
|---|---|---|---|---|---|
| Ave | 0.851 | 1.248 | 1.621 | 1.963 | 2.255 |
| Min | 0.383 | 0.468 | 0.623 | 0.732 | 0.822 |
| Max | 5.993 | 6.956 | 8.809 | 9.707 | 10.513 |
| $DSM_i$ | 14.633 | 13.862 | 13.140 | 12.265 | 11.786 |
| $TGR_{i,i-1}$ | - | 1.467 | 1.299 | 1.211 | 1.149 |

TABLE VIII: Measurement Parameters Under Setting 4

| Setting 4-$i$ | $i$=1 | $i$=2 | $i$=3 | $i$=4 | $i$=5 |
|---|---|---|---|---|---|
| Ave | 2.593 | 4.247 | 6.029 | 7.384 | 8.507 |
| Min | 0.946 | 1.284 | 1.734 | 2.169 | 2.385 |
| Max | 14.023 | 18.214 | 23.700 | 28.355 | 30.187 |
| $DSM_i$ | 13.831 | 13.182 | 12.669 | 12.072 | 11.655 |
| $TGR_{i,i-1}$ | - | 1.638 | 1.420 | 1.225 | 1.152 |

$TGR_{5,4}$ in both Table V and Table VI. It implied that the number of data package had a exponential impact on the time consumption. Similarly, the average time consumption steadily increased each additional one encryption method from $TGR_{2,1}$ to $TGR_{5,4}$ in both Table VII and Table VIII. It implied that the number of encryption method had a linear impact on the time consumption. The changing trend somehow depicted a declining shape for all Setting 1-4 in Fig. 11. It implied a trend that the input complexity (either data package or encryption methods) would have less impact on the time consumption along with its increasing volume.

In summary, main findings from our experiments could be threefold. First, we found that our approach had a better performance in energy saving and preserving privacy (proportion of encrypted data) than representative greedy-based approach. It was logical because our approach produced optimal solution that should be superior to sub-optimal solutions (created by greedy algorithms). Second, the strategy generation time had a positive correlation with the complexity of the input. More volume of the input (either data package or encryption methods) could result in a longer generation time length. There was exponential correlation between data package number and the strategy generation time while data package number had linear dependence with the strategy generation time. Finally, the distribution of the generation time could became smaller with the increment of the input complexity. A similar situation also applied to the growth rate of the generation time. The phenomena implied that a large amount of input data packages or a great number of encryption options could cause low performance of the proposed approach.

## VII. Conclusion

In this paper, we focused on addressing potential privacy leakage due data over-collection by untrusted parties. Specifically, the proposed model uses fog computing as a security medium and supports multiple encryption methods for filtering access. An optimal scheduling (dynamic programming) was designed to maximize the capability of preserving privacy in resource-constrained settings. Findings from evaluations demonstrated the potential utility of our proposed model.

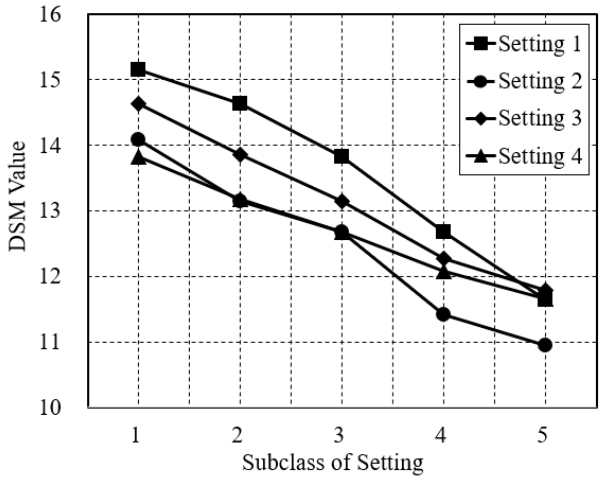


Fig. 10: The trend line of DSM under different settings.

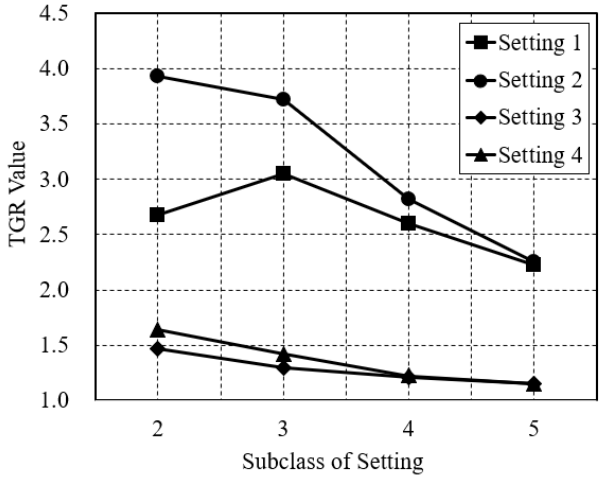


Fig. 11: The trend line of TGR under different settings.

## References

[1] X. Zhang, L. Yang, C. Liu, and J. Chen. A scalable two-phase top-down specialization approach for data anonymization using MapReduce on cloud. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):363–373, 2014.

[2] C. D'Orazio, K.K.R. Choo, and L.T. Yang. Data exfiltration from Internet of Things devices: iOS devices as case studies. *IEEE Internet of Things Journal*, 4(2):524–535, 2017.

[3] X. Hu, W. Pedrycz, and X. Wang. Fuzzy classifiers with information granules in feature space and logic-based computing. *Pattern Recognition*, 80:156–167, 2018.

[4] E. Kim, S. Oh, and W. Pedrycz. Reinforced hybrid interval fuzzy neural networks architecture: Design and analysis. *Neurocomputing*, 303:20–36, 2018.

[5] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2008.

[6] Brett Eterovic-Soric, Kim-Kwang Raymond Choo, Helen Ashman, and Sameera Mubarak. Stalking the stalkers - detecting and deterring stalking behaviours using technology: A review. *Computers & Security*, 70:278–289, 2017.

[7] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):222–233, 2014.

[8] K. Gai and M. Qiu. Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers. *IEEE Transactions on Industrial Informatics*, PP(99):1, 2017.

[9] J. Li, X. Chen, S. Chow, Q. Huang, D. Wong, and Z. Liu. Multi-authority fine-grained access control with accountability and its application in cloud. *Journal of Network and Computer Applications*, PP(99):1, 2018.

[10] R. Baranowski, M. Kochte, and H. Wunderlich. Fine-grained access management in reconfigurable scan networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(6):937–946, 2015.

[11] E. Damiani, S. De Capitani, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security*, 5(2):169–202, 2002.

[12] J. Shao, R. Lu, and X. Lin. FINE: A fine-grained privacy-preserving location-based service framework for mobile devices. In *The 33rd Annual IEEE International Conference on Computer Communications*, pages 244–252, Toronto, Canada, 2014. IEEE.

[13] K. Gai, M. Qiu, and M. Liu. Privacy-preserving access control using dynamic programming in fog computing. In *The 4th IEEE International Conference on Big Data Security on Cloud*, pages 126–132, Omaha, Nebraska, USA, 2018.

[14] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. Cyber-physical systems information gathering: A smart home case study. *Computer Networks*, 138:1–12, 2018.

[15] Z. Su, Q. Xu, J. Luo, H. Pu, Y. Peng, and R. Lu. A secure content caching scheme for disaster backup in fog computing enabled mobile social networks. *IEEE Transactions on Industrial Informatics*, PP(99):1, 2018.

[16] M. Yang, T. Zhu, B. Liu, Y. Xiang, and W. Zhou. Machine learning differential privacy with multifunctional aggregation in a fog computing architecture. *IEEE Access*, 6:17119–17129, 2018.

[17] C. Piao, Y. Shi, J. Yan, C. Zhang, and L. Liu. Privacy-preserving governmental data publishing: A fog-computing-based differential privacy approach. *Future Generation Computer Systems*, 90:158–174, 2019.

[18] L. Lyu, K. Nandakumar, B. Rubinstein, J. Jin, J. Bedo, and M. Palaniswami. PPFA: Privacy preserving fog-enabled aggregation in smart grid. *IEEE Transactions on Industrial Informatics*, 14(8):3733 – 3744, 2018.

[19] J. Kang, R. Yu, X. Huang, and Y. Zhang. Privacy-preserved pseudonym scheme for fog computing supported internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2627–2637, 2018.

[20] J. Wang, X. Zhang, Q. Zhang, M. Li, Y. Guo, Z. Feng, and M. Pan. Data-driven spectrum trading with secondary users' differential privacy preservation. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1, 2019.

[21] A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng. An attribute-based encryption scheme to secure fog communications. *IEEE Access*, 5:9131–9138, 2017.

[22] A. Datta and M. Joye. Cryptanalysis of a privacy-preserving aggregation protocol. *IEEE Transactions on Dependable and Secure Computing*, 14(6):693–694, 2016.

[23] L. Li, R. Lu, K.K.R. Choo, A. Datta, and J. Shao. Privacy-preserving-outsourced association rule mining on vertically partitioned databases. *IEEE Transactions on Information Forensics and Security*, 11(8):1847–1861, 2016.

[24] S. Basudan, X. Lin, and K. Sankaranarayanan. A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing. *IEEE Internet of Things Journal*, 4(3):772–782, 2017.

[25] J. Ni, A. Zhang, X. Lin, and X. Shen. Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine*, 55(6):146–152, 2017.

[26] Q. Wang, D. Chen, N. Zhang, Z. Ding, and Z. Qin. PCP: A privacy-preserving content-based publish–subscribe scheme with differential privacy in fog computing. *IEEE Access*, 5:17962–17974, 2017.

[27] J. Wei, X. Wang, N. Li, G. Yang, and Y. Mu. A privacy-preserving fog computing framework for vehicular crowdsensing networks. *IEEE Access*, 6:43776–43784, 2018.

[28] R. Yang, Q. Xu, M. Au, Z. Yu, H. Wang, and L. Zhou. Position based cryptography with location privacy: A step for fog computing. *Future Generation Computer Systems*, 78:799–806, 2018.

[29] M. Du, K. Wang, X. Liu, S. Guo, and Y. Zhang. A differential privacy-based query model for sustainable fog data centers. *IEEE Transactions on Sustainable Computing*, PP(99):1, 2017.

[30] M. Du, K. Wang, Z. Xia, and Y. Zhang. Differential privacy preserving of training model in wireless big data with edge computing. *IEEE Transactions on Big Data*, PP(99):1, 2018.

[31] M. Li, L. Zhu, and X. Lin. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4573–4584, June 2019.

[32] W. Teng, G. Yang, Y. Xiang, T. Zhang, and D. Wang. Attribute-based access control with constant-size ciphertext in cloud computing. *IEEE Transactions on Cloud Computing*, 5(4):617–627, 2015.

[33] D. Clinton, A. Gail-Joon, and T. Marthony. Policy-driven security management for fog computing: Preliminary framework and a case study. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration*, pages 16–23. IEEE, 2014.

[34] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li. Lightweight fine-grained search over encrypted data in fog computing. *IEEE Transactions on Services Computing*, PP(99):1, 2018.

[35] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu. An efficient access control scheme with outsourcing capability and attribute update for fog computing. *Future Generation Computer Systems*, 78:753 – 762, 2018.

[36] J. Hong, K. Xue, N. Gai, D. Wei, and P. Hong. Service outsourcing in F2C architecture with attribute-based anonymous access control and bounded service number. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1, 2018.

[37] H. Ma, R. Zhang, S. Sun, Z. Song, and G. Tan. Server-aided fine-grained access control mechanism with robust revocation in cloud computing. *IEEE Transactions on Services Computing*, PP(99):1, 2019.

[38] R. Ahuja and S. Mohanty. A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage. *IEEE Transactions on Cloud Computing*, PP(99):1, 2017.

[39] L. Fei, R. Yogachandran, C. Mauro, and R. Muttukrishnan. Robust access control framework for mobile cloud computing network. *Computer Communications*, 68:61–72, 2015.

[40] M. M. Baqer, A. M. A. Kalam, and V. Athanasios. Secure data sharing and searching at the edge of cloud-assisted internet of things. *IEEE Cloud Computing*, 4(1):34–42, 2017.

[41] Z. Su, F. Biennier, Z. Lv, Y. Peng, H. Song, and J. Miao. Toward architectural and protocol-level foundation for end-to-end trustworthiness in cloud/fog computing. *IEEE Transactions on Big Data*, PP(99):1, 2017.

[42] Z. Bilel, A. H. Kaffel-Ben, and H. Wafa. Generic access control system for ad hoc MCC and fog computing. In *International Conference on Cryptology and Network Security*, pages 400–415. Springer, 2016.

[43] K. Gai, K.K.R. Choo, M. Qiu, and L. Zhu. Privacy-preserving content-oriented wireless communication in Internet-of-Things. *IEEE Internet of Things Journal*, 5(4):3059–3067, 2018.

[44] K. Gai and M. Qiu. Reinforcement learning-based content-centric services in mobile sensing. *IEEE Network*, 32(4):34–39, 2018.

[45] K. Gai, M. Qiu, Z. Ming, H. Zhao, and L. Qiu. Spoofing-jamming attack strategy using optimal power distributions in wireless smart grid networks. *IEEE Transactions on Smart Grid*, 8(5):2431 – 2439, 2017.

[46] K. Gai, M. Qiu, M. Liu, and Z. Xiong. In-memory big data analytics under space constraints using dynamic programming. *Future Generation Computer Systems*, 83:219–227, 2018.

[47] W. Dai, L. Chen, M. Qiu, A. Wu, and M. Liu. DASS: A web-based fine-grained data access system for smartphones. In *The 2nd IEEE International Conference on Smart Cloud*, pages 238–243, New York, USA, 2017. IEEE.