Verifiable and privacy-preserving fine-grained data management in vehicular fog computing: A game theory-based approach

Zahra Seyedi
Amirkabir Univeristy of Technology
Farhad Rahmati
Amirkabir Univeristy of Technology
Mohammad Ali (mali71@aut.ac.ir)
Amirkabir Univeristy of Technology
Ximeng Liu
Fuzhou University

Research Article

Keywords: Vehicular fog computing, encrypted data processing, data retrieval, data management, Nash equilibrium

Posted Date: July 31st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3165924/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Peer-to-Peer Networking and Applications on December 20th, 2023. See the published version at https://doi.org/10.1007/s12083-023-01601-x.

Verifiable and privacy-preserving fine-grained data management in vehicular fog computing: A game theory-based approach

Zahra Seyedi[1], Farhad Rahmati[1], Mohammad Ali[1*], Ximeng Liu[2,3]

[1]Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran.

[2]College of Computer and Data Science, Fuzhou University, Fuzhou, 350116, Fujian Province, China.

[3] Key Laboratory of Information Security of Network Systems, Fuzhou, 350116, Fujian Province, China.

*Corresponding author(s). E-mail(s): mali71@aut.ac.ir;
Contributing authors: zahraseyedi@aut.ac.ir; frahmati@aut.ac.ir; snbnix@gmail.com;

Abstract

Vehicular fog computing (VFC) is an effective technology in providing end-users and vehicles with ultra-low-latency services by extending fog computing to regular vehicular networks. It enables connected Vehicular Fog Nodes (VFNs) to process real-time data and promptly respond to users' queries. However, touching unencrypted data by VFNs raises security challenges definitely, the top of which is confidentiality, and giving encrypted data to VFNs causes other problems such as encrypted data processing. Apart from these, how to inspect and encourage VFNs to provide a secure, honest, and user-satisfactory network is of vital importance to this area. To address these challenges, we design a novel fine-grained data management (FGDM) approach for VFC-assisted systems. Our FGDM provides control over both retrieval and access to outsourced data in fine-grained ways. Also, it offers highly efficient approaches for the accuracy verification of operations performed by VFNs. In designing the system, we consider a three-player game among system entities to capture their interactions. We formulate the management problems as a Nash Equilibrium (NE) problem and show the existence of an equilibrium. Our security analysis and empirical results demonstrate that the FGDM is secure in the standard model and acceptably efficient.

Keywords: Vehicular fog computing, encrypted data processing, data retrieval, data management, Nash equilibrium.

1 Introduction

With the increasing growth of the Internet of Things (IoT) and smart cities, a new paradigm for vehicular networks called the Internet of Vehicles (IoV), has emerged. IoV provides vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-pedestrian (V2P) communications. This technology enables drivers and passengers to

access real-time information through their vehicles [1, 2]. The global number of connected vehicles is expected to double in the coming years, with 192 million currently connected vehicles, and a predicted reach of 367 million by 2027 (a 91% increase)[1]. IoV opens up many opportunities and offers various services such as weather programs, transportation and travel information, and other programs that were previously possible through smartphones. Using IoV-based systems can help to improving traffic management and preventing road hazards (currently, according to the World Health Organization (WHO)[2], 1.35 million people die in road accidents every year) [3, 4].

The implementation of IoV applications, however, is faced with several challenges involving data exchange, storage, and processing. Due to the significant latency limitations associated with these emerging applications, it is not feasible to rely on transmitting data from vehicles to a remote server for storage or computation. By moving servers closer to vehicles, fog computing has been proposed as a promising solution to overcome such latency limitations. However, since vehicles are on-the-move users, connecting to fixed fog servers may result in similar latency issues. Furthermore, traversing an area with a limited network infrastructure may disrupt fog computing support for vehicles. Consequently, some IoV applications may stop and decrease the performance of vehicle systems. As a solution to this problem, vehicular fog computing(VFC) that uses vehicles as mobile fog nodes to meet dynamic user needs can be an alternative to support fog computing for vehicular network-based applications [5, 6].

In VFC, resource-constrained vehicles can use the idle resources of parked or moving vehicles to outsource their data and tasks [6]. These vehicles as fog nodes are called vehicular fog nodes (VFNs), which provide computing and data caching services to nearby vehicles [5]. Generally, to outsource and retrieve data in a VFC system, according to Fig. 1, data owners (DOs) that intend to share data, such as traffic data, outsource them to VFNs. Data users (DUs) send requests to nearby VFNs, and VFNs provide them with these data [5, 6]. Despite all the advantages of VFC for the IoV, outsourcing unencrypted data to VFNs and the data retrieval process creates challenges for entities, some of which we will explain in the following:

- **Challenge 1**. Participation of VFNs: One of the basic requirements for creating a VFC system is the presence of VFNs in the network. Each VFN is a combination of vehicles and humans functioning as a single unit. Encouraging their owners to participate in the system is crucial to use the vehicles' resources. Thus, how to motivate VFNs to participate in the system poses a significant problem. Moreover, ensuring reliable and honest user services from VFNs is an important challenge that requires inspection and appropriate motivation.
- **Challenge 2**. Secure data retrieval process: To protect sensitive data stored in VFNs, Searchable encryption schemes are usually utilized to enable DUs to provide fine-grained access control over encrypted data and retrieve their data without revealing sensitive information or risking unauthorized access. However, VFNs may vary in some attributes such as accuracy of search results, computing power, and distance from the DU, so DUs have to set access policies to ensure only compliant VFNs perform search operations. To the best of our knowledge, currently, there is no specific searchable encryption technique that takes into account the attributes of VFNs, presenting a limitation in vehicular fog computing networks.
- **Challenge 3**. Selection of the proper VFNs: Selecting a suitable VFN is crucial for complying with the access control policy of a DU. However, not all VFNs in proximity might meet the policy requirements, leading to two approaches: The DU must either be able to find the appropriate VFN or forward the request to all nearby VFNs. The former solution demands that the DU acquires and maintains knowledge of all VFN attributes, thereby putting their privacy at risk. The latter can be time-consuming and exhausting for the VFNs involved, as they may receive excessive requests that do not align with their attributes, forcing them to scrutinize each request.
- **Challenge 4**. Search result verification: Once the VFN has performed a search operation

[1] https://www.juniperresearch.com/researchstore/operators-providers/connected-vehicles-research-report

[2] https://www.who.int/publications/i/item/9789241565684

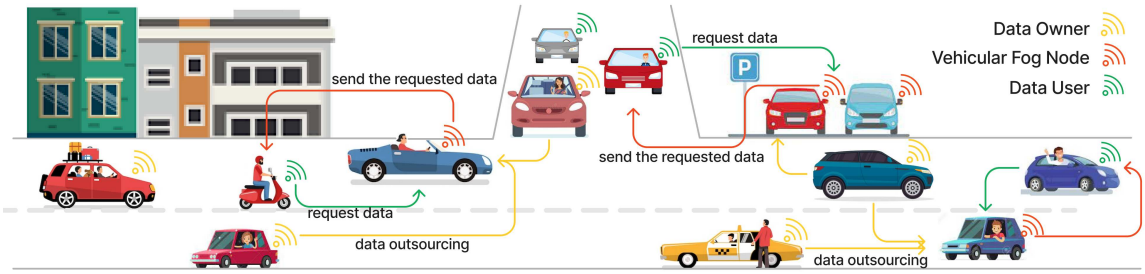


Fig. 1 A VFC-based system.

on behalf of the DU, the DU needs to verify the accuracy of the results. Several schemes have been proposed to address verification challenges; however, new problems will arise if the results are found to be incorrect. This may lead to wasted time for the DUs by receiving the wrong results and make them unsure if future results will be accurate. In some cases, this may even prompt the DU to leave the system altogether.

- **Challenge 5**. Attribute revocation: Depending on the search result verification output, the attributes of the VFN may change. So, whenever a VFN's attributes change, the secret key corresponding to its previous attributes should no longer be valid. Therefore, it is important to update VFN's attributes whenever necessary.

To alleviate the described challenges, we design a fine-grained data management (FGDM) scheme and describe how to implement it in a VFC environment. However, we believe that using only cryptographic approaches is insufficient for creating motivations and system management. The unique features of VFC environments, such as the independence of entities, dynamic and extensive limited knowledge of the environment, relationships between entities, and lack of control over entities' treatment, necessitate new security methods. Game theoretic approaches provide a promising solution to consider the benefits and interests of entities involved in the system and ensure their loyalty to the system. Utilizing this approach, we design an incentive mechanism that captures strategic interactions between the entities in a three-player game. The players' strategies are highly dependent on the actions of other players. The main contributions are summarized as follows:

- Strategic game: To tackle Challenges 1 and 4, we propose a game-based keyword search outsourcing process (KWSOP) approach involving three parties: the CA, a DU, and a VFN. We demonstrate that this game has a Nash Equilibrium (NE), which yields an optimal strategy for each player. In particular, we prove that the VFN role player should aim to perform the search operation accurately and has no incentive to deviate from this strategy. By analyzing the players' utility functions, we establish that not only are VFNs likely to participate in the KWSOP, but they also have a strong motivation to deliver high-quality services.
- Verifiable attribute-based keyword search (VABKS): We have designed a new VABKS cryptosystem with two access control policies enforcement to address Challenges 2 and 4. Apart from enforcing the access control policy of the DOs to utilize outsourced encrypted data, this cryptosystem also permits DUs to regulate the outsourced search operations using their control policies. Moreover, this system enables them to verify if the VFN has conducted the search operation honesty.
- Attribute-based VFN selection mechanism: In order to address Challenge 3 and prevent the leakage of VFNs' attributes, we develop an attribute-based VFN finding mechanism. Our FGDM scheme utilizes an algorithm that generates a list of VFNs according to the policies of the DU. This way, once they receive the list of VFNs, DUs will not know anything beyond that these VFNs are in accordance with their policies.
- Revocable key delegation mechanism: To address Challenge 5, we design a process for updating the attributes of VFNs. When the CA identifies that an attribute of a VFN has been changed, certain algorithms are initiated

to update the VFN's attributes and generate any necessary parameters.
- Efficient search result verification: Our FGDM offers an efficient method for verifying search results, thereby addressing Challenge 4. Drawing on the scheme proposed by Ali et al. in [7], our FGDM scheme enables a DU to validate search results received from a VFN without the need to download the entire data.

Paper Organization: In Section 2 we review the related work. Section 3 provides some preliminary information. Then, we describe our problem formulation in Section 4. Section 5 presents our FGDM scheme in IoV environment. In Section 6, we define our KWSOP game and find its NE. In Section 7, we provide a detailed presentation of our proposed FGDM. Sections 8 describes security analysis and Section 9 presents performance analysis. Concluding remarks can be found in Section 10. Additionally, we include the correctness analysis in the Appendix to ensure easy comprehension and completeness of this paper.

2 Related Work

This section aims to review two primary areas of research. Firstly, we will discuss related work regarding VFC and assess the strengths of our approach in comparison to other schemes outlined in Table 1. Secondly, we will explore related work on attribute-based cryptographic systems and examine various existing schemes in this field, making comparisons with our FGDM scheme in Table 2.

2.1 Vehicular fog computing

VFC is a derivative of fog computing that uses vehicles as fog nodes to overcome limitations such as latency. Existing work in this field has investigated VFC from different perspectives, such as architecture, latency optimization, and utility maximization [22, 23]. Most studies of VFC have focused on its architecture [23], which can be divided into two general categories: infrastructure-based VFC and vehicle-based VFC. The former category considers infrastructure near vehicles [24], and the latter considers vehicles with idle resources [9] as fog nodes. In contrast to an infrastructure-based VFC system, that necessitates extra equipment such as roadside units (RSUs), a vehicle-based system is simpler to implement [12]. For instance, in their proposed architecture, Zhu et al. [25], have used commercial fleets moving along specific routes as fog nodes. However, it is noteworthy that, in contrast to the focus on the architecture of this prominent emerging concept [23], there are few studies on users' and vehicles' utility maximization.

Papers [8, 10, 11, 22, 26] are some of studies that have investigated the entities' interests and maximization of their utilities. To deal with time-sensitive and computationally intensive tasks of vehicles, the authors in [22] present a resource allocation algorithm. In [10], the authors designed a task scheduling method using the ant colony optimization method. Shojafar et al. [11] proposed an adaptive resource scheduling for fog nodes that maximize system performance. In the scenario proposed in [26], vehicles try to take advantage of their own resources by sharing their idle computing resources. In this work, the RSU is responsible for providing task scheduling. In addition, edge and cloud server providers can also be jointly used to provide computing services to requesters such as their nearby vehicles. In [8], users' requests are served first in RSUs, and if the quality of service (QoS) in the RSU is no longer eligible, for example, a large number of requests causes a shortage of RSU resources, these requests are sent to the cloud center. The authors in [13], to deal with the limitation of the computing power of moving vehicles, have proposed to use the computing resources of parked vehicles in VFC. [27] proposed the concept of pooling resources in VFC. In this scheme, by combining the computing resources of vehicles, computing services are jointly provided in a community. Also, a strategy based on genetic algorithm is presented in [27] to solve the utility maximization problem. Other similar studies on resource allocation are also presented in [14, 15]. However, all these work try to maximize the benefits of VFNs in some way only through resource allocation. Underestimating the participants' benefits in this system, one of our main challenges in this paper, makes its development problematic in the real world.

Table 1 Functionalities of VFC system comparison.

	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	ours
$\mathcal{F}_1$					✓		✓		✓
$\mathcal{F}_2$	✓	✓		✓		✓	✓	✓	✓
$\mathcal{F}_3$			✓		✓				✓
$\mathcal{F}_4$	✓	✓				✓	✓		✓
$\mathcal{F}_5$			✓						✓
$\mathcal{F}_6$	✓	✓	✓	✓		✓	✓	✓	✓
$\mathcal{F}_7$					✓				✓
$\mathcal{F}_8$									✓
$\mathcal{F}_9$									✓

Notes. $\mathcal{F}_1$: Considering attributes of fog nodes, $\mathcal{F}_2$: Quick access to VFNs, $\mathcal{F}_3$: Optimizing VFNs' benefits, $\mathcal{F}_4$: Latency Optimization, $\mathcal{F}_5$: Finding available resources nearby, $\mathcal{F}_6$: Moving fog nodes, $\mathcal{F}_7$: Optimal employment of fog nodes, $\mathcal{F}_8$: Update fog nodes attributes, $\mathcal{F}_9$: Finding the equilibrium point of the system.

Table 2 Functionalities of ABKS schemes comparison.

	[16]	[17]	[18]	[19]	[20]	[7]	[21]	ours
$\mathcal{F}_1$	✓	✓	✓	✓		✓	✓	✓
$\mathcal{F}_2$			✓		✓	✓	✓	✓
$\mathcal{F}_3$	✓				✓	✓		✓
$\mathcal{F}_4$						✓		✓
$\mathcal{F}_5$								✓
$\mathcal{F}_6$								✓
$\mathcal{F}_7$						✓	✓	✓

Notes. $\mathcal{F}_1$: Verifiable keyword search, $\mathcal{F}_2$: Multi keyword search, $\mathcal{F}_3$: Lightweightness, $\mathcal{F}_4$: Search results verification without the main retrieved files, $\mathcal{F}_5$: Considering edge nodes attributes, $\mathcal{F}_6$: Updating and revocing VFNs' attribute, $\mathcal{F}_7$: Simultaneous access and search control.

2.2 Attribute-based cryptographic systems

There are two main types of attribute-based approaches that provide fine-grained access and keyword search control over encrypted data: attribute-based encryption (ABE) and attribute-based keyword search (ABKS) methods.

Attribute-based encryption. A popular cryptographic tool for enforcing access control policies is ABE, which was first proposed by Sahai and Waters in [28]. In an ABE scheme, the users' access rights be controlled by considering access policies in ciphertexts or users' secret keys. Depending on how access policies are associated with ciphertexts and secret keys, these schemes can be divided into two categories: ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In CP-ABE, the ciphertext is associated with the access control policy [29], and in KP-ABE, the decryption key is associated with the access control policy [30]. A wide range of ABE schemes has been proposed to enrich the functionality of this technique, for instance, anonymity [31], multi-authorization [32], user revocation [33], policy update [34], and decryption/encryption outsourcing [35]. One significant problem in ABE schemes is preserving users' privacy. To alleviate this challenge, the authors in [31] propose a model that preserves users' anonymity. However, as the number of users in a system increases, scalability issues may arise in communication networks. To this end, in the proposed approach described in [32], the central authority can create new domain authorities as needed for additional computational resources. Besides these functions, if a user loses an attribute or a data owner's policies has changed, many security and privacy problems will arise. Therefore some ABE schemes with attribute revocation and policy update functions were proposed [33, 34].

Attribute-based keyword search. In order to investigate the notion of ABE in keyword search schemes, Zheng et al. proposed the attribute-based keyword search scheme [19]. In this scheme, keywords are encrypted based on an access control policy, and authenticated DUs can search the outsourced encrypted data using generated tokens according to their attributes and the keywords they want to search for. This method has been enhanced with many features that provide different functionalities. In [36], an efficient user-revocable ABKS scheme is proposed. Mayo et al. [37] designed an ABKS scheme in the shared multi-owner settings that supports hidden access policy. The ability to search for multiple keywords is a feature that is required for more practical encryption schemes, although none of the mentioned ABKS schemes mentioned provide this requirement. Several multi-keyword search approaches [12]-[26], [29] have been presented to address the mentioned shortcoming. However, these multi-keyword search schemes, only return the desired ciphertext if the requested keywords match all keywords associated with the data. To improve the data retrieval process, users need the ability to set a threshold for the quantity of matching keywords. The scheme presented in

[21] enables data users to set a limit for the quantity of matching keywords in the generation of search token. Despite all these functionalities, these schemes are still imperfect, and they have ignored the untrustworthiness of the server.

Servers are not trusted entities, and we cannot be sure they will perform the search operation accurately. Therefore, it is very important to provide a verification mechanism to check the validity of the search results. Various solutions have been proposed, such as those discussed in [36], however, some come with drawbacks such as Bloom filters causing a high false positive rate leading to communication overhead. Alternatively, Miao et al. [38] offer a more efficient approach that allows for relatively less complicated verification operations. However, to the best of our knowledge, it is impossible to verify the validity of search results without all retrieved data in aforementioned schemes. Therefore, verifiers typically need to download the retrieved data to perform the verification process, which leads to wasted time and unnecessary communication overhead due to downloading unusable or incorrect data. To address this issue, one of the main contributions of [7] is the ability to authenticate the results of the search operation without requiring full downloads of the retrieved files.

Despite all these features, none of the existing research studies has seriously addressed the issue of system management. For example, in the topic of computation outsourcing, while our system is distributed, e.g. fog computing or edge computing, there are several options, fog servers, for outsourcing the computation. In this case, the existing schemes have not provided any solution or analysis to choose the best option for outsourcing, which is one of our main challenges in this paper.

3 Preliminaries

3.1 Bilinear map

Suppose two cyclic groups G_1 and G_2 of a prime order p. A map $\sigma : G_1 \times G_1 \to G_2$ is be bilinear if it has these three features:

- Bilinearity: We have $\sigma(\alpha^x, \alpha^y) = \sigma(\alpha^x, \alpha^y) = \sigma(\alpha, \alpha)^{xy}$ for each $\alpha \in G_1$ and $x, y \in \mathbb{Z}_p$;
- Non-degeneracy: There exists an element $\alpha \in G_1$ such that $\sigma(\alpha^x, \alpha^y) \neq 1$;
- Computability: There exists an efficient algorithm that computes $\sigma(\alpha, \beta)$ for each $\alpha, \beta \in G_1$.

3.2 Access structure

Let $\mathcal{E} = \{\mathcal{E}_1, ..., \mathcal{E}_n\}$ be a set of entities. An access structure on $\mathcal{E}$ is a non-empty subset $\mathcal{A}$ of $2^{\mathcal{E}}$. If for every $A \in \mathcal{A}$ and $\mathbb{E} \subset \mathcal{E}$ such that $A \subset \mathbb{E}$, we have $\mathbb{E} \in \mathcal{A}$, then we say that $\mathcal{A}$ is monotone. Every set $A \in \mathcal{A}$ is known as an authorized set, while all the other sets in $2^P \setminus \mathbb{A}$ are referred to as unauthorized sets. A set $\mathbb{E} \subset \mathcal{E}$ is said to satisfy an access structure $\mathcal{A}$ if $\mathbb{E} \in \mathcal{A}$.

In ABE, attributes replace entities. We can represent any monotone access structure by an access tree, where each unique attribute is represented by a leaf node [19]. The **Share** and **Combine** algorithms are two practical algorithms that we utilized in our scheme to this end, and were introduced in [39]:

- **Share**$(p, r, \mathcal{T})$: This algorithm is probabilistic polynomial-time (PPT) and that requires three inputs: a prime number p, an access tree $\mathcal{T}$, a secret value $r \in \mathbb{Z}_p$. This algorithm generates a distribution $\{D_i\}_{i \in L_{\mathcal{T}}}$ of r according to the access tree $\mathcal{T}$, where $L_{\mathcal{T}}$ represents the set of leaf nodes in $\mathcal{T}$.
- **Combine**$(\{\sigma(\alpha_1, \alpha_2)^{D_i}\}_{i \in S}, \mathcal{T})$: Let $\mathcal{T}$ be an access tree, and Att is the set of attributes associated with $L_{\mathcal{T}}$. This algorithm, given a bilinear group parameters (p, G_1, G_2, σ), takes a set of values $\{\sigma(\alpha_1, \alpha_2)^{D_i}\}_{i \in S}$ and the access tree $\mathcal{T}$, where $\alpha_1, \alpha_2 \in G_1$, $S \subseteq Att$, and for a secret value $r \in \mathbb{Z}_p$, $\{D_i\}_{i \in L_{\mathcal{T}}}$ represents the output of **Share**$(p, r, \mathcal{T})$. If the set S satisfies $\mathcal{T}$, then the output of the algorithm is $\sigma(\alpha_1, \alpha_2)^r$. If not, an error message $\perp$ is returned as output. For further information about the **Share** and **Combine** algorithms and the concept of access tree, please refer to the source [39].

3.3 DBDH assumption

Assume a PPT algorithm $\mathcal{G}$ which takes as input a security parameter n, and produces a tuple (p, G_1, G_2, σ) as output, where p, G_1, G_2, and σ are the same as Section 3.1. The decisional bilinear Diffie–Hellman (DBDH) assumption is true for $\mathcal{G}$ if, for any tuple $(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^w)$, the

advantage of any PPT distinguisher $\mathcal{D}$ in distinguishing the case where w equals xyz from the case where w is a uniform element of $\mathbb{Z}_p$ is a negligible function in n. Here n is a security parameter, (p, G_1, G_2, σ) is an output of $\mathcal{G}(1^n)$, $\alpha \in G_1$ and $x, y, z \in \mathbb{Z}_p$ are selected uniformly at random, and w is either a uniform element of $\mathbb{Z}_p$ or equals to xyz. In other words, for each PPT distinguisher $\mathcal{D}$, a negligible function $negl$ exists such that:

$$|Pr(\mathcal{D}(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^{xyz}) = 1) - Pr(\mathcal{D}(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^w) = 1)| \leqslant negl(n),$$

where the probabilities are calculated based on the random selection of $x, y, z, w \in \mathbb{Z}_p$, as well as the randomness utilized in $\mathcal{D}$ and $\mathcal{G}$.

3.4 Commitment scheme

A commitment scheme is composed of these three algorithms: $\mathbf{KeyGen}(1^n)$, $\mathbf{Commit}(pk, m)$, and $\mathbf{Open}(c, d)$. $\mathbf{KeyGen}(1^n)$ is a PPT algorithm that generates the necessary public parameters pk and defines the message space M. $\mathbf{Commit}(pk, m)$ is a PPT algorithm that uses the public parameters pk, a message m from the message space M to produce a value c that represents the commitment to m and d as the opening value. $\mathbf{Open}(c, d)$ is a deterministic polynomial-time algorithm that uses the public parameters pk, opening value d, and commitment value c to output a boolean value $b \in \{0, 1\}$ indicating whether c is a valid commitment to m.

Security properties of a commitment scheme include (1) correctness, i.e. the generated commitment for every message is valid. (2) perfect or computational hiding which guarantees that an attacker cannot get information with any or negligible advantage about m from c. (3) perfect or computational binding where m is exclusively linked to c or discovering another message with the same commitment has a negligible probability of success.

3.5 Strategic Game and Nash Equilibrium

A *strategic game* is a tuple $(P, \{S_i\}_{i \in P}, \{f_i\}_{i \in P})$ in which:

- P is a finite set (the set of players);
- For each player $i \in P$, S_i is a nonempty set (the set of actions available to player i);
- For each player $i \in P$, f_i is a payoff function on $S = \prod_{i \in P} S_i$.

We use the notation $s_i \in S_i$ for a strategy of player i. A collection of players' strategies is given by $s = \{s_i\}_{i \in P}$ and is referred to as a strategy profile. A collection of strategies for all players but the i-th one is denoted by $s_{-i} = (s_1, s_2, ..., s_{i-1}, s_{i+1}, ..., s_p) \in S_{-i}$. A (pure) *Nash equilibrium* is a strategy profile S^* from which no player can unilaterally deviate and improve its payoff. Formally, the strategy profile $S^* = (s_i^*, s_{-i}^*)$ is a Nash Equilibrium if $f_i(s_i^*, s_{-i}^*) \geqslant f_i(s_i^*, s_{-i})$ for every $s_i \in S_i$ and $i \in P$.

4 Problem formulation

To aid our FGDM scheme comprehension and adherence, in this section, we present its system model, threat model, and security requirements, respectively.

4.1 System Model

This work focuses on a VFC-based IoV scenario. The system involves five entities, as illustrated in Fig. 2: A Central Authority (CA), a Cloud Server (CS), a number of Data Owners (DOs), Data Users (DUs), and Vehicular Fog Nodes (VFNs). The entities have distinct roles and relationships, which are detailed below.

- **Central Authority**: The CA is responsible of initializing the system parameters and generating its own master secret key. Besides, it should generate required keys and update the attributes of VFNs.
- **Data Owners**: Each DO defines its policies for data access control and outsources the encrypted data under the defined access policy along with the assigned keywords to its surrounding VFNs.
- **Data Users**: Each DU looks for data with specific keywords. It is responsible for generating searchable tokens, defining the token access policy, encrypting tokens under this policy, and verifying the search result.
- **Vehicular Fog Nodes**: VFNs are individuals that have devices with moderate computing power and are responsible for caching data or forwarding them to the CS as edge servers. They

retrieve the requested data from its database or the CS.

- **Cloud Server**: The CS is a remote server with a large storage capacity that stores and shares encrypted data with VFNs through a public channel.

4.2 Threat Model and Security Requirements

The CA and DOs, in this work, are considered to be completely trustworthy. Once the CA initiates the system and generates the required parameters, it securely provides the corresponding keys to each entity. The CS and VFNs, either intentionally or accidentally, may fail to accurately perform the outsourced operations, so they are unreliable in this respect. Also, by collaborating with unauthorized DUs, they might attempt to gain access to unauthorized information regarding the outsourced data and related keywords. The DUs are malicious and can carry out any attacks. An authorized DU that can decrypt a ciphertext would not reveal any details about the content of the data and linked keywords in any way. Furthermore, unauthorized DUs may collaborate with each other and the CS to obtain unauthorized access to the outsourced data. We summarize the security requirements of our design in the following two cases:

- *Indistinguishability*: Only authorized DUs or VFNs whose attributes meet the required access policies are allowed access to the ciphertext or tokens. And the encrypted files must not reveal any information about their data, keywords, or tokens.
- *Unforgeability*: In the verification phase, if a VFN fails to correctly execute the necessary operations required in the search algorithm, then the search results returned to DUs must not be validated.

5 Proposed FGDM in IoV environment

This section utilizes our FGDM as the fundamental component in designing a secure VFC-based IoV system. The notations employed in this section and Section 7 are listed in Table 3.

As illustrated in Fig. 2, the proposed system comprises of nine distinct phases. In Phases 1 and 2 the CA initializes the system parameters and produces secret-keys for all entities, respectively. During Phase 3, a DO, choosing a specific access tree and a set of keywords, encrypts its data using that access tree and outsources the encrypted data along with the encrypted keywords to near VFNs. In phase 4, a DU defines a set of policies, and corresponding to each of these policies receives a list of VFNs from the CA. In phase 5, a DU considers a set of keywords and creates a token that matches these keywords. Then, the DU encrypts the token under a chosen policy and sends a search request to VFNs corresponding to this policy. In phase 6, after receiving data queries from a DU, one of the VFNs receiving the request accepts it and decrypts the requested token. It can decrypt the token only if its attributes satisfy the DU's access tree. Then it searches the local storage, interacts with other VFNs, or requests the CS to find all ciphertexts that meet these criteria: (1) Their access trees align with the DU's attributes; (2) They contain $\mathcal{R}$ or more matching keywords from the DU's selected keywords set. Afterward, it sends the search result to the DU. Since a DU does not fully trust VFNs, in phase 7, it can verify the accuracy of operations performed in the search process before downloading the encrypted ciphertexts. If the search result is not verified, the DU will inform the CA about that specific VFN. Otherwise, it downloads the ciphertexts. Then, in phase 8, if its attribute set matches the access policy in the ciphertext, it retrieves the data associated with ciphertexts by performing very efficient operations. In phase 9, when a VFN's attribute has changed, its access right will be updated. To this end, at first, the CA changes the parameters associated with the attribute and produces its corresponding update-key. Then, according to the new parameters, the CA updates VFN's secret-keys.

5.1 Definition of proposed FGDM scheme

This scheme is consists of the following 14 algorithms. The detailed construction of all these algorithms is presented in Section 7.

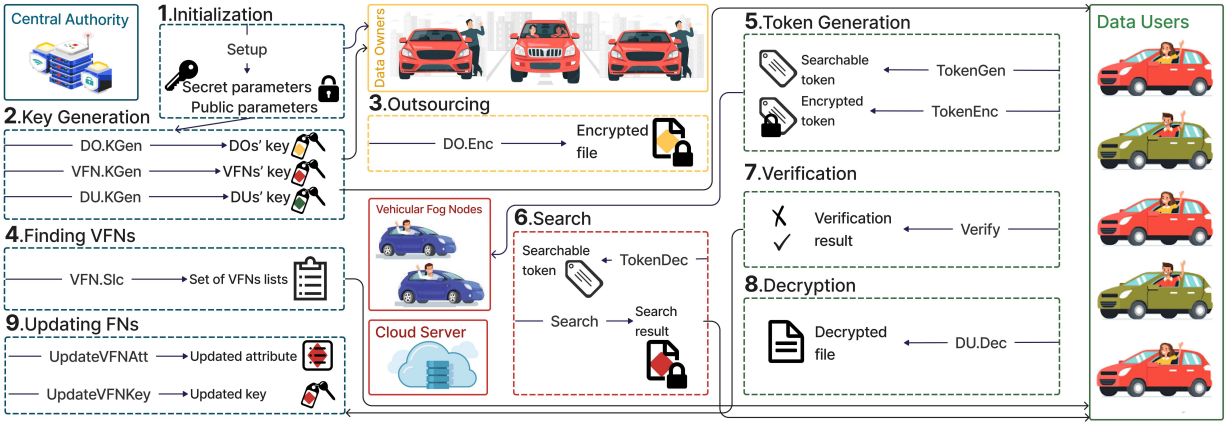


Fig. 2 System model.

Table 3 Notations.

Notation	Description
n	System's security parameter
$\mathbb{U}$	Universal attribute set
PK	System's Public parameters
MSK	System's master secret-key
ID_{vfn}	Identifier of a vehicular fog node
ID_{du}	Identifier of a DU
Att_{du}	Attribute set of a DU
SK_{du}	Secret-key of a DU
ω_j	jth keyword assigned to a file
$\hat{\omega}_j$	jth keyword in a search query
M	A message
SK_{do}	A DO's secret-key
PK_{do}	A DO's public-key
$\mathcal{T}$	An access tree
$SCT_{\mathcal{T}}$	A searchable ciphertext
TK_{du}	Search token generated by a DU
CT_{Token}	Search token ciphertext produces by a DU
$\mathcal{R}$	A threshold value
Π_{sign}	A signature scheme
Π_{com}	A commitment scheme
Π_{enc}	A symmetric encryption scheme
Π_{mac}	A message authentication code scheme

1. **Setup**$(1^n, \mathbb{U}, \mathbb{U}')$: The algorithm takes the security parameter n as input, and generates the global public parameters PK and the master secret-key MSK.
2. **DO.KeyGen**(MSK, PK, ID_{do}): It takes as input the master secret-key MSK, and an identifier of a data owner ID_{do}. It outputs (PK_{do}, SK_{do}).
3. **VFN.KeyGen**(MSK, PK, ID_{vfn}, a): The master secret-key MSK, the VFN's identifier ID_{vfn}, and an attribute a are the inputs of this algorithm. It outputs a secret-key $SK_{a,vfn}$.
4. **DU.KeyGen**$(MSK, PK, ID_{du}, Att_{du})$: It takes the master secret-key MSK, a data user's identifier ID_{du}, and its attribute set Att_{du} as inputs. Then, the algorithm returns a secret-key SK_{du}.
5. **DO.Enc**$(PK, PK_{do}, SK_{do}, ID_{do}, M, \{\omega_j\}_{j=1}^m, \mathcal{T})$: A DU's public-key PK_{do}, secret-key SK_{do}, and identifier ID_{do}, a message M, a keyword set $\{\omega_j\}_{j=1}^m$, and an access tree $\mathcal{T}$ are the inputs of this algorithm. It returns a searchable ciphertext $SCT_{\mathcal{T}}$.
6. **VFN.Slc**$(\{\mathcal{T}_{p_i}\}_{p_i \in P_{du}}, \{Att_{v_i}\}_{v_i \in V_{du}})$: It takes a DU's set of access trees, $\{\mathcal{T}_{p_i}\}_{p_i \in P_{du}}$, and its set of near VFNs' attributes, $\{Att_{v_i}\}_{v_i \in V_{du}}$, as inputs. It returns a set of VFNs corresponding to each policy, $\{L_i\}_{i \in P_{du}}$.
7. **TokenGen**$(PK, ID_{du}, SK_{du}, \{\hat{\omega}_j\}_{j=1}^l)$: The DU's identifier, its secret-key, and a keyword set $\{\hat{\omega}_j\}_{j=1}^l$ are the inputs of the algorithm. Its output is a token TK_{du} associated with the keyword set.
8. **TokenEnc**$(PK, TK_{du}, \mathcal{T}_{du})$: It takes keywords token TK_{du}, and an access policy tree $\mathcal{T}_{du}$ as inputs. It returns a search token ciphertext CT_{Token} according to the DU's access tree.
9. **TokenDec**$(PK, SK_{a,vfn}, CT_{Token})$: On input a VFN's secret-key $SK_{a,vfn}$ and the ciphertext of a the DU's keywords token CT_{Token}, this algorithm returns the token corresponding to this ciphertext.
10. **Search**$(PK, TK_{du}, ID_{du}, SCT_{\mathcal{T}}, \mathcal{R})$: It takes a searchable keywords token TK_{du}, a DU's identifier ID_{du}, a searchable ciphertext $SCT_{\mathcal{T}}$, and

a threshold value $\mathcal{R}$ as inputs. The algorithm outputs a response Re.

11. **Verify**$(PK, Re, ID_{do}, ID_{vfn}, \mathcal{K})$: On input Re, ID_{do}, and ID_{vfn}, this algorithm verifies the search result Re's validity. If it is valid, the algorithm returns 1. Otherwise, 0 is output which means the search result is not valid.
12. **DU.Dec**$(PK, \mathcal{K}, \mathcal{TK}, C_{do}, t_{do})$: Given a ciphertext C_{do}, parameters $\mathcal{K}$ and $\mathcal{TK}$ from search result Re, and the DU's token t_{do}, this algorithm first checks whether this ciphertext matches with the DU's token or not. If not, it aborts and outputs $\perp$. Otherwise, it recovers the message corresponding to C_{do}.
13. **VFN.AttUpdate**(PK, a_0, MSK): This algorithm takes an attribute a_0, and the master secret-key MSK. It updates the system parameters associated with the attribute and produces an update-key UK_{a_0} .
14. **VFN.KeyUpdate**$(PK, a_0, UK_{a_0}, SK_{a_0,vfn}, ID_{vfn})$: It takes an attribute a_0, an update-key UK_{a_0}, a secret-key of a VFN associated with the attribute, $SK_{a_0,vfn}$, and its identifier, ID_{vfn}. It outputs an updated secret-key $SK'_{a_0,vfn}$.

5.2 System overview

In this section, we will explain how to implement our FGDM scheme on the proposed IoV system in nine phases:

- **System Initialization**. At first, the CA chooses a security parameter n and two attribute sets $\mathbb{U}$ and $\mathbb{U}'$. Next, it executes **Setup**$(1^n, \mathbb{U}, \mathbb{U}')$ algorithm to produce public parameters PK and the master secret-key MSK. The PK is shared with other entities while keeping MSK private.
- **Key Generation**. The CA carries out this phase to create the required keys for each entity. Upon receiving the MSK, it executes the algorithm **DO.KeyGen**(MSK, PK, ID_{do}) to generate secret and public keys for each DO based on its identity. Then, it uses the **DU.KeyGen**$(MSK, PK, ID_{du}, Att_{du})$ algorithm to compute secret keys for each DU with an identifier ID_{du} and attribute set Att_{du}. Similarly, it employs the **VFN.KeyGen**(MSK, PK, ID_{vfn}, a) algorithm to generate a secret key for each VFN corresponding to its identity and attributes.
- **Encryption**. Before outsourcing a message M, a DO first evaluates a set of keywords $\{\omega_j\}_{j=1}^m$. Then, using its public-key PK_{do}, secret-key SK_{do}, and identifier ID_{do}, it runs **DO.Enc**$(PK, PK_{do}, SK_{do}, ID_{do}, M, \{\omega_j\}_{j=1}^m, \mathcal{T})$ algorithm to produce a searchable ciphertext SCT_{do}. Subsequently, it sets up an access tree $\mathcal{T}$ specifying which DUs have permission to access the message. Finally, it sends the encrypted data to VFNs around it.
- **VFN Selection**. In this phase, a DU submits a set of policies to the CA. The CA then utilizes the **VFN.Slc**$(\{P_i\}_{i=1}^m, \{vfn_j\}_{j\in VFNs})$ algorithm to generate, for each policy, a list of VFNs that are accessible to the DU whose attributes meet the policy requirements. The CA subsequently shares these lists with the DU.
- **Token Generation**. A DU, using its secret key SK_{du} and identifier ID_{du}, executes the **TokenGen**$(PK, ID_{du}, SK_{du}, \{\hat{\omega}_j\}_{j=1}^l)$ algorithm to generate a token that corresponds to its selected keywords. To ensure that only authorized VFNs can perform search operations, the DU encrypts the resulting token according to its access control policies $\mathcal{T}_{du}$ using the **TokenEnc**$(PK, TK_{du}, \mathcal{T}_{du})$ algorithm.
- **Search**. To perform the search operation, a VFN must have access to searchable tokens. To this end, it first decrypts the encrypted tokens CT_{Token} using its secret-key $SK_{a,vfn}$, and the execution of **TokenDec**$(PK, SK_{a,vfn}, CT_{Token})$ algorithm. The VFN then runs the **Search**$(PK, TK_{du}, ID_{du}, SCT_{\mathcal{T}}, \mathcal{R})$ algorithm with the searchable token TK_{du}, to find the DU's desired files and sends a response Re, back to the DU.
- **Verification**. To ensure the validity of the search result, the DU carries out a verification phase. Before downloading the response Re from the VFN, the DU utilizes the **Verify**$(PK, Re, ID_{do}, ID_{vfn}, \mathcal{K})$ algorithm to confirm the accuracy of the search operation. If the verification fails, the DU reports the identity of the VFN responsible for the search, ID_{vfn}, to the CA. Otherwise, it proceeds to download Re.

- **Decryption**. In this phase, the DU runs the **DU.Dec**$(PK, \mathcal{K}, \mathcal{TK}, C_{do}, t_{do})$ algorithm to retrieve the original file.
- **Updating VFNs**. According to the reports of DUs from search results or in different conditions, the attributes of VFNs may change. Therefore, upon receiving these reports, the CA first updates the attributes of VFNs by running the **VFN.AttUpdate**(PK, MSK, a_0) algorithm. Then, it runs the **VFN.KeyUpdate**$(PK, a_0, UK_{a_0}, SK_{a_0,vfn}, ID_{vfn})$ algorithm to update their secret-keys.

In the next section, we will adopt a game-theoretic approach to address the issue of how to incentivize VFNs to participate in the system and ensure that they do not deviate from their tasks.

6 KWSOP Game

To manage the data retrieval process, especially the keyword search outsourcing process (KWSOP), in the previous section, we presented our FGDM scheme and explained how to implement it in a VFC environment. However, due to the reasons we stated in the introduction, this scheme is not adequate to overcome the mentioned challenges. To this end, in the following, we will investigate the KWSOP in the form of a strategic game:

6.1 Game overview

Three entities with different goals participate in the KWSOP: a DU, a VFN, and the CA. Each of these three players has its goal of being in the VFC system. The DU's primary objective is to outsource the keyword search operation to a VFN according to its policies and get the desired result. The VFN, on the other hand, intends to earn the highest wage in exchange for executing the operation. Meanwhile, the CA strives to maintain its clientele and not lose VFNs' participation in the network while paying the lowest wage possible. These players employ different strategies to achieve their goals. The DU can decide which VFNs to delegate the search operation to, according to its policies. Once the request is accepted by one of the VFNs, that VFN can decide whether to cancel the request or not. And if it does not cancel the request, whether to carry out the task accurately or not. Besides, the CA will have different strategies based on the reports received from the DU. If the DU reports any violation linked to the VFN, the CA should decide whether to update the attributes of the VFN or not. Or it should decide whether to pay the VFN in full or not. Therefore, we formulate the KWSOP as a sequential game:

Definition 1 (KWSOP Game)**.** *The sequential KWSOP game is the following triplet G, in which the DU is the first, the VFN is the second, and the CA is the last player (Fig. 3):*

$$G = (P,\ \{S_i\}_{i\in\{du,\,vfn,\,CA\}},\ \{f_i\}_{i\in\{du,\,vfn,\,CA\}}),$$

where

- $P = \{vfn \in \{VFNs\}, du \in \{DUs\}, CA\}$,
- $S_{vfn} = \{A : \textit{Accurate execution}, nA : \textit{not Accurate execution}, C : \textit{Cancel}\}$,
- $S_{du} = \{P_i \,|\, \forall\, i \in \{1, ..., m\},\ P_i \textit{ is an access policy}\}$,
- $S_{CA} = \{WP : \textit{Wage Payment},$
  $AU : \textit{Attributes Update},$
  $WP{+}AU : \textit{Wage Payment+Attributes Update}\}$.

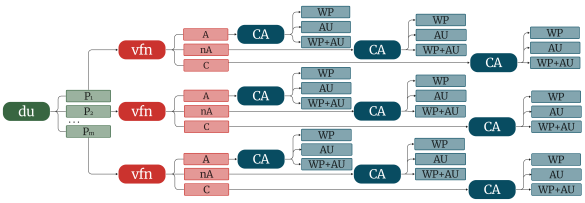


Fig. 3 Game tree.

For each VFN and each operation O, if the VFN accurately operates O, we denote its reward as R_O and its penalty in case of making a mistake as P_O. Its penalty in case of canceling the request is also shown with P_C. Then, the utility function of the VFN will be as follows:

- $f_{vfn} = \begin{cases} R_O & A \\ -P_O & nA \\ -P_C & C. \end{cases}$

For each user DU and each operation O, the payoff of the DU will be 0 or U_O if it requests from the VFN, depending on the result of VFN's operation, and will be U'_O if it does not request. Then, the utility function of DU will be as follows:

- $f_{du}(P_i) = cd_i(U_O)$ where, $c \in \{0, 1\}$.

If the CA chooses the WP strategy, it will lose the loyalty of DUs since the presence of VFNs with incorrect attributes in the system will cause DUs dissatisfaction. Therefore, the only benefit of this strategy is to keep the VFN in the system, which we represent with U_{vfn}. In the same way, by choosing the AU strategy, the only benefit will be to keep DUs, U_{du}. Then, the utility function of the CA will be as follows:

- $f_{CA} = \begin{cases} U_{vfn} & WP \\ U_{du} & AU \\ U_{vfn} + U_{du} & WP + AU. \end{cases}$

We now introduce the concept of Nash equilibrium,

Definition 2. *A strategy profile $s^* = (s^*_{du}, s^*_{vfn}, s^*_{CA})$ is a Nash equilibrium of the KWSOP game if at the equilibrium s^*, no player can further upgrade its utility by unilaterally changing its strategy, i.e.,*

$$f_n(s^*_n, s^*_{-n}) \leq f_n(s_n, s^*_{-n}),\ \forall s_n \in S_n,\ n \in P.$$

The Nash equilibrium has a self-stability property such that the users at the equilibrium point can achieve a mutually satisfactory solution, and no user has the incentive to deviate. Since the VFNs are owned by different individuals, and they may act in their own interests, this property is very important to KWSOP management. The existence of Nash equilibrium shows us there is a strategy profile that all three players tend to occur. Furthermore, according to the NE, we find out whether VFNs generally have the desire to participate in the system or not.

6.2 Game property

We then study the existence of the Nash equilibrium of the KWSOP game. To proceed, we first introduce an important concept of the best response:

Definition 3. *Given the strategies s_{-n} of the other players, player n's strategy $s^*_n \in S_n$ is a best response if*

$$f_n(s^*_n, s_{-n}) \leq f_n(s_n, s_{-n}),\ \forall s_n \in S_n.$$

According to Definitions 2 and 3, we see that at the Nash equilibrium all the users play the best response strategies towards each other. Based on the concept of best response and the table of players' utilities, Fig.4 , we have the following observation for the KWSOP game. For the DU's best response, without loss of generality, we can order its set of access policies so that $U(P_1) \geq U(P_2) \geq ... \geq U(P_m)$. Then, the DU's best response will be P_1, the VFN's best response will be A, and the CA's best response will be $WP + AU$. Therefore, according to Fig. 5, the Nash equilibrium of the game is equal to: $(P_1, A, WP + AU)$.

7 FGDM Construction

Our main goal in designing FGDM is to enable DUs to outsource keyword search operations to authorized VFNs, along with an efficient search results verification mechanism that can be implemented for devices with limited resources of an IoV system. In the following, we present the construction of our FGDM in detail.

1. **Setup**$(1^n, \mathbb{U}, \mathbb{U}') \rightarrow (PK, MSK)$: This algorithm takes as input a security parameter n and two attribute sets $\mathbb{U}$ and $\mathbb{U}'$. First, it executes $\mathcal{G}(1^n)$ to generate a bilinear group parameters (p, G_1, G_2, σ). Then it randomly picks $\alpha_0, \alpha_1, \alpha_2, \Gamma \in G_1$. Now, it randomly picks $s_a \in \mathbb{Z}_p$ and $s'_b \in \mathbb{Z}_p$, for each $a \in \mathbb{U}$ and $b \in \mathbb{U}'$. Also for each $a \in \mathbb{U}$ and $b \in \mathbb{U}'$, it sets $pk_a = \alpha_0^{s_a}$ and $pk'_b = \alpha_0^{s'_b}$ as the public parameter associated with the a-th attribute in $\mathbb{U}$ and b-th attribute in $\mathbb{U}'$. Moreover, it selects an ID-based signature scheme $\Pi_{sign} = (\mathbf{Setup}_{sign}, \mathbf{KeyGen}_{sign}, \mathbf{Sign}, \mathbf{Vrfy}_{sign})$, a commitment scheme $\Pi_{com} = (\mathbf{Commit}, \mathbf{Open})$, a symmetric encryption scheme $\Pi_{enc} = (\mathbf{Enc}, \mathbf{Dec})$, a message authentication code (MAC) $\Pi_{mac} = (\mathbf{Mac}, \mathbf{Vrfy}_{mac})$, and a hash function $H : G_2 \rightarrow \{0,1\}^m$, where m is a natural number. By running $\mathbf{Setup}_{sign}(1^n)$, it generates (PK_{sign}, MSK_{sign}), where PK_{sign} and MSK_{sign} are public parameters and master secret-key associated with Π_{sign}. Afterward, it selects two random values $s, t \in \mathbb{Z}_p$ and computes $\beta_0 = \alpha_0^s$, $\beta_1 = \alpha_0^t$, $\beta_2 = \Gamma\beta_1^s$, $\beta_3 = \alpha_1^s$, $\beta_4 = \beta_0^t$, $\Sigma_1 = \sigma(\beta_0, \alpha_1)$, and $\Sigma_2 = \sigma(\beta_0^{-1}, \beta_1)$. Finally it outputs $MSK = (s, t, \Gamma, \{s_a\}_{a\in\mathbb{U}}, \{s'_b\}_{b\in\mathbb{U}'}, MSK_{sign}, \beta_3, \beta_4)$ and $PK = (n, p, G_1, G_2, \sigma, \alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1, \beta_2, \Sigma_1, \Sigma_2, \{pk_a\}_{a\in\mathbb{U}}, \{pk'_b\}_{b\in\mathbb{U}'}, H, PK_{sign}, \Pi_{sign}, \Pi_{com}, \Pi_{enc}, \Pi_{mac})$.
2. **DO.KeyGen**$(MSK, PK, ID_{do}) \rightarrow (PK_{do}, SK_{do})$: This algorithm first calls $\mathbf{KeyGen}_{sign}(MSK_{sign}, PK_{sign}, ID_{do})$ to generate a secret-key SK_{do}, where ID_{do} is the identifier of data user. Then, it randomly selects $sk_{do} \in \mathbb{Z}_p$ and calculates $pk^{(1)}_{do} = \alpha_2^{-sk_{do}}$, $pk^{(2)}_{do} = \Sigma_2^{-sk_{do}}$, $pk^{(3)}_{do} = \alpha_0^{-sk_{do}}$, and $pk_{do,a} = (pk_a\alpha_2^{-1})^{-sk_{do}}$, for each $a \in \mathbb{U}$. Finally, it outputs (PK_{do}, SK_{do}), where $PK_{do} = (pk^{(1)}_{do}, pk^{(2)}_{do}, pk^{(3)}_{do}, \{pk_{do,a}\}_{a\in\mathbb{U}}, sk_{do})$.

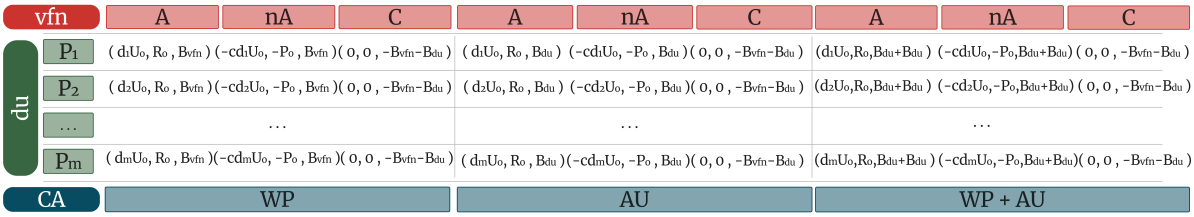


Fig. 4 Game table.

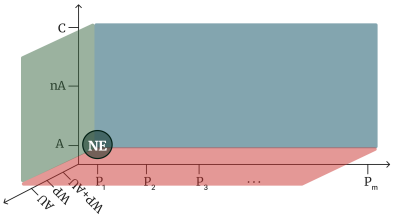


Fig. 5 NE diagram.

3. **VFN.KeyGen**$(MSK, PK, ID_{vfn}, a) \rightarrow SK_{a,vfn}$: Initially, this algorithm verifies if $a \in \mathbb{U}'$. If not, the algorithm returns $\perp$. Otherwise, it produces a secret-key associated with (a, ID_{vfn}) as follows:

$$SK_{a,vfn} = \Gamma\beta_3 ID_{vfn}^{s'_a}.$$

4. **DU.KeyGen**$(MSK, PK, ID_{du}, Att_{du}) \rightarrow SK_{du}$: Given a DU's identifier, $ID_{du} \in G_1$ and an attribute set, Att_{du}, for each $a \in Att_{du}$, this algorithm calculates $sk_{du,a} = \beta_3 \Gamma ID_{du}^{s_a}$. It returns a secret-key $SK_{du} = \{sk_{du,a}\}_{a \in Att_{du}}$.
5. **DO.Enc**$(PK, PK_{do}, SK_{do}, ID_{do}, M, \{\omega_j\}_{j=1}^m, \mathcal{T}) \rightarrow SCT_{\mathcal{T}}$: This algorithm takes $PK_{do} = (pk_{do}^{(1)}, pk_{do}^{(2)}, pk_{do}^{(3)}, \{pk_{do,a}\}_{a \in \mathbb{U}})$, an access tree $\mathcal{T}$ and selects a random value $r \in \mathbb{Z}_p$ and computes $r' = r + sk_{do}$. Then, computes $C_1 = \alpha_2^{r'} pk_{do}^{(1)} = \alpha_2^r$ and $C_2 = \Sigma_2^{r'} pk_{do}^{(2)} = \Sigma_2^r$. Afterward, by running **Share**$(p, r', \mathcal{T})$, it provides a distribution $\{D_i\}_{i \in L_{\mathcal{T}}}$ of r', and for each $i \in L_{\mathcal{T}}$, it calculates $C_{v_i} = \alpha_0^{d_i} pk_{do}^{(3)} = \alpha_0^{D_i - sk_{do}}$ and $C'_{v_i} = (pk_i \alpha_2^{-1})^{D_i} pk_{do,i} = (pk_i \alpha_2^{-1})^{D_i - sk_{do}}$. Then, it computes $k = H(\Sigma_1^{-r})$ and runs **Commit**(k) to generate a commitment pair (c_{do}, d_{do}), where c_{do} is the commitment value and d_{do} is the opening value. Also, it encrypts d_{do} to C'_{do} by running the symmetric encryption algorithm **Enc**(k, d_{do}). Afterward, it encrypts the message M to C_{do} by running **Enc**(k, M). Also, it generates a tag t_{do} for C_{do} by running **Mac**(k, C_{do}). Then, it runs **Sign**$(PK_{sign}, ID_{do}, SK_{do}, (C'_{do} || c_{do}))$ and generates a signature σ_{do}. Also, it selects $r_j \in \mathbb{Z}_p$ uniformly at random and calculates $W'_j = \Sigma_1^{-r_j}$, for each $j \in N_{do}$, where N_{do} is a DO's dictionary cardinality. Finally, for each $j = 1, ..., m$, it calculates $W_j = \omega_j - r + r_j$. It outputs a ciphertext: $SCT_{\mathcal{T}} = (\mathcal{T}, C_{do}, C'_{do}, C_1, C_2, c_{do}, t_{do}, \sigma_{do}, \{W_j\}_{j=1}^m, \{W'_j\}_{j=1}^m, \{C_{v_i}\}_{i \in L_{\mathcal{T}}}, \{C'_i\}_{i \in L_{\mathcal{T}}})$.
6. **VFN.Slc**$(\{\mathcal{T}_{p_i}\}_{p_i \in P_{du}}, \{Att_{v_i}\}_{v_i \in V_{du}}) \rightarrow \{L_i\}_{i \in P_{du}}$: It takes a DU's set of access trees, $\{\mathcal{T}_{p_i}\}_{p_i \in P_{du}}$, and its set of near VFNs' attributes, $\{Att_{v_i}\}_{v_i \in V_{du}}$ where P_{du} and V_{du} are a DU's set of access policies and set of near VFNs. For each $p_i \in P_{du}$, it considers an initially emty set, L_{p_i}, then, for each $v_j \in V_{du}$, checks whether Att_{v_j} satisfies $\mathcal{T}_{p_i}$ or not. If so, it replaces $L_{p_i} \cup Att_{v_j}$ with L_{p_i} and $\{Att_{v_i}\}_{v_i \in V_{du}} \setminus Att_{v_j}$. Finally, this algorithm returns $\{L_i\}_{i \in P_{du}}$.
7. **TokenGen**$(PK, ID_{du}, SK_{du}, \{\hat{\omega}_j\}_{j=1}^l) \rightarrow TK_{du}$: Given the DU's identifier $ID_{du} \in G_1$, secret-key $SK_{du} = \{sk_{du,a}\}_{a \in Att_{du}}$, and a keyword set $\{\hat{\omega}_j\}_{j=1}^l$, it selects $\mathcal{K}, \mathcal{K}' \in \mathbb{Z}_p$ uniformly at random and computes $\lambda_1 = \alpha_0^{\mathcal{K}}$, $\lambda_2 = \beta_1^{\mathcal{K}}$, $\lambda_3 = ID_{du}^{\mathcal{K}}$, $\lambda_4 = \alpha_0^{\mathcal{K}'}$, and $\lambda_5 = \mathcal{K}\mathcal{K}'$. Also, for each $j = 1, ..., l$, it sets $\hat{W}_j = H(\Sigma^{\hat{\omega}_j + \mathcal{K}})$, and for each $a \in Att_{du}$, it computes $\lambda_{du,a} = sk_{du,a} \alpha_2^{\mathcal{K}'}$. Finally, it outputs $(TK_{du}, \mathcal{K})$, where $TK_{du} = (\{\lambda_i\}_{i=1}^5, T_{du} = \{\lambda_{du,a}\}_{a \in Att_{du}}, \{\hat{W}_j\}_{j=1}^l)$.
8. **TokenEnc**$(PK, TK_{du}, \mathcal{T}_{du}) \rightarrow CT_{Token}$: This algorithm selects $x \in \mathbb{Z}_q$ uniformly at random and runs **Share**$(x, q, \mathcal{T}_{du}) \rightarrow \{q_{v_a}(0)\}_{v_a \in L_{\mathcal{T}_{du}}}$. Then it computes,

$$\begin{aligned} V = TK_{du}\Sigma_0^x &= (\{\lambda_i \Sigma_0^x\}_{i=1}^5, T_{du}\Sigma_0^x \\ &= \{\lambda_{du,a}\Sigma_0^x\}_{a \in Att_{du}}, \{\hat{W}_j \Sigma_0^x\}_{j=1}^l). \end{aligned}$$

Finally, the algorithm outputs:

$$CT_{Token} = (T, V, V' = \Sigma_1^x, C = \alpha_2^x, \{C_a = \alpha_0^{q_{v_a}(0)}, C'_a = (pk'_a \alpha_2^{-1})^{q_{v_a}(0)}\}_{v_a \in L_{\mathcal{T}_{du}}}).$$

9. **TokenDec**$(PK, SK_{a,vfn}, CT_{Token}) \rightarrow TK_{du}$: At first, for each $a \in S$, such that S is the attribute satisfying the access tree of a ciphertext, it calculates

$$B_{v_a,vfn} = \frac{\sigma(C_a, SK_{a,vfn})}{\sigma(C_a, \Gamma\beta_4).\sigma(C'_a, ID_{vfn})}$$

$$= \Sigma_1^{q_{v_a}}.\Sigma_2^{q_{v_a}}.\sigma(\alpha_2, ID_{vfn})^{q_{v_a}}. \quad (1)$$

Then, this algorithm runs **Combine**$(q, \mathcal{T}_{Token}, \{B^d_{v_a,vfn}\}_{j=1}^t)$ algorithm and obtains $B_{R,vfn} = \Sigma_1^k.\Sigma_2^k.\sigma(\alpha_2, ID_{vfn}))^k$. Considering,

$$B_{R,vfn} = \Sigma_1^k.\Sigma_1^k.\sigma(\alpha_2, ID_{vfn}))^k$$

$$= \Sigma_1^k.\Sigma_2^k.\sigma(\alpha_2^k, ID_{vfn}))$$

$$= \Sigma_1^k.V'.\sigma(C, ID_{vfn})), \quad (2)$$

the algorithm computes:

$$TK_{du} = \frac{V}{\frac{B_{R,vfn}}{V'.\sigma(C, ID_{vfn})}}. \quad (3)$$

10. **Search**$(PK, TK_{du}, ID_{du}, SCT_{\mathcal{T}}, \mathcal{R}) \rightarrow Re/\perp$: Given a token $TK_{du} = (\{\lambda_i\}_{i=1}^5, T_{du} = \{\lambda_{du,a}\}_{a \in Att_{du}}, \{\hat{W}_j\}_{j=1}^l)$ associated with an attribute set Att_{du}, a searchable ciphertext $SCT_{\mathcal{T}} = (\mathcal{T}, C_{do}, C'_{do}, C_1, C_2, c_{do}, t_{do}, \sigma_{do}, \{W_j\}_{j=1}^m, \{W'_j\}_{j=1}^m, \{C_{v_a}\}_{a \in L_{\mathcal{T}}}, \{C'_a\}_{a \in L_{\mathcal{T}}})$, and a threshold value $\mathcal{R}$, if an attribute set $S \subseteq Att_{du}$ satisfying $\mathcal{T}$ does not exist, it aborts. Otherwise, for any $a \in S$, this algorithm calculates:

$$L_a = \frac{\sigma(C_{v_a}\lambda_1, \lambda_{du,a})}{\sigma(C_{v_a}\lambda_1, \beta_2).\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du}).\sigma(\alpha_0, \alpha_2)^{\lambda_5}}$$

$$= (\Sigma_1.\Sigma_2.\sigma(\alpha_2, ID_{du}))^{Da - SK_{do} + \mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^{Da - SK_{do}}. \quad (4)$$

Afterward it executes **Combine**$(\{L_a\}_{a \in S}, \mathcal{T})$ to compute $L = (\Sigma_1.\Sigma_2.\sigma(\alpha_2, ID_{du}))^{r+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^r$, and then it compute $\mathcal{TK} = LC_2^{-1}.\sigma(\beta_0, \lambda_2).\sigma(C_1, ID_{du}\lambda_4)^{-1}.\sigma(\alpha_2, \lambda_3)^{-1} = \Sigma_1^{r+\mathcal{K}}$. Finally, the algorithm returns a response $Re = (C'_{do}, c_{do}, \sigma_{do}, \mathcal{TK})$ as output if $|\{H(\mathcal{TK}.\Sigma_1^{W_j}.W'_j)\}_{j=1}^m \cap \{\hat{W}_j\}j = 1^l| \geq \mathcal{R}$. Otherwise, it returns an error message $\perp$.

11. **Verify**$(PK, Re, ID_{do}, ID_{vfn}, \mathcal{K}) \rightarrow (ver \in \{0,1\})$: For $Re = (C'_{do}, c_{do}, \sigma_{do}, \mathcal{TK})$, it runs $\mathbf{Vrfy}_{sign}(PK_{sign}, ID_{do}, (C'_{do}||c_{do}), \sigma_{do})$ to verify the signature σ_{do}. If the signature is not valid, an error message $\perp$ is output. Otherwise, it computes $k' = H(\mathcal{TK}^{-1}.\Sigma_1^{\mathcal{K}})$ and $d'_{do} = Dec(k', C'_{do})$ and verifies the commitment c_{do}, by running **Open**(c_{do}, d'_{do}). If the commitment is confirmed, the algorithm returns $(1, ID_{vfn})$ meaning the search result is reliable. Otherwise, $(0, ID_{vfn})$ is output which means the search result is not valid.
12. **DU.Dec**$(PK, \mathcal{K}, \mathcal{TK}, C_{do}, t_{do}) \rightarrow M/\perp$: At first, this algorithm computes $k = H(\mathcal{TK}^{-1}\Sigma_1^{\mathbb{K}})$. Then, it executes $\mathbf{Vrfy}_{mac}(k, C_{do}, t_{do})$ to check whether the ciphertext C_{do} matches with tag t_{do} or not. If not, an error message $\perp$ is returned as output. Otherwise, by running the symmetric decryption algorithm, it recovers the message $M = \mathbf{Dec}(k, C_{do})$.
13. **VFN.AttUpdate**$(PK, a_0, MSK) \rightarrow UK_{a_0}$: This algorithm selects $\hat{s}'_{a_0} \in \mathbb{Z}_q$ uniformly at random and sets $UK_{a_0} = \hat{s}'_{a_0} s'^{-1}_{a_0}$.
14. **VFN.KeyUpdate**$(PK, a_0, UK_{a_0}, SK_{a_0,vfn}, ID_{vfn}) \rightarrow SK'_{a,vfn}$: Given an update-key UK_{a_0} and a VFN with identifier ID_{vfn}, this algorithm updates a secret-key $SK_{a_0,vfn}$ of the VFN as follows: $SK'_{a,vfn} = SK_{a,vfn} ID_{vfn}^{UK_{a_0}}$.

8 Security analysis

This section demonstrates that our FGDM scheme meets the security prerequisites noted in Section 4.2. Firstly, we precisely outline indistinguishability security. Then, we verify that our scheme is secure in the standard model as per the definition. Finally, we show that our FGDM also attains unforgeability.

8.1 Indistinguishability

Suppose the following experiment for adversarial indistinguishability, where $\mathcal{A}$ and $\mathcal{C}$ represent a PPT adversary and a PPT challenger, respectively:

- Target : The adversary $\mathcal{A}$ selects an access tree $\mathcal{T}^*$ which it wants to be challenged upon, and provide it to $\mathcal{C}$.
- Setup: First, the challenger $\mathcal{C}$ chooses attribute sets $\mathbb{U}$ and $\mathbb{U}'$ along with a security parameter n. Then, to create the public parameters PK and the master secret-key MSK, it runs $\mathbf{Setup}(1^n, \mathbb{U}, \mathbb{U}')$. $\mathcal{C}$ gives PK to $\mathcal{A}$, and keeps MSK confidential. Moreover, the challenger $\mathcal{C}$ considers $ID_1, ..., ID_{q(n)}$ for a natural polynomial q, as the identifiers of DUs in a hypothetical system and returns these parameters to $\mathcal{A}$.
- Phase 1: The adversary $\mathcal{A}$ can query the following oracles for polynomially many times. The challenger $\mathcal{C}$ keeps an initially empty list L_u for each $u \in \{1, ..., q(n)\}$ and considers an initially empty list L_{KW}.
  - $\mathcal{O}_{\mathbf{DU.KeyGen}}(ID_u, Att)$: If $\mathcal{T}^*$ is satisfied by $L_u \cup Att$, then the challenger aborts. Otherwise, to produce a secret-key SK_u, it executes $\mathbf{DU.KeyGen}(MSK, PK, ID_u, Att)$. Then, it replaces $L_u \cup Att$ with L_u.
  - $\mathcal{O}_{\mathbf{TokenGen}}(ID_u, \{\hat{\omega}_j\}_{j=1}^{l}, Att)$: The challenger $\mathcal{C}$ runs $\mathbf{DU.KeyGen}(MSK, PK, ID_u, Att)$ to create a secret-key SK_u. Afterward, $\mathcal{C}$ creates $(TK_u, \mathcal{K})$ using the algorithm $\mathbf{TokenGen}(PK, ID_u, SK_u, \{\hat{\omega}_j\}_{j=1}^{l})$. $\mathcal{C}$ returns TK_u to $\mathcal{A}$, then replaces L_{KW} with $L_{KW} \cup \{\hat{\omega}_j\}_{j=1}^{l}$.
- Challenge: After the end of Phase 1, $\mathcal{A}$ selects two pairs $(M_0, \{\omega_j^{(0)}\}_{j=1}^{m})$ and $(M_1, \{\omega_j^{(1)}\}_{j=1}^{m})$, where m is an arbitrary natural number, and $|M_0| = |M_1|$. $\mathcal{C}$ verifies if $(\{\omega_j^{(0)}\}_{j=1}^{m} \cup \{\omega_j^{(1)}\}_{j=1}^{m}) \cap L_{KW} \neq \emptyset$. If so, it aborts. If not, after randomly selecting a coin $b \in \{0, 1\}$, it executes $\mathbf{DO.KeyGen}(PK, PK_{do}, SK_{do}, ID_{do}, M_b, \{\omega_j^{(b)}\}_{j=1}^{m}, \mathcal{T}^*)$ to create a ciphertext $SCT_{\mathcal{T}^*}$. $\mathcal{C}$ gives $SCT_{\mathcal{T}^*}$ and PK_{do} to $\mathcal{A}$.
- Phase 2: This phase is comparable to Phase 1, with the exception that adversary $\mathcal{A}$ is unable to make a query on $\mathcal{O}_{\mathbf{TokenGen}}(ID_{du}, \{\hat{\omega}_j\}_{j=1}^{l}, Att)$ if $\{\hat{\omega}_j\}_{j=1}^{l} \cap (\{\omega_j^{(0)}\}_{j=1}^{m} \cup \{\omega_j^{(1)}\}_{j=1}^{m}) \neq \emptyset$.
- Guess: $\mathcal{A}$ returns a guess $b' \in \{0, 1\}$ of b.

The winner of the indistinguishability game is $\mathcal{A}$ if and only if $b' = b$. Let us consider a function $Adv_{\mathcal{A},\Pi}(n) = |Pr(b = b') - 1/2|$ named advantage function, where Π is our FGDM scheme, n is a security parameter, and $\mathcal{A}$ is a PPT adversary in the described indistinguishability experiment.

Definition 4. *Our FGDM scheme is indistinguishably secure if the advantage function $Adv_{\mathcal{A},\Pi}(n)$ is a negligible function in n for any PPT adversary $\mathcal{A}$.*

Theorem 1. *Assuming that the DBDH assumption is true for $\mathcal{G}$, in the standard model, our FGDM scheme is indistinguishably secure.*

Proof. Assume that $\mathcal{A}$ is a PPT adversary in the indistinguishability experiment. Let $\mathcal{D}$ be a PPT distinguisher that receives a tuple $(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^w)$ and wants to determine the value of w, where n is a security parameter, (p, G_1, G_2, σ) is an output of $\mathcal{G}(1^\lambda)$, $\alpha \in G_1$ is a random generator, $x, y, z \in \mathbb{Z}_p$ are selected uniformly at random, and w is either equal to xyz or is selected uniformly at random from $\mathbb{Z}_p$. We create $\mathcal{D}$ such that it emulates the challenger $\mathcal{C}$ in the described adversarial indistinguishability experiment. Later on, $\mathcal{D}$ will verify if $\mathcal{A}$ has achieved success or not. If successful, $\mathcal{D}$ will assume that $w = xyz$. If unsuccessful, it concludes that w is chosen uniformly at random from $\mathbb{Z}_p$. To elaborate, $\mathcal{D}$ executes $\mathcal{A}$ as a subroutine in the following manner:

- Target : The adversary $\mathcal{A}$ determines an access tree $\mathcal{T}^*$.
- Setup: First, the distinguisher $\mathcal{D}$ chooses universal attribute sets $\mathbb{U}$ and $\mathbb{U}'$, and for a natural polynomial q, it picks $q(n)$ identifiers $ID_1, ..., ID_{q(n)}$. Then, it selects two random values $s, t \in \mathbb{Z}_p$ and $\beta_2 \in G_1$ and sets $\alpha_0 = \alpha$, $\alpha_1 = \alpha^x$, $\alpha_2 = \alpha^t$, $\beta_0 = \alpha^y$, $\beta_1 = \alpha^x \alpha^{-s}$, $\Sigma_1 = \sigma(\beta_0, \alpha_1)$, and $\Sigma_2 = \sigma(\beta_0^{-1}, \beta_1)$. Now, it randomly picks $s_a \in \mathbb{Z}_p$ and $s'_b \in \mathbb{Z}_p$, for each $a \in \mathbb{U}$ and $b \in \mathbb{U}'$. Also for each $a \in \mathbb{U}$ and $b \in \mathbb{U}'$, it sets $pk_a = \alpha^{s_a}$ and $pk'_b = \alpha^{s'_b}$. Moreover, it selects an ID-based signature scheme $\Pi_{sign} = (\mathbf{Setup}_{sign}, \mathbf{KeyGen}_{sign}, \mathbf{Sign}, \mathbf{Vrfy}_{sign})$, a commitment scheme $\Pi_{com} = (\mathbf{Commit}, \mathbf{Open})$, a symmetric encryption scheme $\Pi_{enc} = (\mathbf{Enc}, \mathbf{Dec})$, a message authentication code (MAC) $\Pi_{mac} = (\mathbf{Mac}, \mathbf{Vrfy}_{mac})$, and a hash function $H : G_2 \to \{0, 1\}^m$, where m is a natural number. By running $\mathbf{Setup}_{sign}(1^n)$, it generates (PK_{sign}, MSK_{sign}), where PK_{sign} and MSK_{sign} are public parameters and master secret-key associated with Π_{sign}. Then, $\mathcal{D}$ gives

$\{ID_i\}_{i=1}^{q(n)}$ and $PK = (n, p, G_1, G_2, \sigma, \alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1, \beta_2, \Sigma_1, \Sigma_2, \{pk_a\}_{a\in\mathbb{U}}, \{pk'_b\}_{b\in\mathbb{U}'}, H, PK_{sign}, \Pi_{sign}, \Pi_{com}, \Pi_{enc}, \Pi_{mac})$ to $\mathcal{A}$. Note that, if we assume that $sk = y$ and $\beta_2 = \Gamma\beta_1^y = \Gamma\beta_1^{sk}$, for an unknown value $\Gamma \in G_1$, then it can be observed that PK has been appropriately chosen.

- Phase 1: The adversary $\mathcal{A}$ queries the following oracles, and for every $u \in \{1, ..., q(n)\}$, $\mathcal{D}$ creates a list L_u and responds the queries as follows:
  - $\mathcal{O}_{\mathbf{DU.KeyGen}}(ID_u, Att)$: $\mathcal{D}$ first checks whether $L_u \cup Att$ satisfies the access tree $\mathcal{T}^*$ or not. If so, it aborts. If not, for every $a \in Att$, $\mathcal{D}$ computes $sk_{u,a} = \alpha_1^s \Gamma ID_u = \beta_3 \Gamma ID_u$ and gives $\{sk_{u,a}\}_{a\in Att}$ to $\mathcal{A}$. Otherwise, to produce a secret-key SK_u, it executes $\mathbf{DU.KeyGen}(MSK, PK, ID_u, Att)$. Also, $\mathcal{D}$ replaces $L_u \cup Att$ with L_u.
  - $\mathcal{O}_{\mathbf{TokenGen}}(ID_u, \{\hat{\omega}_j\}_{j=1}^l, Att)$: The distinguisher $\mathcal{D}$, at first, calls the oracle $\mathcal{O}_{\mathbf{DU.KeyGen}}(ID_u, Att)$ to produce a secret-key SK_u. Afterward, it executes the $\mathbf{TokenGen}(PK, ID_u, SK_u, \{\hat{\omega}_j\}_{j=1}^l)$ algorithm and gives a token TK_u to $\mathcal{A}$. Also, it replaces L_{KW} with $L_{KW} \cup \{\hat{\omega}_j\}_{j=1}^l$.
- Challenge: For a natural number m, $\mathcal{A}$ returns two pairs $(M_0, \{\omega_j^{(0)}\}_{j=1}^m)$ and $(M_1, \{\omega_j^{(1)}\}_{j=1}^m)$ where m is an arbitrary natural number, and $|M_0| = |M_1|$. $\mathcal{D}$ checks if $(\{\omega_j^{(0)}\}_{j=1}^m \cup \{\omega_j^{(1)}\}_{j=1}^m) \cap L_{KW} \neq \emptyset$. If so, it aborts. If not, it selects a coin $b \in \{0, 1\}$ randomly, and encrypts $(M_b, \{\omega_j^{(b)}\}_{j=1}^m)$ as follows:

  At first, $\mathcal{D}$ chooses a random value $r' \in \mathbb{Z}_p$ and assumes that $r' = z + sk_{do}$ for an unknown value $sk_{do} \in \mathbb{Z}_p$. Then, it calculates $pk_{do}^{(1)} = \alpha_2^{-r'}\alpha^{tz} = \alpha_1^{-z-sk_{do}}\alpha_2^z = \alpha_2^{-sk_{do}}$, $pk_{do}^{(2)} = \sigma(\alpha, \alpha)^{-w}\sigma(\alpha^y, \alpha^{x-s})^{r'}\sigma(\alpha^y, \alpha^z)^s$, $pk_{do}^{(3)} = \alpha^{-r'}\alpha^z = \alpha^{-sk_{do}-z}\alpha^z = \alpha_0^{-sk_{do}}$, and $pk_{do,a} = (pk_a\alpha_2^{-1})^{-r'}(\alpha^z)^{s_a - t} = (pk_a\alpha_2^{-1})^{-sk_{do}}$, for each $a \in \mathbb{U}$. After, it computes $C_1 = \alpha_2^{r'} pk_{do}^{(1)} = \alpha_2^r$ and $C_2 = \Sigma_2^{r'} pk_{do}^{(2)} = \Sigma_2^r$. Then, by running $\mathbf{Share}(p, r', \mathcal{T})$, $\mathcal{D}$ provides a distribution $\{D_i\}_{i\in L_\mathcal{T}}$ of r', and for each $i \in L_\mathcal{T}$, it calculates $C_{v_i} = \alpha_0^{d_i} pk_{do}^{(3)} = \alpha_0^{D_i - sk_{do}}$ and $C'_{v_i} = (pk_i\alpha_2^{-1})^{D_i} pk_{do,i} = (pk_i\alpha_2^{-1})^{D_i - sk_{do}}$. Also, it runs $\mathbf{Sign}(PK_{sign}, ID_{do}, SK_{do}, (C'_{do}||c_{do}))$ and generates a signature σ_{do}. Afterward, $\mathcal{D}$ computes $k = H(\sigma(\alpha, \alpha)^{-w})$ and runs $\mathbf{Commit}(k)$, $\mathbf{Enc}(k, d_{do})$, $\mathbf{Enc}(k, M)$, and $\mathbf{Mac}(k, C_{do})$ to produce (c_{do}, d_{do}), C'_{do}, C_{do}, and t_{do}. Then, for every $j = 1, ..., m$, it chooses a value $W_j \in \mathbb{Z}_p$ uniformly at random, and considers that, for an unknown value $r_j \in \mathbb{Z}_p$, $W_j = \omega_j - z + r_j$. It sets $W'_j = \Sigma_1^{W_j}\Sigma_1^{-\omega_j}\sigma(\alpha, \alpha)^w$. Finally, the distinguisher $\mathcal{D}$ returns a searchable ciphertext $SCT^*_\mathcal{T}$ and PK_{do} to $\mathcal{A}$ where,

$$SCT_\mathcal{T} = (\mathcal{T}, C_{do}, C'_{do}, C_1, C_2, c_{do}, t_{do}, \sigma_{do}, \{W_j\}_{j=1}^m, \{W'_j\}_{j=1}^m, \{C_{v_i}\}_{i\in L_\mathcal{T}}, \{C'_i\}_{i\in L_\mathcal{T}}).$$

- Phase 2: $\mathcal{A}$ submits more queries to the oracles, and $\mathcal{D}$ answers them the same as in Phase 1.
- Guess: $\mathcal{A}$ returns a guess $b' \in \{0, 1\}$ of b.

When $\mathcal{D}$ receives b' from $\mathcal{A}$, it verifies whether $b = b'$ or not. If so, $\mathcal{D}$ returns 1 meaning $w = xyz$. If not, it outputs 0, which means w is a uniform element of $\mathbb{Z}_p$. We see that if $w = xyz$, then $k = H(\sigma(\alpha, \alpha)^{-xyz}) = H(\Sigma_1^{-z})$, $W'_j = \Sigma_1^{W_j}\Sigma_1^{-\omega_j}\sigma(\alpha, \alpha)^{xyz} = \Sigma_1^{\omega_j - z + r_j}\Sigma_1^{-\omega_j}\Sigma_1^z$, and $pk_{do}^{(2)} = \sigma(\alpha, \alpha)^{-xyz}\sigma(\alpha^y, \alpha^{x-s})^{r'}\sigma(\alpha^y, \alpha^z)^s = \sigma(\alpha^y, \alpha^{x-s})^{sk_{do}} = \sigma(\beta_0, \alpha_1^{-1})^{-sk_{do}} = \Sigma_2^{-sk_{do}}$. Therefore, the given responses to the adversary $\mathcal{D}$ are valid when $w = xyz$. Consequently, we have $Pr(\mathcal{A}(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^{xyz}) = 1) \geq Adv_{\mathcal{A},\Pi}(n) + 1/2$. On the other hand, when w is a random value, we see that $Pr(\mathcal{A}(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^w) = 1) = 1/2$. Therefore, $Pr(\mathcal{A}(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^{xyz}) = 1) - Pr(\mathcal{A}(n, p, G_1, G_2, \sigma, \alpha, \alpha^x, \alpha^y, \alpha^z, \sigma(\alpha, \alpha)^w) = 1) \geq Adv_{\mathcal{A},\Pi}(n)$. Moreover, according to the DBDH assumption, the left hand side of the above inequality, and thus $Adv_{\mathcal{A},\Pi}(n)$, is a negligible function in n. It proves the theorem.

□

8.2 Unforgeability

In this section, we show that our search results verification approach achieves unforgeability. That means, for a token TK_{du} and a threshold $\mathcal{R}$, a response Re issued by the CS passes the verification test only if for a searchable ciphertext $SCT_\mathcal{T} = (\mathcal{T}, C_{do}, C'_{do}, C_1, C_2, c_{do}, t_{do}, \sigma_{do}, \{W_j\}_{j=1}^m, \{W'_j\}_{j=1}^m, \{C_{v_i}\}_{j\in L_\mathcal{T}}, \{C'_i\}_{i\in L_\mathcal{T}})$ and $\mathcal{TK} = \mathbf{Search}(PK, TK_{du}, ID_{du}, SCT_\mathcal{T}, \mathcal{R}) \neq \perp$, Re is equal to

$(C'_{do}, c_{do}, \sigma_{do}, \mathcal{TK})$.

Theorem 2. *If the digital signature and the commitment schemes utilized in our FGDM scheme are secure, then the verification process achieves unforgeability.*

Proof. Let the VFN receives a search token TK_{du} and a threshold value $\mathcal{R}$ form a DU and outputs a tuple $Re = (C'_{do}, c_{do}, \sigma_{do}, \bar{\mathcal{TK}})$ such that $\bar{\mathcal{TK}}$ has not been generated by running the algorithm $\textbf{Search}(PK, TK_{du}, ID_{du}, SCT_{\mathcal{T}}, \mathcal{R})$, for a searchable ciphertext $SCT_{\mathcal{T}}$. Let us assume, by contradiction, that we have $\textbf{Verify}(PK, Re, ID_{do}, ID_{vfn}, \mathcal{K}) = 1$, where $\mathcal{K}$ is the private-key associated with TK_{du}. Since $\textbf{Verify}(PK, Re, ID_{do}, ID_{vfn}, \mathcal{K}) = 1$, the signature σ_{do} is valid. Therefore, according to the assumption that says the utilized digital signature scheme is secure, one concludes that C'_{do}, c_{do}, and σ_{do} are components of an outsourced searchable ciphertext $SCT_{\mathcal{T}}$. Suppose that $\hat{k} = H(\bar{\mathcal{TK}}^{-1}\Sigma_1^{\mathcal{K}})$, and k is the privet-key associated with $SCT_{\mathcal{T}}$. Since the VFN does not perform the search algorithm correctly, we can assume that $k \neq \hat{k}$, and thus $d_{do} = \textbf{Dec}(k, C'_{do}) \neq \textbf{Dec}(\hat{k}, C'_{do}) = \hat{d}_{do}$. On the other hand, as $\textbf{Verify}(PK, Re, ID_{do}, ID_{vfn}, \mathcal{K}) = 1$, we have $\textbf{Open}(c_{do}, \hat{d}_{do}) = \textbf{Open}(c_{do}, d_{do}) = 1$. Thus, if the DU cooperates with the VFN and has the d_{do}, then it can compromise the binding security of the commitment scheme in the real-time [40]. This goes against the assumption that the commitment scheme is secure. □

9 Performance analysis

To evaluate the performance of our FGDM scheme in terms of execution time, communication overhead, and storage cost, we compare its efficiency with schemes [38, 41, 42]. We have found the results of these comparisons through real implementation of aforementioned schemes and calculating their asymptomatic complexity. As for the actual performance analysis, we have implemented the aforementioned schemes using the actual Traffic Violations dataset provided by Data.gov. This dataset, which is updated daily, contains about 800MB of traffic violation information from all electronic traffic violations issued in the Montgomery County of Maryland. We performed the implementation using the python Pairing-Based Cryptography (pyPBC) [43] and hashlib [44] libraries on an Ubuntu 20.04 laptop equipped with an Intel Core i5 Processor 2.4 GHz and 4 GB RAM In addition, we have used type A pairings for these implementations. Type A pairings are constructed on the curve $y^2 = x^3 + x$ over the finite field F_q [45], for a prime number q such that $q = 3 \bmod 4$. As we mentioned before, the $\mathcal{G}$ algorithm, described in Section 3.3, takes a security parameter n and outputs a tuple (p, G_1, G_2, σ). In Type A pairings, G_1 and G_2 are p-order subgroups of F_q and F_{q^2} , respectively [45]. In this section, we consider that the length of p and q are 160-bit and 512-bit, respectively. Also, in this implementation, we use 256-bit SHA-3 algorithm as the hash function, the ID-based signature scheme presented in [46], AES (Advanced Encryption Standard) symmetric encryption, a commitment scheme $\Pi_{com} = (\textbf{Commit}, \textbf{Open})$, and a MAC scheme $\Pi_{mac} = (\textbf{Mac}, \textbf{Vrfy}_{mac})$, where Π_{com} and Π_{mac} are described below. Moreover, it is easy to check that Π_{com} is a secure commitment scheme in the random oracle model, and Theorem 4.6 and Exercise 5.11 of [47], shows that Π_{mac} is secure in the random oracle model.

- $\textbf{Commit}(M, H)$: This algorithm takes the SHA-3 hash function H and a message M as inputs. Then, for a random element k of $\mathbb{Z}_p$, it computes a commitment value $c = H(k \mid\mid M)$ and an opening value $d = (k, M)$.
- $\textbf{Open}(M, H, c, d)$: On input a message M, the SHA-3 hash function H, and commitment and opening values, c and $d = (k, M)$, if $c = H(k \mid\mid M)$, the algorithm outputs 1. Otherwise, it outputs 0.
- $\textbf{Mac}(M, H, k)$: Given a message M, the SHA-3 hash function H, and a symmetric-key k, this algorithm returns a tag $t = H(k \mid\mid M)$.
- $\textbf{Vrfy}_{mac}(M, t, k)$: For a message M, a tag t, and a symmetric-key k, it outputs 1, if $t = H(k \mid\mid M)$, and 0 otherwise.

9.1 Execution time

In Table 4 we have shown our asymptotic analysis. This table presents different computational overhead incurred using the FGDM approach and the schemes proposed in [38, 41, 42]. In this table, we considered the time consumption of pairing operation, T_p, exponential operation in G_1, T_{e_1}, exponential operation in G_2, T_{e_2}, random selection of an element from $\mathbb{Z}_p$, $T_{rand_{\mathbb{Z}}}$, a symmetric encryption and decryption, T_enc and T_dec, signature verification, T_{vrfy}, and commitment verification time, T_{open}.

Fig. 6 presents our experimental results. In this implementation, we assumed that the number of data files to be encrypted ranged between 100 and 600, the number of search results to be decrypted ranged between 10 and 60 and the number of attributes ranged from 25 to 150. Parts (a), (b), and (c) of Fig. 6 compares the actual performance of key generation

algorithms in our FGDM and the schemes presented in [38, 41, 42] (i.e. SV-KSDS, VFKSM, LFGS). As for the computational costs of key generation, the VFNs and DUs in FGDM system have much less computational burden than those in [38, 41, 42] schemes. This is because the theoretical costs of **VFNKeyGen** and **DUKeyGen** algorithms in aforementioned three schemes are $(|Att| + 5)T_{e_1} + T_{e_2} + 2T_{rand_{\mathbb{Z}}}$, $4T_{e_1} + T_{e_2} + 2T_{rand_{\mathbb{Z}}}$, $(2|Att| + 3)T_{e_1} + 2T_{e_2} + 2|Att|T_{rand_{\mathbb{Z}}}$, respectively. Part (d) of Fig. 6 compares the execution time of search token generation algorithm in mentioned schemes. As for the computational costs of token generation, DUs in our system have a much lower computational overhead than the [41] and [42] schemes. Indeed, from table 4, the number of exponential operations in these schemes grows with $2|Att|$, while in our FGDM it grows with $|Att|$, where $|Att|$ is the number of DU's attributes. However, as we see in this bar chart, in our FGDM, the token generation time overhead is relatively more than the [38] scheme. It appears that minimizing the token generation time overhead in FGDM is a feasible and compelling problem. Parts (e)-(j) of Fig. 6 illustrate the encryption time of our FGDM and other three schemes for various number of attributes. Again, we see that our FGDM is significantly more efficient than the [41] and [42] schemes, specifically when the number of attributes increases. Parts (h) and (i) compare our FGDM with other schemes in terms of search operation and decryption execution time, respectively. It can be observed that FGDM significantly decreases the amount of time required for search operations.

9.2 Storage cost and communication overhead

In comparison to the existing methods, our FGDM decreases communication overhead and storage expenses. This fact can be derived from the asymptotic analysis given in Table 5 and Fig. 7. Parts (a) , (b), and (c) of Fig. 7 present our experimental results in measuring keys length. In Part (c), we see that VFN Key length in our FGDM is incredibly lower than the length of a VFN key in other three schemes, specifically [42]. From Table 5, we notice that size of VFN key in schemes increases with the number of attributes, that of FGDM is just influenced by a variable of length l_{G_1}. Also, part (d) of the figure and Table 5 show the ciphertext length produced in the encryption process. We see that our FGDM effectively reduces the ciphertext length compare with [42] scheme.

10 Conclusion

In this paper, we designed a novel fine-grained data management (FGDM) approach for VFC-assisted IoV systems. We have shown that our FGDM provides control over both retrieval and access to outsourced data in fine-grained ways. We also demonstrated that it offers highly efficient approaches for the accuracy verification of operations performed by VFNs. In designing this system, we considered a three-player game between system entities to capture their interactions. We formulate the management problems as a Nash equilibrium problem and show the existence of an equilibrium. Using the NE, and players' utility functions, we proved that system entities, especially VFNs, not only have the tendency to take part in the system, but they also are loyal to the system and provide accurate services. We also demonstrated the deployment of our FGDM in a VFC-based IoV environment. Our FGDM was proven to achieve indistinguishability and unforgeability in the standard model. Moreover, an evaluation of its functionality and performance revealed that our FGDM is both highly effective and practical in actual applications. Furthermore, we demonstrated how incorporating a keyword threshold can enhance the adaptability of multi-keyword search protocols by enabling accurate query input from users.

11 Declarations

Ethics Approval Not applicable.
Conflict of Interest The authors declare that they have no conflict of interest.
Data Availability All data generated or analyzed during this study are included in this published article.
Author Contribution Seyedi and Ali conceived, designed the scheme, proved the security, analyzed the data, performed the experiments, and wrote the paper Liu. and Rahmati reviewed and edited the manuscript.
Funding Not applicable.
Consent to publish Not applicable.

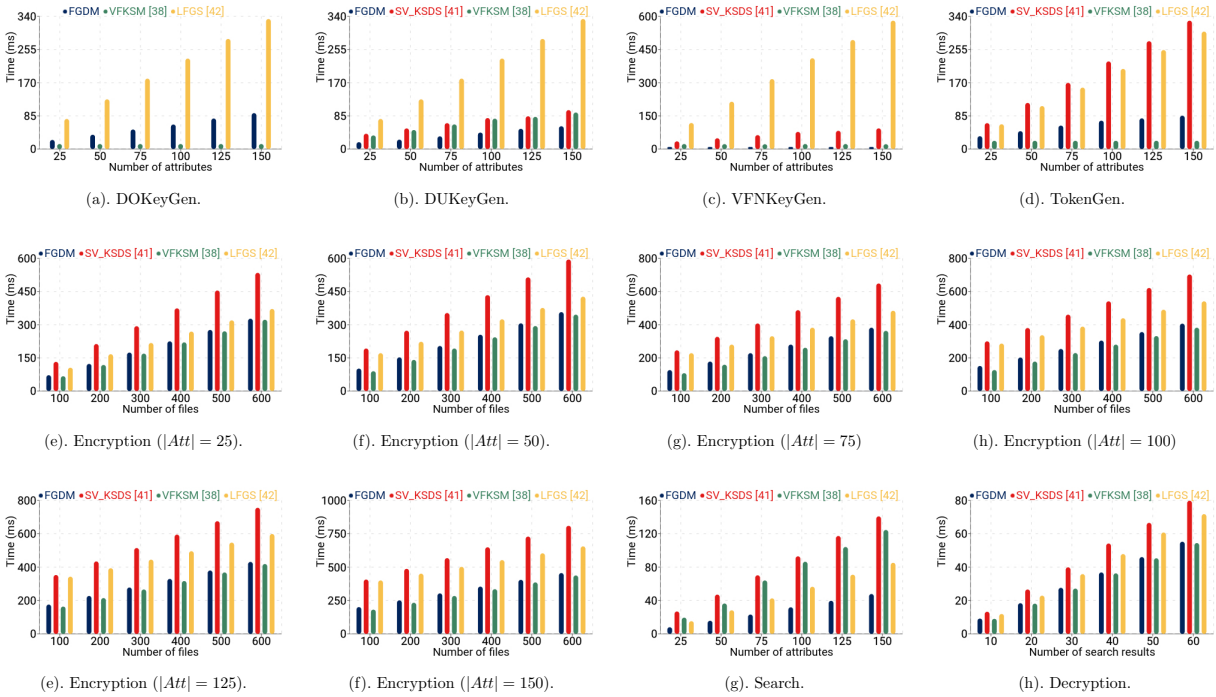
Appendix A Correctness proof

Definition 5. *Given a security parameter n, two universal attribute sets $\mathbb{U}$ and $\mathbb{U}$, an attribute set Att_{du}, an access tree $\mathcal{T}$, two keyword sets $\{\omega_j\}_{j=1}^m$ and $\{\hat{\omega}_j\}_{j=1}^m$, a threshold value $\mathcal{R}$, and identifiers ID_{do} and ID_{du}, our FGDM is said to be correct if for*

Table 4 Comparison of computation overhead.

	SV-KSDS[41]	VFKSM[38]	LFGS[42]	FGDM
DOKey.Gen	–	$T_{rand_{\mathbb{Z}}} + T_{e_1}$	$(\|Att\|+1)T_{e_1} + 2T_{e_2} + (\|Att\|+3)T_{rand_{\mathbb{Z}}}$	$(\|Att\|+3)T_{e_1} + T_{rand_{\mathbb{Z}}}$
VFNKey.Gen	$(\|Att\|+5)T_{e_1} + T_{e_2} + 2T_{rand_{\mathbb{Z}}}$	$4T_{e_1} + T_{e_2} + 2T_{rand_{\mathbb{Z}}}$	$(2\|Att\|+3)T_{e_1} + 2T_{e_2} + 2\|Att\|T_{rand_{\mathbb{Z}}}$	$2T_{e_1}$
DUKey.Gen	$(\|Att\|+6)T_{e_1} + 2T_{e_2} + 2T_{rand_{\mathbb{Z}}}$	$(\|Att\|+5)T_{e_1} + T_{e_2} + 2T_{rand_{\mathbb{Z}}}$	$(\|Att\|+1)T_{e_1} + 2T_{e_2} + (\|Att\|+3)T_{rand_{\mathbb{Z}}}$	$(\|Att\|+1)T_{e_1}$
Enc	$(4\|L_T\|+7)T_{e_1} + (2\|L_T\|+4)T_{e_2} + NT_p + 2T_{rand_{\mathbb{Z}}}$	$(\|L_T\|+2)T_{e_1} + T_{e_2} + T_{rand_{\mathbb{Z}}} + NT_{enc}$	$(\|L_T\|+5)T_{e_1} + (\|L_T\|+2)T_{rand_{\mathbb{Z}}} + NT_{enc}$	$(2\|L_T\|+2)T_{e_1} + T_{rand_{\mathbb{Z}}} + NT_{enc}$
Token.Gen	$(\|Att\|+2)(T_{e_1} + T_{e_2}) + T_{rand_{\mathbb{Z}}}$	$3T_{e_1} + 3T_{e_2} + 2T_{rand_{\mathbb{Z}}}$	$(2\|Att\|+1)T_{e_1} + T_{rand_{\mathbb{Z}}}$	$(\|Att\|+5)T_{e_1} + 2T_{rand_{\mathbb{Z}}}$
Search	$\|Att\|T_{e_2} + (2\|Att\|+1)T_P$	$(2\|Att\|+1)T_{e_1} + 2T_{e_2} + 3T_P$	$2T_{e_1} + 2T_{e_2} + (3\|Att\|+1)T_P$	$T_{e_2} + (2\|Att\|+1)T_P$
Vrfy	$\odot$	$(2N_f+1)T_{e_1} + 2T_P$	$\odot$	$T_{vrfy} + T_{open} + T_{e_1}$
Dec	$4T_{e_1} + 6T_{e_2} + 6T_P$	$2T_{e_2} + T_P + T_{dec}$	$2T_{e_1} + 3T_{e_2} + 3T_P + T_{dec}$	$T_{vrfy_{MAC}} + T_{dec}$

Notes. $|Att|$: Number of attributes, N_f: Number of retrieved files in the scheme [38], T_{vrfy}: A signature verification time, T_{open}: A commitment verification time; $\odot$: No operations.



(a). DOKeyGen. (b). DUKeyGen. (c). VFNKeyGen. (d). TokenGen.

(e). Encryption ($|Att| = 25$). (f). Encryption ($|Att| = 50$). (g). Encryption ($|Att| = 75$) (h). Encryption ($|Att| = 100$)

(e). Encryption ($|Att| = 125$). (f). Encryption ($|Att| = 150$). (g). Search. (h). Decryption.

Fig. 6 Execution time.

Table 5 Comparison of storage cost.

	SV-KSDS[41]	VFKSM[38]	LFGS[42]	FGDM
DOKeyGen	–	$l_{G_1} + l_{\mathbb{Z}_p}$	$(\|L_T\|+1)l_{G_1} + 2l_{G_2} + l_{\mathbb{Z}_p}$	$(\|L_T\|+2)l_{G_1} + l_{G_2} + l_{\mathbb{Z}_p}$
VFNKeyGen	$(\|L_T\|+3)l_{G_1}$	$(\|L_T\|+3)l_{G_1} + l_{\mathbb{Z}_p}$	$(2\|L_T\|+3)l_{G_1} + l_{G_2}$	l_{G_1}
DUKeyGen	$(\|L_T\|+3)l_{G_1}$	$2l_{G_1} + l_{\mathbb{Z}_p}$	$(2\|L_T\|+3)l_{G_1} + l_{G_2}$	$\|L_T\|l_{G_1}$
Enc	$(m(2\|L_T\|+1)+5)l_{G_1} + 2l_{G_2}$	$(\|L_T\|+m+3)l_{G_1} + 2l_{G_2} + l_{enc}$	$(2\|L_T\|+5)l_{G_1} + 2l_{G_2} + \|L_T\|l_{\mathbb{Z}_p} + l_{enc}$	$(2\|L_T\|+1)l_{G_1} + (m+1)l_{G_2} + ml_{\mathbb{Z}_p} + l_c + l_m + l_s + l_{enc}$

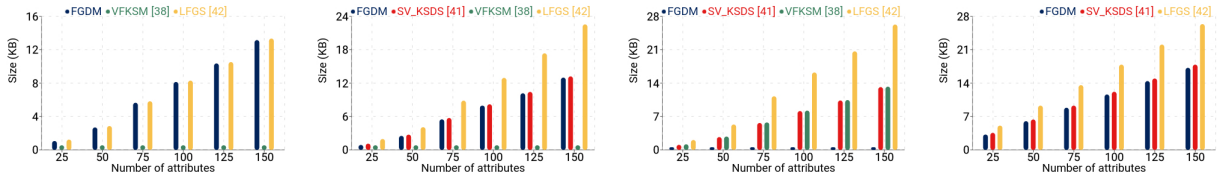
Notes. l_{enc}: Length of a ciphertext generated by the symmetric encryption algorithm, $|L_T|$: The number of leaf nodes in an access tree T, l_{G_1}: Bit-length T, l_{G_1}: Bit-length of elements in G_1, l_{G_1}: Bit-length of elements in G_2; $l_{\mathbb{Z}_p}$: Bit-length of elements in $\mathbb{Z}_p$; m: The number of keywords; l_c: Length of a commitment; l_m: Length of a tag generated by the **MAC** algorithm; l_s: Length of a signature.

each file F we have $\boldsymbol{DU.Dec}(PK, \mathcal{K}, \mathcal{TK}, C_{do}, t_{do}) = F$ *if and only if* Att_{du} *satisfies* $\mathcal{T}$ *and* $|\{\omega_j\}_{j=1}^m \cap \{\hat{\omega}_j\}_{j=1}^m| \geq \mathcal{R}$.

Theorem 3. *The proposed FGDM scheme is correct.*

Proof. Assume the presented parameters in Definition 5. First we have to show that Token Decryption is correct.

$$B_{v_a,vfn} = \frac{\sigma(C_a, SK_{a,vfn})}{\sigma(C_a, \Gamma\beta_4).\sigma(C'_a, ID_{vfn})}$$



(a). DOKey size. (b). DUKey size. (c). VFNKey size. (d). Ciphertext size.

Fig. 7 Storage costs.

$$
\begin{aligned}
&= \frac{\sigma(\alpha_0^{q_{v_a}}, \Gamma\beta_3 ID_{vfn}^{s'_a})}{\sigma(C_a, \Gamma\beta_4).\sigma(C'_a, ID_{vfn})} \\
&= \frac{\sigma(\alpha_0^{q_{v_a}}, \beta_3)\sigma(\alpha_0^{q_{v_a}}, \Gamma\beta_0^{-t}\beta_0^t ID_{vfn}^{s'_a})}{\sigma(C_a, \Gamma\beta_4).\sigma(C'_a, ID_{vfn})} \\
&= \frac{\sigma(\alpha_0^{q_{v_a}}, \beta_3)\sigma(C_a, \Gamma\beta_4)\sigma(\alpha_0^{q_{v_a}}, \beta_0^{-t})\sigma(\alpha_0^{q_{v_a}}, ID_{vfn}^{s'_a})}{\sigma(C_a, \Gamma\beta_4).\sigma(C'_a, ID_{vfn})} \\
&= \frac{\sigma(\alpha_0^{q_{v_a}}, \beta_3)\sigma(\alpha_0^{q_{v_a}}, \beta_0^{-t})\sigma(\alpha_0^{q_{v_a}}, ID_{vfn}^{s'_a})}{\sigma(C'_a, ID_{vfn})} \\
&= \frac{\sigma(\alpha_0^{q_{v_a}}, \beta_3)\sigma(\alpha_0^{q_{v_a}}, \beta_0^{-t})\sigma((\alpha_2\alpha_2^{-1}\alpha_0^{s'_a})^{q_{v_a}}, ID_{vfn})}{\sigma(C'_a, ID_{vfn})} \\
&= \frac{\sigma(\alpha_0^{q_{v_a}}, \beta_3)\sigma(\alpha_0^{q_{v_a}}, \beta_0^{-t})\sigma(C'_a, ID_{vfn})\sigma(\alpha_2^{q_{v_a}}, ID_{vfn})}{\sigma(C'_a, ID_{vfn})} \\
&= \sigma(\beta_0, \alpha_1)^{q_{v_a}}.\sigma(\beta_0^{-1}, \beta_1)^{q_{v_a}}.\sigma(\alpha_2, ID_{vfn})^{q_{v_a}} \\
&= \Sigma_1^{q_{v_a}}.\Sigma_2^{q_{v_a}}.\sigma(\alpha_2, ID_{vfn})^{q_{v_a}}.
\end{aligned}
$$

This proves Equation 1. Moreover,

$$
TK_{du} = \frac{V}{\frac{B_{R,fn}}{V'.\sigma(C, ID_{vfn})}} = \frac{TK_{du}\Sigma_1^k}{\frac{\Sigma_1^k V'\sigma(C, ID_{vfn})}{V'\sigma(C, ID_{vfn})}} = TK_{du}.
$$

This proves Equation 3.

Now, let F be an arbitrary file. In the following, we show that $\text{DU.Dec}(PK, \mathcal{K}, \mathcal{TK}, C_{do}, t_{do}) = F$ if and only if Att_{du} satisfies $\mathcal{T}$ and $|\{\omega_j\}_{j=1}^m \cap \{\hat{\omega}_j\}_{j=1}^m| \geq \mathcal{R}$. We first show that Equation 4 holds:

$$
\begin{aligned}
L_a &= \frac{\sigma(C_{v_a}\lambda_1, \lambda_{du,a})}{\sigma(C_{v_a}\lambda_1, \beta_2).\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du}).\sigma(\alpha_0, \alpha_2)^{\lambda_5}} \\
&= \frac{\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \alpha_1^s \Gamma ID_{du}^{s_a}\alpha_2^{\mathcal{K}'})}{\sigma(C_{v_a}\lambda_1, \beta_2).\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du}).\sigma(\alpha_0, \alpha_2)^{\lambda_5}} \\
&= \frac{\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \alpha_1^s \beta_1^s \beta_1^{-s} ID_{du}^{s_a}\alpha_2^{\mathcal{K}'})}{\sigma(C_{v_a}\lambda_1, \beta_2).\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du}).\sigma(\alpha_0, \alpha_2)^{\lambda_5}} \\
&= \frac{\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \alpha_1^s \beta_2 ID_{du}^{s_a}\alpha_2^{\mathcal{K}'})}{\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \beta_2).\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du}).\sigma(\alpha_0, \alpha_2)^{\lambda_5}} \\
&= \frac{\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \alpha_1^s \beta_1^{-s} ID_{du}^{s_a})\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \alpha_2^{\mathcal{K}'})}{\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du}).\sigma(\alpha_0, \alpha_2)^{\mathcal{K}\mathcal{K}'}} \\
&= \frac{\sigma(\alpha_0^{D_a - sk_{do}+\mathcal{K}}, \alpha_1^s \beta_1^{-s} ID_{du}^{s_a})\sigma(\alpha_0^{D_a - sk_{do}}, \alpha_2^{\mathcal{K}'})}{\sigma(pk_a\alpha_2^{-1}, \lambda_3).\sigma(C'_{v_a}, ID_{du})} \\
&= \frac{(\sigma(\alpha_0, \alpha_1^s\beta_1^{-s})\sigma(pk_a\alpha_2^{-1}\alpha_2, ID_{du}))^{D_a - sk_{do}+\mathcal{K}}\sigma(\alpha_0^{D_a - sk_{do}}, \alpha_2^{\mathcal{K}'})}{\sigma(pk_a\alpha_2^{-1}, ID_{du}^{\mathcal{K}}).\sigma((pk_a\alpha_2^{-1})^{D_a - sk_{do}}, ID_{du})} \\
&= (\sigma(\beta_0, \alpha_1)\sigma(\beta_0^{-1}, \beta_1)\sigma(\alpha_2, ID_{du}))^{D_a - sk_{do}+\mathcal{K}}\sigma(\alpha_0^{D_a - sk_{do}}, \alpha_2^{\mathcal{K}'}) \\
&= (\Sigma_1.\Sigma_2.\sigma(\alpha_2, ID_{du}))^{D_a - sk_{do}+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^{D_a - sk_{do}}.
\end{aligned}
$$

Now, according to the definition of the **Combine** algorithm [39], the output of $\textbf{Combine}(\{L_a\}_{a\in S}, \mathcal{T})$ is equal to

$$
\begin{aligned}
L &= (\Sigma_1.\Sigma_2.\sigma(\alpha_2, ID_{du}))^{r' - sk_{do}+\mathcal{K}}.\sigma(\lambda_4, \alpha_2)^{r' - sk_{do}} \\
&= (\Sigma_1.\Sigma_2.\sigma(\alpha_2, ID_{du}))^{r+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^r.
\end{aligned}
$$

if and only if Att_{du} satisfies $\mathcal{T}$. Moreover,

$$
\begin{aligned}
\mathcal{TK} &= LC_2^{-1}.\sigma(\beta_0, \lambda_2).\sigma(C_1, ID_{du}\lambda_4)^{-1}.\sigma(\alpha_2, \lambda_3)^{-1} \\
&= \frac{\Sigma_1^{r+\mathcal{K}}\Sigma_2^{r+\mathcal{K}}.\sigma(\alpha_2, ID_{du})^{r+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^r.C_2^{-1}\sigma(\beta_0, \lambda_2)}{\sigma(C_1, ID_{du}\lambda_4).\sigma(\alpha_2, \lambda_3)} \\
&= \frac{\Sigma_1^{r+\mathcal{K}}\Sigma_2^{r+\mathcal{K}}.\sigma(\alpha_2, ID_{du})^{r+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^r.C_2^{-1}\sigma(\beta_0, \lambda_2)}{\sigma(\alpha_2^r, ID_{du}\alpha_0^{\mathcal{K}'}).\sigma(\alpha_2, ID_{du}^{\mathcal{K}})} \\
&= \frac{\Sigma_1^{r+\mathcal{K}}\Sigma_2^{r+\mathcal{K}}.\sigma(\alpha_2, ID_{du})^{r+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^r.C_2^{-1}\sigma(\beta_0, \lambda_2)}{\sigma(\alpha_2^r, ID_{du}).\sigma(\alpha_2^r, \alpha_0^{\mathcal{K}'}).\sigma(\alpha_2, ID_{du}^{\mathcal{K}})} \\
&= \frac{\Sigma_1^{r+\mathcal{K}}\Sigma_2^{r+\mathcal{K}}.\sigma(\alpha_2, ID_{du})^{r+\mathcal{K}}.\sigma(\alpha_0^{\mathcal{K}'}, \alpha_2)^r.C_2^{-1}\sigma(\beta_0, \lambda_2)}{\sigma(\alpha_2, ID_{du})^{r+\mathcal{K}}.\sigma(\alpha_2^r, \alpha_0^{\mathcal{K}'})} \\
&= \Sigma_1^{r+\mathcal{K}}\Sigma_2^{r+\mathcal{K}}C_2^{-1}\sigma(\beta_0, \lambda_2) = \Sigma_1^{r+\mathcal{K}}\Sigma_2^{r+\mathcal{K}}\Sigma_2^{-r}\sigma(\beta_0, \beta_1^{\mathcal{K}}) \\
&= \Sigma_1^{r+\mathcal{K}}\Sigma_2^{\mathcal{K}}.\sigma(\beta_0^{-1}, \beta_1)^{-\mathcal{K}} = \Sigma_1^{r+\mathcal{K}}.
\end{aligned}
$$

Now, we see that $\mathcal{TK}.\Sigma_1^{W_j}.W'_j = \Sigma_1^{r+\mathcal{K}}.\Sigma_1^{\omega_j - r + r_j}.\Sigma_1^{-r_j} = \Sigma_1^{\mathcal{K}+\omega_j}$ On the other hand, for each $\omega, \hat{\omega} \in \mathcal{Z}_p$, we know that $\omega = \hat{\omega}$ if and only if $\Sigma_1^{\mathcal{K}+\omega} = \Sigma_1^{\mathcal{K}+\hat{\omega}}$. Then, $|\{\omega_j\}_{j=1}^m \cap \{\hat{\omega}_j\}_{j=1}^l| = |\{H(\mathcal{TK}.\Sigma_1^{W_j}.W'_j)\}_{j=1}^m \cap \{\hat{W}_j\}_{j=1}^l|$. Therefore, $|\{\omega_j\}_{j=1}^m \cap \{\hat{\omega}_j\}_{j=1}^l| \geq \mathcal{R}$ if and only if $|\{H(\mathcal{TK}.\Sigma_1^{W_j}.W'_j)\}_{j=1}^m \cap \{\hat{W}_j\}_{j=1}^l| \geq \mathcal{R}$.

So, we proved that $\textbf{Search}(PK, TK_{du}, ID_{du}, SCT_{\mathcal{T}}, \mathcal{R})$ produces a valid response $Re = (C'_{do}, c_{do}, \sigma_{do}, \mathcal{TK})$ if and only if Att_{du} satisfies $\mathcal{T}$, and $|\{\omega_j\}_{j=1}^m \cap \{\hat{\omega}_j\}_{j=1}^l| \geq \mathcal{R}$. It proves the theorem as in this case $F = \text{DU.Dec}(PK, \mathcal{K}, \mathcal{TK}, C_{do}, t_{do})$.

□

References

[1] J. Contreras-Castillo, S. Zeadally, J. A. Guerrero-Ibañez, Internet of vehicles: architecture, protocols, and security, IEEE internet of things Journal 5 (5) (2017) 3701–3709.

[2] M. S. Rathore, M. Poongodi, P. Saurabh, U. K. Lilhore, S. Bourouis, W. Alhakami, J. Osamor, M. Hamdi, A novel trust-based security and privacy model for internet of vehicles using encryption and steganography, Computers and Electrical Engineering 102 (2022) 108205.

[3] A. Hbaieb, S. Ayed, L. Chaari, A survey of trust management in the internet of vehicles, Computer Networks 203 (2022) 108558.

[4] J. Wang, K. Zhu, E. Hossain, Green internet of vehicles (iov) in the 6g era: Toward sustainable vehicular communications and networking, IEEE Transactions on Green Communications and Networking 6 (1) (2021) 391–423.

[5] W. Mao, O. U. Akgul, A. Mehrabi, B. Cho, Y. Xiao, A. Ylä-Jääski, Data-driven capacity planning for vehicular fog computing, IEEE Internet of Things Journal 9 (15) (2022) 13179–13194.

[6] A. M. A. Hamdi, F. K. Hussain, O. K. Hussain, Task offloading in vehicular fog computing: State-of-the-art and open issues, Future Generation Computer Systems (2022).

[7] M. Ali, M.-R. Sadeghi, X. Liu, Y. Miao, A. V. Vasilakos, Verifiable online/offline multi-keyword search for cloud-assisted industrial internet of things, Journal of Information Security and Applications 65 (2022) 103101.

[8] C. Tang, X. Wei, C. Zhu, Y. Wang, W. Jia, Mobile vehicles as fog nodes for latency optimization in smart cities, IEEE Transactions on Vehicular Technology 69 (9) (2020) 9364–9375.

[9] Z. Ning, J. Huang, X. Wang, Vehicular fog computing: Enabling real-time traffic management for smart cities, IEEE Wireless Communications 26 (1) (2019) 87–93.

[10] J. Feng, Z. Liu, C. Wu, Y. Ji, Ave: Autonomous vehicular edge computing framework with aco-based scheduling, IEEE Transactions on Vehicular Technology 66 (12) (2017) 10660–10675.

[11] M. Shojafar, N. Cordeschi, E. Baccarelli, Energy-efficient adaptive resource management for real-time vehicular cloud services, IEEE Transactions on Cloud computing 7 (1) (2016) 196–209.

[12] X. Peng, K. Ota, M. Dong, Multiattribute-based double auction toward resource allocation in vehicular fog computing, IEEE Internet of Things Journal 7 (4) (2020) 3094–3103.

[13] S.-s. Lee, S. Lee, Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information, IEEE Internet of Things Journal 7 (10) (2020) 10450–10464.

[14] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, Z. Li, Delay-sensitive task offloading in the 802.11 p-based vehicular fog computing systems, IEEE Internet of Things Journal 7 (1) (2019) 773–785.

[15] X. Peng, K. Ota, M. Dong, Multiattribute-based double auction toward resource allocation in vehicular fog computing, IEEE Internet of Things Journal 7 (4) (2020) 3094–3103.

[16] Q. Chai, G. Gong, Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers, in: 2012 IEEE international conference on communications (ICC), IEEE, 2012, pp. 917–922.

[17] S. Benabbas, R. Gennaro, Y. Vahlis, Verifiable delegation of computation over large datasets, in: Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31, Springer, 2011, pp. 111–131.

[18] D. Fiore, R. Gennaro, Publicly verifiable delegation of large polynomials and matrix computations, with applications, in: Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 501–512.

[19] Q. Zheng, S. Xu, G. Ateniese, Vabks: Verifiable attribute-based keyword search over

outsourced encrypted data, in: IEEE INFOCOM 2014-IEEE conference on computer communications, IEEE, 2014, pp. 522–530.

[20] Y. Lu, J. Li, Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices, IEEE Transactions on Mobile Computing 21 (12) (2021) 4397–4409.

[21] Y. Miao, R. H. Deng, K.-K. R. Choo, X. Liu, J. Ning, H. Li, Optimized verifiable fine-grained keyword search in dynamic multi-owner settings, IEEE Transactions on Dependable and Secure Computing 18 (4) (2019) 1804–1820.

[22] C. Tang, C. Zhu, X. Wei, H. Wu, Q. Li, J. J. Rodrigues, Intelligent resource allocation for utility optimization in rsu-empowered vehicular network, IEEE Access 8 (2020) 94453–94462.

[23] Y. Xiao, C. Zhu, Vehicular fog computing: Vision and challenges, in: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), IEEE, 2017, pp. 6–9.

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

[25] C. Zhu, G. Pastor, Y. Xiao, A. Ylajaaski, Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges, IEEE Communications Magazine 56 (10) (2018) 58–63.

[26] C. Tang, C. Zhu, X. Wei, W. Chen, J. J. Rodrigues, Rsu-empowered resource pooling for task scheduling in vehicular fog computing, in: 2020 International Wireless Communications and Mobile Computing (IWCMC), IEEE, 2020, pp. 1758–1763.

[27] C. Tang, S. Xia, Q. Li, W. Chen, W. Fang, Resource pooling in vehicular fog computing, Journal of Cloud Computing 10 (2021) 1–14.

[28] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24, Springer, 2005, pp. 457–473.

[29] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE symposium on security and privacy (SP'07), IEEE, 2007, pp. 321–334.

[30] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM conference on Computer and communications security, 2006, pp. 89–98.

[31] H. Nasiraee, M. Ashouri-Talouki, Anonymous decentralized attribute-based access control for cloud-assisted iot, Future Generation Computer Systems 110 (2020) 45–56.

[32] M. Ali, J. Mohajeri, M.-R. Sadeghi, X. Liu, A fully distributed hierarchical attribute-based encryption scheme, Theoretical Computer Science 815 (2020) 25–46.

[33] L.-Y. Yeh, P.-Y. Chiang, Y.-L. Tsai, J.-L. Huang, Cloud-based fine-grained health information access control framework for lightweightiot devices with dynamic auditing andattribute revocation, IEEE transactions on cloud computing 6 (2) (2015) 532–544.

[34] J. Li, S. Wang, Y. Li, H. Wang, H. Wang, H. Wang, J. Chen, Z. You, An efficient attribute-based encryption scheme with policy update and file update in cloud computing, IEEE Transactions on Industrial Informatics 15 (12) (2019) 6500–6509.

[35] S. Lin, R. Zhang, H. Ma, M. Wang, Revisiting attribute-based encryption with verifiable outsourced decryption, IEEE Transactions on Information Forensics and Security 10 (10) (2015) 2119–2130.

[36] W. Sun, S. Yu, W. Lou, Y. T. Hou, H. Li, Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, IEEE Transactions on Parallel and Distributed Systems 27 (4) (2014) 1187–1198.

[37] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, J. Ma, Privacy-preserving attribute-based keyword search in shared multi-owner setting, IEEE Transactions on Dependable and Secure Computing 18 (3) (2021) 1080–1094.

[38] Y. Miao, R. H. Deng, K.-K. R. Choo, X. Liu, J. Ning, H. Li, Optimized verifiable fine-grained keyword search in dynamic multi-owner settings, IEEE Transactions on Dependable and Secure Computing 18 (4) (2021) 1804–1820.

[39] M. Ali, J. Mohajeri, M.-R. Sadeghi, X. Liu, Attribute-based fine-grained access control for outscored private set intersection computation, Information Sciences 536 (2020) 222–243.

[40] J. B. Damgaard, Ivanand Nielsen, Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor, Springer Berlin Heidelberg, 2002.

[41] K. Gu, W. Zhang, X. Li, W. Jia, Self-verifiable attribute-based keyword search scheme for distributed data storage in fog computing with fast decryption, IEEE Transactions on Network and Service Management 19 (1) (2022) 271–288.

[42] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, H. Li, Lightweight fine-grained search over encrypted data in fog computing, IEEE Transactions on Services Computing 12 (5) (2019) 772–785.

[43] The python pairing based cryptography library, online: https://github.com/debatem1/pypbc (accessed on 10 december 2019).

[44] Secure hashes and message digests library, online: https://docs.python.org/3/library-hashlib.html# module-hashlib (accessed on 10 december 2019).

[45] B. Lynn, Pbc library manual 0.5. 11 (2006).

[46] K.-A. Shim, An id-based aggregate signature scheme with constant pairing computations, Journal of Systems and Software 83 (10) (2010) 1873–1880.

[47] J. Katz, Y. Lindell, Introduction to modern cryptography, CRC press, 2014.