Fog-Aided Verifiable Privacy Preserving Access Control for Latency-Sensitive Data Sharing in Vehicular Cloud Computing

Kaiping Xue, Jianan Hong, Yongjin Ma, David S. L. Wei, Peilin Hong, and Nenghai Yu

Abstract

VCC is an emerging computing paradigm developed for providing various services to vehicle drivers, and has attracted more and more attention from researchers and practitioners over the last few years. However, privacy preserving and secure data sharing has become a very challenging and important issue in VCC. Unfortunately, existing secure access control schemes consume too many computation resources, which prevents them from being performed on computing resource constrained vehicle onboard devices. Also, these cloud-based schemes suffer large latency and jitter due to their centralized resource management, and thus may not be suitable for real-time applications in VANETs. In this article, we thus propose a novel fog-to-cloud-based architecture for data sharing in VCC. Our scheme is a cryptography-based mechanism that conducts fine-grained access control. In our design, the complicated computation burden is securely outsourced to fog and cloud servers with confidentiality and privacy preservation. Meanwhile, with the prediction of a vehicle's mobility, pre-pushing data to specific fog servers can further reduce response latency with no need to consume more resources of the fog server. In addition, with the assumption of no collusion between different providers for the cloud and fog servers, our scheme can provide verifiable auditing of fog servers' reports. The scheme is proved secure against existing adversaries and newborn security threats. Experimental test shows significant performance improvement in edge devices' overhead saving and response delay reduction.

Introduction

The increasing demands on improving driving safety, traffic efficiency, and entertainment have brought more and more attention from researchers and practitioners on vehicular ad hoc networks (VANETs). In VANETs, due to the needs for data collection and dissemination, the vehicles themselves, the communications among vehicles, and the communications between vehicles and the infrastructure unsurprisingly generate huge amounts of data. In order to enhance the scalability and improve the quality of service (QoS) of VANETs, an emerging computing paradigm that integrates the technologies of cloud computing and VANETs, called vehicular cloud computing (VCC), is proposed [1–3]. Meanwhile, numerous cryptography-based schemes have been proposed to address the security and privacy challenges in the semi-trust cloud model. With these security mechanisms, even sensitive data can be outsourced to the public cloud while still guaranteeing confidentiality, integrity, and other security features [4–6].

However, the following two issues impede VCC from being widely adopted:

- Moving huge volumes of data from the network edge to a central data center for processing and analysis not only adds latency [7], but also consumes limited network bandwidth in VANETs. Thus, data sharing applications that are sensitive to latency or jitter, such as live video streaming, may not work well in this service model.
- The equipped onboard devices on vehicles are usually resource constrained, and may not be able to support highly complex cryptographic algorithms. Moreover, the increased delay in implementing these complicated algorithms will result in poor user experience.

As such, in fog computing, fog servers are employed and deployed between the cloud and network edge to enable users to get better services. Therefore, in VCC, by allocating adequate resources at the edge of networks, only one hop from users' mobile devices, fog computing can provide more reliable and higher-quality services to end users. However, taking into account the increasingly popular data sharing service, the fog alone cannot completely replace the cloud, as this service paradigm usually requires massive storage and some other auxiliary functions that fog servers cannot undertake alone. Therefore, fog-to-cloud architecture [8, 9] has been proposed to make full use of powerful computing and storage resources of the cloud, while the integrated fog system helps avoid high latency.

This article focuses on the scenarios where fog and cloud computing are used to both leverage the performance of VANETs and support some features needed in various real-time applications of data sharing. Onboard devices in vehicles usually have limited computing resources, which means that complicated encryption and decryption tasks would not be able to be directly imple-

Kaiping Xue, Jianan Hong, Yongjin Ma, Peilin Hong, and Nenghai Yu are with the University of Science and Technology of China; David S. L. Wei is with Fordham University

Digital Object Identifier: 10.1109/MNET.2018.1700341

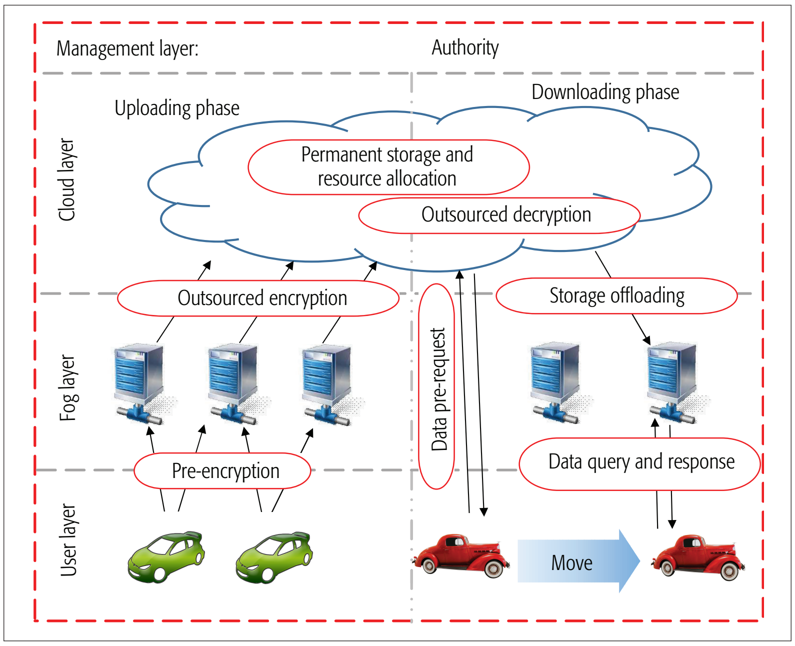

FIGURE 1. An illustration of fog-to-cloud data sharing architecture.

mented on them. Furthermore, fine-grained access control is one of the most important demands we want to achieve in a VCC system, but its cryptographic techniques, such as attribute-based encryption (ABE) [10], are usually highly complex. To address this issue, an encryption and decryption outsourcing scheme is adopted, which leaves only light tasks for users. However, the existing related works still have some disadvantages in the scenario.

Encryption Outsourcing: Few works have paid attention to the area of encryption outsourcing. Li *et al.* [11] first proposed the idea that an energy-constrained device sends plaintext to a more powerful entity and lets it execute ABE encryption. In this scenario, the powerful entity is assumed to be fully trusted (e.g., user's laptop or PDA). However, this approach is not suitable for the fog-to-cloud architecture, as fog servers cannot be assumed to be fully trusted by vehicle drivers. Thus, a scheme for the semi-trust fog-to-cloud model should be adopted.

Decryption Outsourcing: In cloud computing, decryption outsourcing very successfully achieves the objective in terms of security and efficiency. For instance, the work of [12] only leaves one exponentiation, one multiplication, and symmetric decryption for users, without leaking any information to the cloud. However, a fog system can hardly replace the cloud, since the servers need to maintain users' attribute-related keys and store the outsourced data for a long time. Moreover, taking users' mobility into account, to let fog servers undertake the decryption outsourcing requires duplicated storage of users' keys on fog servers. This will consume too much storage resource and may leak users' private information.

In this article, we propose a fine-grained data access control scheme for a fog-to-cloud-based VANET architecture in which the users (the data owner and/or the consumer) of vehicle onboard devices only need lightweight operations. To achieve this goal, the encryption task and decryption task are outsourced to semi-trusted fog servers and the cloud server, respectively. From the perspective of efficiency improvement, cloud can execute the outsourced decryption in advance based on a user's prediction and push the transformed data to one or more selected fog servers. As the cloud and multiple fogs are usually provided by different service providers, the fog should convince the cloud of the validity of its claimed service amount such that the payment between them can be conducted and completed [13]. In fact, selfish fogs may fake the amount to charge more illegally. Our design of interaction between vehicle onboard devices and fog server can help prevent this threat, while introducing only light overhead and still preserving privacy.

The main contributions of this article include:

•To our best knowledge, our work is the first secure and efficient fine-grained data access control framework for VCC based on fog-to-cloud architecture. Our scheme efficiently reduces vehicles' overhead and service latency, while still maintaining privacy preservation.

•An encryption/decryption outsourcing mechanism is integrated into our proposed access control framework for VCC, where the computationally complicated encryption is outsourced to two non-colluding fogs, and the computationally complicated decryption is outsourced to the cloud. Based on the assumption of no collusion, neither the cloud or fog servers will get private information about users' outsourced data; nor will any other unauthorized entities. Meanwhile, data can be offloaded to the fogs in advance based on a user's mobility prediction. Hence, latency-sensitive data sharing services can be deployed with QoS guarantee.

•The proposed interactive protocol between fog servers and vehicle onboard devices can truthfully record fog servers' real service amount, with the preservation of users' anonymity and unlinkability.

System Model

As shown in Fig. 1, by integrating the advanced technologies of cloud computing and fog computing, the whole data sharing system is composed of five entities, namely authority, cloud server, fog servers, data owners, and users.

Architecture

The roles of these components are as follows.

Authority: It is responsible for managing the security protection of the data sharing system by publishing system parameters and issuing secret keys to other entities.

Cloud: It supports other entities with its unlimited storage and strong computing power. As a core component of this service paradigm, it is responsible for permanent data storage, decryption outsourcing, resource allocation, and other related tasks.

Fog: It can take some computation and temporary storage for other entities with the consideration of service quality, including reducing latency and confidentiality preservation.

Owner: A vehicle onboard device user in VANETs, who generates, publishes, and shares data. It gathers data nearby and uploads them in an encrypted form. However, as an energy-constrained entity, it can only bear lightweight operations, like symmetric encryption.

User: It is also a vehicle onboard device user in VANETs, who queries and obtains the stored data in this system. Due to the practical demand in vehicular networks, it cannot tolerate a long delayed response. Additionally, a heavy decryption burden is not acceptable due to its energy constraint.

The details of the architecture are expressed along the horizontal dimension and along the vertical one, as depicted in Fig. 1.

On the aspect of layered structure, the system can be divided into four layers. The management and cloud layers are responsible for the system's security and performance, respectively. In order to provide more convenient service, the cloud layer transfers some computationally complicated tasks or offloads specific data in advance to entities in the lower layer. The fog layer comprises multiple fog servers, which are located at the edge of networks and are usually managed by different service providers. Different from the entities in the user layer, fog servers can conduct some complicated computations, while also providing temporary data caching/storage. Our scheme aims at the scenario where cloud and fog servers are implemented by different providers. Note that the cloud is also responsible for charging the right amount to fog servers according to the services provided by the fog servers deployed on entities, such as roadside infrastructures, network access points, and base station control units, at the network edge. The user layer is composed of diverse vehicles including vehicle onboard device users, who are either data owners, consumer users, or both. Vehicle onboard device users are the clients in this service model, who are assumed to be computing power constrained.

On the aspect of system flows, there are two phases:

- The uploading phase is associated with the procedure of data encryption and uploading. The entities involved include owners, fog servers, and the cloud.
- The downloading phase is responsible for data decryption and the response to data requests.

The entities involved include users, the fog system, and the cloud. In particular, fog servers in these two phases undertake different tasks. In the uploading phase, fog servers mainly conduct outsourced encryption with designed access policy, which consumes fog servers' computation resources. In the downloading phase, fog servers implement the temporary data caching and respond to users' data requests, which mainly rely on fog servers' storage resources.

Security Assumption and Requirements

Similar to many works on public cloud storage security [14, 8], the cloud is assumed to be honest but curious in this article. It first offers a reliable service and promotes service quality in order to attract more clients. However, it may try to gain unauthorized information for its own benefits. The fog servers are also assumed to be honest but curious, but the system's security threats from them are somewhat different from those of the cloud. In the uploading phase, we assume that fog servers obey the correct access policies to encrypt the shared data. However, before encrypting the data, fog servers may also be interested in the private information in these data. In addition, each vehicle onboard device is usually equipped with multiple network interfaces, which makes it possible to connect many access points that are maintained by different providers simultaneously. Thus, it is reasonable to assume that it is not hard to find a pair of fog servers that will not collude with each other. Also, the mobility of a vehicle can help connect non-colluding fog nodes. In the downloading phase, besides the curiosity regarding the data, fog servers may also try to present a false service amount report to the cloud in order to gain more pay. The authority, as a core entity in the management layer, can be totally trusted, strictly issuing a secret key to each user according to its owned attributes. We assume that some malicious users will try to obtain unauthorized data by any means, including colluding with other users.

Users with mobile devices usually download and access the shared data within some locations, where the geographical location and users' interesting data can be predicted. This prediction can be utilized to reduce the access delay and computation overhead if a fog system is integrated into the system.

Based on the above security assumption for each entity, the data sharing framework in this article should be designed to possess the following security features.

Data Confidentiality: It includes two aspects. First, unauthorized users cannot access the data by any means. Second, the outsourced encryption and decryption tasks will not leak any knowledge of the data.

Verifiable Service Witness: In the downloading phase, a fog server's service includes data caching and user query response. The former is exactly known to the cloud, but the latter is reported by the fog server. Thus, an effective mechanism is critical to resist fake reporting of the service amount.

User Anonymity and Unlinkability: Privacy is one of the most critical issues in vehicular network systems since the leakage of identity information will further expose more information, such as locations. To achieve the above requirement, when a user requests some data, some credentials should be presented to a fog server for accessing the data. During this phase, a user's identity should be concealed against the fog server [15]. Furthermore, a user's multiple data accesses cannot be linked by a fog server.

Preliminaries of Attribute-Based Encryption

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [10, 12] has been utilized to conduct fine-grained data access control for public cloud storage.

This section briefly introduces the basic procedures and features of the CP-ABE scheme in [12] in five steps.

Setup. This algorithm publishes the public parameters and securely stores a master secret key for the whole system.

Encrypt. It uses public parameters to encrypt a file under and access policy, and then uploads it to the cloud.

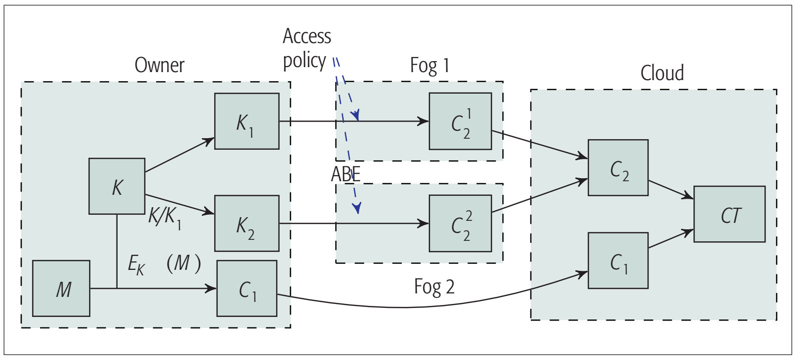

FIGURE 2. Procedures of data encryption outsourcing.

Key Generation. Based on a user's attribute set, this algorithm generates the secret key. A secret key is composed of two parts: the transformation key (*TK*), which is associated with a user's attributes stored in the cloud,; and the private key (*SK*), which should be securely distributed to the user, and without which, any entity cannot finally decrypt the file even with satisfied *TK*.

Transform. When a user's attribute set satisfies the access policy, this algorithm uses the user's *TK* to transform the ciphertext to a new format. We use *CT* to represent the original ciphertext that is generated in the *Encrypt* step, and *CT*′ is the notation of the transformed one.

Decrypt. When a user gets a corresponding *CT*′, it can decrypt it with *SK* and recover the plaintext.

Our scheme is similar to the scheme in [12] with some modifications.

Performance. The outsourced *Transform* step undertakes almost all of the complicated burden, leaving only an ELGamal-like decryption for the user.

Confidentiality. CP-ABE requires that only by satisfying the access policy can a user decrypt the file.

Format of CT. In [12], *CT* can be expressed as a vector, composed of a series of components, denoted by $(x_1, x_2, ..., x_n)$, in which n is the length of the vector.

Our Modification. The work of [12] only encrypts a randomly selected K, and K is a symmetric key to encrypt the shared file. Thus, to practically use this algorithm, we redefine the format of *CT* as: $\{C_1 = E_k(M), C_2 = (x_1, x_2, ..., x_n)\}$. Accordingly, *CT*′ is expressed with two sectors $(C_1, C_{3,i})$ if it is transformed using user U_i's *TK*.

The data confidentiality algorithm adopted as an element in our system basically follows the scheme of [12], which provides a more detailed description of CP-ABE.

Construction

The system and the interaction between different components in our scheme are introduced in three parts: the management phase, uploading phase, and downloading phase. The latter two phases are briefly depicted in Fig. 1, and the first phase involves system setup and users' secret key distribution.

Management Phase

This phase mainly consists of the system's setup and users' secret key generation. The setup is for the authority to generate the master secret key *MSK* and the public parameters *PK*, where *MSK* is for users' key generation, and the latter is for owners or fog servers to encrypt the shared data. Specifically, in our scheme, a hash function $H : (0, 1)^* \rightarrow (0, 1)^n$ is defined as one component of *PK*.

A user's secret key is generated according to the attribute set, following the key generation step of [12] presented in the above section.

Uploading Phase

The encrypt step in traditional CP-ABE brings about too much computation overhead for data owners. Thus, securely outsourcing the complicated computation task is necessary. This phase involves the owner, two non-colluding fog servers (denoted by F_1, F_2), and the cloud, as shown in Fig. 1. Furthermore, a brief procedure depiction of the encrypt phase between the three layers is shown in Fig. 2.

To share a file M, first, the owner randomly selects K, K_1 and computes: $C_1 = E_K(M)$, $K_2 = K/K_1$. K_1 and K_2 are securely sent to F_1 and F_2, respectively, and C_1 is sent to the cloud.

Fog node F_1 obtains K_1 and implements the encrypt step to encrypt K_1 with an access policy P, where the policy can be determined by the owner or the fog servers, depending on the practical scenarios. The outcome of this step is organized as $C_2^1 = (x_1^1, x_2^1, ... , x_n^1)$. The other fog server, F_2, receives K_2, encrypts it with the same P, and outputs $C_2^2 = (x_1^2, x_2^2, ... , x_n^2)$.

The two fog servers send C_2^1 and C_2^2 to the cloud. The cloud computes $C_2 = (x_1, x_2, ..., x_n)$, where $x_i = x_i^1 \cdot x_i^2$, $\forall 1 \leq i \leq n$. Together with C_1, $CT = \{C_1, C_2\}$ is an encryption result of M with access policy P.

Note that each shared file has a unique file ID B_j, which can be provided in different messages to correlate C_1, C_2^1, and C_2^2, and it is also used in subsequent steps.

Downloading Phase

Users with mobile devices usually download and access shared data within certain locations, where the geographical location and users' interesting data can be predicted by some means. This prediction can be utilized to reduce the access delay and computation overhead if the fog system is integrated into the system. According to the depiction of this phase in Fig. 1, we describe it in more detail in the following two sub-phases.

Data Pre-Request: The user (denoted by U_i) predicts its future visiting place based on its mobility trend and the data it wants to access there. Then it sends a pre-request message to the cloud, including interesting data, location, and time information.

Upon receipt of a pre-request message, the cloud randomly selects a temporary pseudonym Pse_i. The cloud securely caches the identity and pseudonym pair (U_i, Pse_i) and securely sends Pse_i to U_i.

The cloud searches for and selects a user's requested data. It checks whether U_i can access each of the ciphertexts (denoted by CT_j with file ID B_j) with U_i's attribute set, which is implicitly embedded in its transformation key *TK*. If the access should be denied, the cloud simply drops the request. Otherwise, the cloud executes the transform step and outputs $CT_j' = (C_{1,j}, C_{3,ji})$. Then the cloud computes $AUTH_{ji} = H^2(B_j || Pse_i)$ to complete the decryption outsourcing.

Based on U_i's request location, the cloud selects a fog server and sends $\{B_j, AUTH_{ji}, CT'\}$ to it.

Data Query and Downloading: Considering that the same fog server will obtain multiple transformed ciphertexts with the same B_j, the cached contents are organized as shown in Table 1. For multiple requests for the same file from different users, the fog server can cache only one copy of C_1, but different C_3. As the sector C_1 contains the encryption form of real files, which has far larger volume than C_3, such structure can well reduce the storage and communication overhead of fog servers.

When user U_i gets close to the predicted fog server, to request a file with B_j, U_i computes $CHECK_{ji} = H(B_j || Pse_i)$ and includes it in the request and sends it to the fog server. The fog server examines the cached data with ID B_j to make sure that there is an item whose authentication satisfies $AUTH_{ji} = H(CHECK_{ji})$. If none exists, the fog server will abort the interaction and indicate that there is no relevant ciphertext for U_i. Otherwise, the fog server gets the corresponding $C_{3,ji}$, and sends $(C_{1,j}, C_{3,ji})$ to U_i. U_i executes the decrypt step to recover the content M of this file.

After the interactions, the fog server obtains $CHECK_{ji}$, which can be used to prove that the service has already been provided for users. Thus, the fog server can request the pay from the cloud based on its service amount.

All files stored in the cloud are in encrypted format to prevent entities, including the cloud server, from obtaining private information. Although every user's transformation key (TK) is dominated by the cloud for decryption outsourcing, the cloud cannot get any content due to the lack of the user's private key.

File ID	Sector 1	Sector 2	Authentication
B_{j1}	$C_{1,j1}$	$C_{3,j1i1}$ $C_{3,j1i2}$ $C_{3,j1i3}$ …	$AUTH_{j1i1}$ $AUTH_{j1i2}$ $AUTH_{j1i3}$ …
B_{j2}	$C_{1,j2}$	$C_{3,j2i1}$ $C_{3,j2i4}$ …	$AUTH_{j2i1}$ $AUTH_{j2i4}$ …
Others	…	…	…

TABLE 1. Cached ciphertext in the fog.

Security and Performance Analysis

This section aims to illustrate the security features and analyze the performance of our proposed scheme compared to existing cloud-based data sharing systems.

Security Analysis

Confidentiality against Unauthorized Users: The basic CP-ABE scheme in [12] has been proved semantically secure against chosen-plaintext adversaries. In detail, this security protection level ensures that only authorized users whose attribute sets satisfy the access policy can successfully decrypt the data. On the contrary, unauthorized users are not able to access the plaintext even if they have obtained the encrypted data. In addition, collusion attacks can be resisted. Unauthorized users cannot earn additional benefits by colluding with cloud, fog servers, or other unauthorized users. In particular, the prevention of collusion between multiple unauthorized users is achieved by the random blindness in each user's secret key.

Confidentiality against Cloud and Fog: All files stored in the cloud are in the encrypted format to prevent entities, including the cloud server, from obtaining the private information. Although every user's transformation key (TK) is dominated by the cloud for decryption outsourcing, the cloud cannot get any content due to the lack of the user's private key.

Meanwhile, in the downloading phase, fog servers dominate no more useful information than the cloud, and thus confidentiality can also be preserved against them. When analyzing fog servers in the uploading phase, we find that each of the two selected fog servers only obtains $C_1 = E_K(M)$ and a masked $K_b (b = 1,2)$. Based on the assumption of no collusion between two fog servers, only dominating either K_1 or K_2 but not both of them will gain no advantage for the knowledge of symmetric key K. Therefore, confidentiality against fog servers is well preserved.

Fog Cheating Resistance: Users should pay the cloud for their received data services, and meanwhile the cloud needs to pay specific fog servers' providers for their assistance with services by offering computing, storage, and transmission. A new security threat arises that the fog servers may fake their service amounts to get more pay. In this section, we thoroughly analyze the protection mechanism against this threat.

For any user U_i and accessed file F_j, before responding to the service request, the fog only caches $AUTH_{ji} = H^2(B_j || Pse_i)$. Without knowing the user's current pseudonym, the fog cannot guess $CHECK_{ji}$, which is the pre-image of $AUTH_{ji}$ in hash function. Thus, the service amount can be estimated by the obtained $CHECK_{ji}$ as the witnesses. As a fog cannot generate a correct witness without the help of the corresponding user, only after responding to the service request of the user can the fog server get it. Due to the feature of one-way hash function, the fog server can easily verify the validity of $CHECK_{ji}$ by executing $H(CHECK)_{ji}$.

Collusion between a fog server and a user (i.e., the user submits $CHECK_{ji}$ without receiving service) is also considered. The users pay the cloud based on the amount of received service. Thus, although the service can be partially offloaded to fog servers, users still directly pay the cloud. As the user is usually a rational one from the perspective of profit, and to hand over the witnesses means increasing its own payment, the user will not hand over $CHECK_{ji}$ without enjoying the requested service.

User's Identity Privacy: It should be noted that a user's identity cannot be concealed from the cloud, since the cloud is responsible for decryption outsourcing and service offloading according to the user's identity. Thus, identity privacy in this article only takes the fog servers in the downloading phase into account.

Model	Cloud-only model	Li *et al.*'s model [11]		Proposed scheme
		Sensor	Laptop	
Encrypt/ms	$2.3 + 1.35C_P + T_{sym}$	0	$2.3 + 1.35C_P + T_{sym}$	$T_{sym} + 0.00083$
Transferred payload/byte	$64 + 32\,C_P + \|M\|$	$\|M\|$	$64 + 32\,C_P + \|M\|$	$64 + \|M\|$

• C_P is the complexity of access policy, quantified by the embedded attributes.
• M is the size of file that is encrypted using symmetric key.
• T_{sym} is the consumed time of symmetric encryption, which varies according to $|M|$.

TABLE 2. Owner's uploading cost.

From the interactions, a fog server can get $H(B_j || Pse_i)$ of user U_i. Only the cloud and U_i know the association between the user's unique identity U_i and current pseudonym Pse_i. Thus, the fog server cannot guess the user identity from this interaction. From the other perspective, this protection can help further hide a user's access interests, which is also an important need in privacy.

Additionally, when receiving two access requests for different files, the fog server will obtain two checks (denoted by $H(B_{j1} || Pse_{i1})$ and $H(B_{j2} || Pse_{i2})$, respectively). With only the knowledge of B_{j1} and B_{j2}, the fog cannot distinguish whether $Pse_{i1} = Pse_{i2}$ or not. Thus, the unlinkability of users' different requests is preserved.

Note that the implementation of temporary pseudonyms can further resist an adversary's privacy guessing: Compared to the scheme using a user's real identity, the adversary should traverse far larger space to guess a user's identity from the authentication interactions.

Discussion: It is worth noting that in order to achieve the latter two security requirements, only one round of security-related signals is needed in the pre-request interaction between user and cloud. In this interaction, only one pseudonym is needed to provide a user's anonymity and unlinkability.

Performance Analysis

In this section, we analyze the performance of our scheme compared to the traditional cloud-based data sharing model based on the following platform.

A fog server is emulated and run on a standard 64-bit Fedora Release 21 operating system with Intel (R) Core (TM) i3-4130 3.4 GHz. The rented cloud server is provided by Aliyun. The user device is simulated as a laptop on a 64-bit Windows 7 operating system with Intel (R) Core (TM) i7-4790, 3.6 GHz. The utilized network is a real topology in our laboratory, while about 50 devices access the LAN simultaneously as the background flows. The link between fog and user is a two-hop one, one of which is WiFi access. In the real world, a user's device is usually much weaker than that in our experiment, but it will not affect the performance comparison result between the compared schemes too much.

The related work about attribute-based encryption is run in a C-based program, using PBC library 0.5.14 with type-A curve. Hash function is executed using SHA256 in Python 2.7.8. Symmetric encryption is executed as $E_{H(K)}(M)$, where $E(\cdot)$ is AES256 with CFB mode in OpenSSL 1.0.1k, and $H(\cdot)$ is the hash function introduced above.

We discuss the encryption cost of owners and other efficiency performance on some important aspects in what follows.

Uploading Cost of Owner: For almost all CP-ABE based schemes, symmetric encryption is used in the uploading phase for data owners, and will perform the same if they use the same encryption algorithm. Despite the fact that it is the traditional cloud only model, Li *et al.*'s scheme [11] is also used for comparison. Table 2 illustrates the owner's computation and communication overhead comparison between the different schemes.

In terms of computation, Li *et al.*'s scheme [11] does not save on encryption cost, but it migrates the heavy task from an energy-constrained device to another that can afford the ABE-based encryption tasks. Different from Li *et al.*'s scheme [11], our scheme allows the owner to outsource heavy tasks to a semi-trusted fog server, leaving only lightweight tasks for himself/herself. As shown in the table, despite the task of symmetric encryption (denoted as T_{sym}), the proposed scheme leaves negligible cost for the owner. Although T_{sym} varies with respect to the file size, its overhead is far lighter than that of conducting any asymmetric algorithm.

In terms of communication, the result is similar to the comparison outcomes of computation. Li *et al.*'s scheme [11] can migrate heavy tasks from energy-constrained user devices to a more powerful one, which only needs to send the plaintext to the laptop. Our scheme can further save the communication cost, as the owner only needs to transmit two symmetric keys to two non-colluding fog servers.

Response Delay: In the downloading phase, utilizing fog servers can reduce the interaction delay of implementing the request/response signals, as some tasks can be completed in advance, and then the mission can be conducted only at the edge of the network. Under the assumption that cloud and fog are in different management domains requires a little more time to authenticate the data access request. Next, we measure and analyze whether this trade-off is worthwhile.

Table 3 presents the transmission and computation time for two models. The average time (avg.) roughly indicates the latency feature of the schemes, and the standard deviation (*stdev.*) indicates the jitter from some aspect.

Our proposed scheme significantly reduces the transmission (or round-trip) time due to the adoption of the fog-to-cloud architecture. From Table 3, we can see that over 30 ms is saved. The difference of *stdev*. time between the two schemes is minor, because the first wireless hop at the user end is very unstable. In fact, if WiFi is replaced by a cellular network, this effect can be basically removed. For example, latency in 5G should be less than 1 ms for latency-sensitive service. Thus, the proposed scheme will have smaller latency and negligible jitter, while the jitter in the cloud-only model remains about 3.5 ms based on our measurement. The additional authentication/verification procedure in our scheme consumes only 0.25 μs, taking into account the execution time by both fog server and user, which is far less than the transmission time gap between the two schemes. Even taking the user's authentication execution into account, this procedure only spends 0.42 μs.

Based on the measurements shown above, the computation time can be ignored in our proposed scheme. It is also shown that our scheme, the fog-to-cloud model, far outperforms the cloud-only model in latency-sensitive data sharing.

Conclusion

In this article, we propose a fine-grained access control scheme under fog-to-cloud-based VANET architecture. The use of fog servers is to reduce latency, as well as to reduce vehicle onboard devices' encryption burdens. In the proposed scheme, the prediction of vehicles' mobility can help the cloud push useful data to the specific fog server, such that the data access latency constraint can be satisfied. Security protection in our scheme not only preserves data confidentiality when outsourcing computations, but also solves other security threats in the new service model. In addition, the proposed interactive protocol provides an approach of convincing pay contract between cloud and fog servers. The security analysis is presented briefly, but thoroughly illustrates that our covered security and privacy issues were successfully addressed. The performance measurements based on real-world experimentation validate the superiority of our scheme in latency reduction and user-side computation saving. Thus, data sharing between vehicles under the fog-to-cloud model is worthwhile to explore further.

Acknowledgment

This work is partly supported by the National Key Research and Development Plan of China under Grant No.2016YFB0800301, the National Natural Science Foundation of China under Grant Nos. 61379129 and No. 61671420, Youth Innovation Promotion Association CAS under Grant No. 2016394, and the Fundamental Research Funds for the Central Universities.

	Cloud-only model		Proposed scheme	
	Avg.	Stdev.	Avg.	Stdev.
Transmission time /ms	36.121	8.882	3.542	8.163
Computation time /ms	N/A		2.5×10^{-4}	

TABLE 3. Delay measurement.

References

[1] R. Yu *et al.*, "Toward Cloud-Based Vehicular Networks with Efficient Resource Management," *IEEE Network*, vol. 27, no. 5, Sept./Oct. 2013, pp. 48–55.
[2] C. Huang *et al.*, "PTVC: Achieving Privacy-Preserving Trust-Based Verifiable Vehicular Cloud Computing," *Proc. 2016 IEEE GLOBECOM*, 2016, pp. 1–6.
[3] A. Alamer, Y. Deng, and X. Lin, "A Privacy-Preserving and Truthful Tendering Framework for Vehicle Cloud Computing," *Proc. 2017 IEEE ICC*, 2017, pp. 1–7.
[4] H. Li *et al.*, "Engineering Searchable Encryption of Mobile Cloud Networks: When QoE Meets QoP," *IEEE Wireless Commun.*, vol. 22, no. 4, Aug. 2015, pp. 74–80.
[5] Y. Zhang *et al.*, "Efficient Public Verification of Data Integrity for Cloud Storage Systems from Indistinguishability Obfuscation," *IEEE Trans. Info. Forensics and Security*, vol. 12, no. 3, 2017, pp. 676–88.
[6] K. Xue *et al.*, "RAAC: Robust and Auditable Access Control with Multiple Attribute Authorities for Public Cloud Storage," *IEEE Trans. Info. Forensics and Security*, vol. 12, no. 4, 2017, pp. 953–67.
[7] R. Roman, J. Lopez, and M. Mambo, "Mobile Edge Computing, Fog *et al.*: A Survey and Analysis of Security Threats and Challenges," *Future Generation Computer Systems*, vol. 78, no. 2, 2018, pp. 680–98.
[8] X. Masip-Bruin *et al.*, "Foggy Clouds and Cloudy Fogs: A Real Need for Coordinated Management of Fog-to-Cloud Computing Systems," *IEEE Wireless Commun.*, vol. 23, no. 5, Oct. 2016, pp. 120–28.
[9] A. Sinaeepourfard *et al.*, "A Novel Architecture for Efficient Fog to Cloud Data Management in Smart Cities," *Proc. 37th IEEE Int'l. Conf. Distrib. Comp. Systems*, 2017, pp. 2622–23.
[10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc. 28th IEEE Symp. Security and Privacy*, 2007, pp. 321–35.
[11] M. Li, W. Lou, and K. Ren, "Data Security and Privacy in Wireless Body Area Networks," *IEEE Wireless Commun.*, vol. 17, no. 1, Feb. 2010.
[12] M. Green *et al.*, "Outsourcing the Decryption of ABE Ciphertexts," *Proc. USENIX Security Symp.*, 2011.
[13] Y. Zhong *et al.*, "ESTRA: Incentivizing Storage Trading for Edge Caching in Mobile Content Delivery," *Proc. 2015 IEEE GLOBECOM*, 2015, pp. 1–6.
[14] J. Li *et al.*, "Privacy-Preserving Public Auditing Protocol for Low-Performance End Devices in Cloud," *IEEE Trans. Info. Forensics and Security*, vol. 11, no. 11, 2016, pp. 2572–83.
[15] S. Yi, Z. Qin, and Q. Li, "Security and Privacy Issues of Fog Computing: A Survey," *Proc. Int'l. Conf. Wireless Algorithms, Systems, and Application*, Springer, 2015, pp. 685–95.

Biographies

Kaiping Xue [M'09, SM'15] received his B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an associate professor in the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security.

Jianan Hong received his B.S. degree from the Department of Information Security, USTC, in 2012. He is currently working toward a Ph.D. degree in information security from the Department of EEIS, USTC. His research interests include secure cloud computing and mobile network security.

Yongjin Ma received his B.S. degree from the Department of Information Security, USTC, in July, 2017. He is currently a graduate student in information and communication engineering in the Department of EEIS, USTC. His research interests include next-generation Internet and network security.

David S. L. Wei [SM'07] received his Ph.D. degree in computer and information science from the University of Pennsylvania in 1991. He is currently a professor in the Computer and Information Science Department at Fordham University. Currently, His research interests include cloud computing, big data, IoT, and cognitive radio networks.

Peilin Hong received her B.S. and M.S. degrees from the Department of EEIS, USTC, in 1983 and 1986. Currently, she is a professor in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security.

Nenghai Yu received his B.S. degree from Nanjing University of Posts and Telecommunications, China, in 1987, his M.E. degree from Tsinghua University, Beijing, China, in 1992, and his Ph.D. degree from USTC in 2004. Currently, he is a professor in the Department of Information Security and Department of EEIS, USTC. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.