11-2020

A secure flexible and tampering-resistant data sharing system for vehicular social networks

Jianfei SUN
University of Electronic Science and Technology of China

Hu XIONG
University of Electronic Science and Technology of China

Shufan ZHANG
University of Waterloo

Ximeng LIU
Fuzhou University

Jiaming YUAN
Singapore Management University, jmyuan@smu.edu.sg

See next page for additional authors

Citation

SUN, Jianfei; XIONG, Hu; ZHANG, Shufan; LIU, Ximeng; YUAN, Jiaming; and DENG, Robert H.. A secure flexible and tampering-resistant data sharing system for vehicular social networks. (2020). *IEEE Transactions on Vehicular Technology*. 69, (11), 12938-12950.
Available at: https://ink.library.smu.edu.sg/sis_research/5997

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology

A Secure Flexible and Tampering-Resistant Data Sharing System for Vehicular Social Networks

Jianfei Sun, Hu Xiong (Corresponding Author), *Member, IEEE*, Shufan Zhang, Ximeng Liu, *Member, IEEE*, Jiaming Yuan and Robert H. Deng, *Fellow, IEEE*

***Abstract*—Vehicular social networks (VSNs) have emerged as the promising paradigm of vehicular networks that can improve traffic safety, relieve traffic congestion and even provide comprehensive social services by sharing vehicular sensory data. To selectively share the sensory data with other vehicles in the vicinity and reduce the local storage burden of vehicles, the vehicular sensory data are usually outsourced to vehicle cloud server for sharing and searching. However, existing data sharing systems for VSNs can neither provide secure selective one-to-many data sharing and verifiable data retrieval over encrypted data nor ensure that the integrity of retrieved data. In this paper, we propose FTDS, a secure flexible and tampering-resistant data sharing system for VSNs by introducing a novel secure key-aggregate search encryption scheme and a tampering-resistant blockchain technology. With the proposed FTDS system for VSNs, the vehicular sensory data can be selectively shared and retrieved in a fine-grained way. Besides, our system allows vehicle data users to detect any unauthorized manipulation. Then, we present the detailed security analysis to prove that the proposed data sharing system can achieve both selective security and verifiability. We also evaluate its performance and demonstrate that it is efficient and practical for the VSNs scenarios.**

***Index Terms*—Vehicular social networks, flexible, tempering-resistant, blockchain, data sharing.**

I. Introduction

VEHICULAR transportation is commonly viewed as an indispensable part of modern cities; however, the ever-growing number of traffic congestions, road accidents, and other issues have become hindrances for the realization of smart cities [1]–[3]. As the deep convergence of social networks and Internet of vehicles, vehicle social networks (VSNs) are promising vehicular user-centric networks that mainly focus on sensory data sharing between vehicles and vehicles or roadside infrastructures to improve the traffic safety and relieve traffic congestion and even provide comprehensive social services for citizens [4]–[7]. With these potential merits, the leading IT giants have already deployed in developing and researching this area [8]–[10]. For instance, the vehicle system Carplay has been launched in March 2014 by Apple. Besides, Google has increased investments in the development of VSNs and already issued the VSNs product of Android Auto in June 2014.

In the VSNs environment, vehicles are generally equipped with smart devices or vehicular sensors (e.g., acoustic detectors, vibration sensors and chemical spill detectors) and wireless communication modules that allow vehicles to gather and broadcast sensory data (e.g. road condition, speed, location) to other vehicles and road-side units in the vicinity. Such communications are also known as vehicle-to-vehicle and vehicle-to-infrastructure communications. Vehicular sensory data in VSNs are usually collected and sent from the heterogeneous sources (e.g., infrastructure, on-board units, pedestrians, passengers, drivers, and smart mobile devices) to some centralized vehicle cloud server. Considering the sensitive nature of the sensory data (e.g., traffic data, personal data, vehicle information), there are security and privacy concerns to be addressed [11]–[13]. Before data outsourcing, encryption on the sensitive vehicular sensory data is a straightforward solution, but how to selectively grant access privileges over multiple types of data encrypted with different encryption keys remains an ongoing issue. For instance, a vehicle driver shares the traffic-related information with his/her colleagues. For distinct vehicle users who may live in different areas, they desire to be able to selectively obtain useful traffic information related to them. As promising primitives, both key-aggregate encryption (KAE) [14]–[19] and key-aggregate searchable encryption (KASE) primitives [21]–[24] can allow a vehicle data owner to achieve selective one-to-many data sharing for different encrypted vehicular sensory data. However, existing KAE schemes can just return the whole matched search results, which incurs the considerable waste of storage and computation resources since the vehicle data user may not be interested in or need to know all the returned search results. KASE indeed allows a vehicle data owner to share different encrypted data with multiple users, such that vehicle data users can search and access multiple encrypted data with a constant aggregate secret key. However, current KASE works cannot guarantee the security of the proposed scheme. This is because in existing KASE works [21]–[24], keyword privacy-leakage directly leads to that plaintext can be recovered without the secret key. *Until now, there is no such a secure data sharing scheme with KASE for the VSN that can securely and selectively share multiple encrypted data with multiple receivers.* Hence, how to design

Jianfei Sun and Hu Xiong are with the Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu, China; Prof. Hu Xiong is the corresponding author; (email: sjf215.uestc@gmail.com and xionghu.uestc@gmail.com).

Shufan Zhang is with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1. (email: sfzhang.cn@gmail.com).

Ximeng Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China; (email:snbnix@gmail.com).

Robert. H Deng and Jiaming Yuan are with the school of Information and Systems, Singapore Management University, Singapore; (e-mail: robertdeng@smu.edu.sg and jmyuan@smu.edu.sg).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology

a secure flexible KASE scheme used for the VSN scenario is an ongoing challenge.

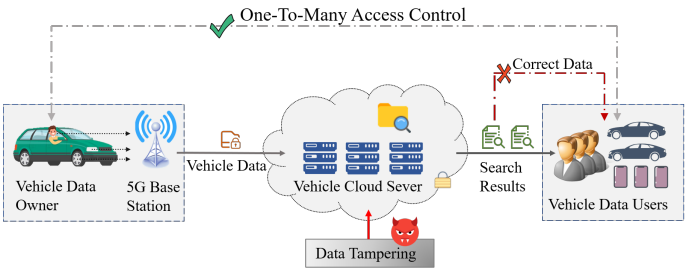


Fig. 1: Security challenges in vehicular social networks

Besides, *there is also no such a data sharing system with KASE for VSN that can well-integrate the blockchain technology and verifiable searching technology to support the data tamper resistance and verifiable searching simultaneously.* Specifically, there are several security challenges in VSNs with reference to Fig. 1. In a VSN scenario, the encrypted vehicular data stored on the vehicle cloud server may be easily vulnerable since vehicle drivers can tamper with the previous data by uploading a new version and the untrusted vehicle cloud server can physically control the stored data. When a traffic dispute or a traffic accident occurs, vehicle accident bearers have great temptations to change the sensory data for evading criminal punishments or financial compensations and impeding the exposures to maintain their reputations. Besides, the vehicle cloud server may also have strong motivations to tamper with the stored data, e.g. vehicles cloud server may help accident bearers to modify and even delete the relevant sensory records for deriving bribes. Thus, the stored vehicular data should be free from being manipulated. Further, since incorrect search results returned during keyword-based ciphertext retrieval greatly affects the quality of search experience, even misleads the vehicles to cause traffic accidents. Hence, the retrieval result returned from the vehicle cloud server to vehicle users should be verified. While there are already some efforts [25]–[27] to independently solve the problem of data tampering or verifiable retrieval, there is no such a scheme that can simultaneously support data tamper resistance and verifiability due to the heterogeneity and particularity of the respective technology. Hence, how to simultaneously achieve data tamper resistance and verifiability of searching is another challenge.

The aforementioned challenges lead to an urgent task to design a secure flexible and tampering-resistant data sharing system with verifiable retrieval for the VSNs. In this paper, we address the problems of secure one-to-many key aggregate data sharing, verifiable data retrieval and data tampering resistance simultaneously by proposing a secure **F**lexible and **T**amper-resistant **D**ata **S**haring (FTDS) system. Specifically, our FTDS solves the secure key aggregate data sharing issue in [21]–[24]. Compared to the work [26] that can just provide keyword-based ciphertext retrieval instead of supporting the verifiability and tamper resistance, our FTDS can not only support the verification of search results and secure data sharing, but also resist data tampering. While the recent work [27] can provide data tampering and data searching, the verifiable searching is not considered. In our FTDS, the issues of data tampering, verifiable searching are well resolved. The contributions of this paper are mainly as follows:

- *Selective one-to-many data sharing*: Our FTDS enables a vehicle data owner to selectively share his/her data encrypted under various encryption keys with other vehicle data users, such that the vehicle data users can securely and efficiently access these data with one authorized aggregate secret key.
- *Verifiable keyword-based ciphertext retrieval*: Our FTDS provides key aggregate keyword search over encrypted vehicular sensory data, such that the vehicle cloud server helps a vehicle data user to retrieve the target vehicular sensory data. Moreover, the vehicle data user can determine the correctness of the returned search result.
- *Data tampering resistance*: Our FTDS supports both data confidentiality and data integrity of vehicular sensory data, which ensures that vehicular sensory data can neither be learned by non-authorization vehicle users nor be manipulated by malicious users.
- *Constant trapdoor and aggregate secret key*: In our FTDS, the vehicle data users have constant trapdoor size and aggregate secret key size, which makes the decryption and retrieval operation efficient and practical.

We also formally prove that the FTDS system is secure against both the selective indistinguishability against chosen plaintext attacks (s-IND-CPA) and the selective indistinguishability against chosen keyword attacks (s-IND-CKA). In addition, the verifiability of the proposed system to be achieved is also proved and the performance of our FTDS system is evaluated to show its practicability.

Organization: The remainder of this paper is outlined below. Section II introduces the related works. In Section III, we illustrate the preliminaries used in this paper. In Section V and Section VI, the concrete FTDS system and its security analysis are given, respectively. In Section VII, the implementation and evaluation are shown. Finally, the conclusion is given in Section VIII.

II. Related Work

Existing literature in one-to-many key aggregate data sharing and searching over encrypted data can be broadly divided into key aggregate encryption (KAE) and searchable encryption (SE).

Key Aggregate Encryption. To solve the limitation that any subset of the ciphertexts can be decryptable with a constant-size decryption key in conventional public key encryption schemes, Chu *et al.* [14] proposed the notion of KAE. In a KAE scheme, a data owner is allowed to perform selective one-to-many encryption over different data encrypted with distinct encryption keys while data users could decrypt these encrypted data with a constant-size aggregate secret key. Later, a large number of KAE-related schemes focusing on different security properties, functionalities and applications have been proposed. For example, Patranabis *et al.* [15] proposed a

dynamic key aggregate cryptosystem for online data sharing, which supports dynamic revocation of user access privileges. Patranabis *et al.* [16] also put forward a provably secure key aggregate cryptosystem with broadcast aggregate keys for cloud data sharing, which is the first scheme that is secure against the chosen ciphertext attacks. Gan *et al.* [18] introduced a revocable KAE scheme for cloud data sharing. Guo *et al.* [17] raised a novel key aggregate authentication cryptosystem for data sharing in the dynamic cloud environment. Wang [19] suggested two provably secure key-aggregate cryptosystems with auxiliary inputs for cloud data sharing, which are resistant to the chosen plaintext attacks (CPA) and chosen keyword attacks (CKA), respectively. However, the above-mentioned KAE-related schemes do not provide keyword-based ciphertext retrieval.

Searchable Encryption. Since Boneh *et al.* [20] first proposed the SE scheme in the public key setting, numerous public key encryption schemes with keyword search (PEKS) have been put forward, such as identity based encryption with keyword search (IBKS) [29], [30] and attribute based encryption with keyword search (ABKS) [31], [32]. However, these schemes either only support keyword-based ciphertext retrieval in the single-user setting or have linear-size ciphertext sharing in the multi-user setting. To achieve one-to-many data sharing and searching with a constant-size search token, key aggregate searchable encryption (KASE) was first proposed by Cui *et al.* [21]. In KASE, a data owner only requires to issue a single key to a data user who only needs to delegate a single search token to the cloud for retrieving multiple encrypted data. Subsequently, several KASE schemes focusing on distinct functionalities were raised. For example, Liu *et al.* [22] put forward a verifiable KASE scheme by modifying Cui *et al's* scheme to determine the correctness of the returned search result. Later, a novel verifiable KASE scheme for data sharing and searching was proposed based on the Bloom filter technique by Li *et al.* [23]. Liu et al. [24] proposed a verifiable key aggregate searchable encryption scheme, regrettably, this scheme is proven insecure to achieve data confidentiality in the works [25], [26] and has not been well-resolved at present. To mitigate the insecure issue, Zhou *et al.* [26] suggested a novel secure KASE scheme. While the work in [26] can support secure search function it can neither achieve the verification of search results nor support data tamper resistance. Very recently, Niu *et al.* [27] suggested a blockchain-based key aggregate searchable encryption, which can achieve data tamper resistance and data searching, however the verifiable searching is not considered. Besides, it also faces the inefficiency and the same insecure issue as that in [24] due to its basic construction based on the work [24].

To summarize, as shown in TABLE I, most of the existing similar schemes were also neither designed for VSNs environment, nor support data tamper resistance, key aggregate data sharing and secure verifiable data retrieval simultaneously, which motivates our research. In this paper, we first propose a novel key aggregate searchable encryption and then incorporate the tamper-resistant blockchain technology to simultaneously realize the verification of search results, secure data sharing, data tamper resistance.

TABLE I: Functionality Comparisons in Different Schemes

Schemes	$\mathbf{F}_1$	$\mathbf{F}_2$	$\mathbf{F}_3$	$\mathbf{F}_4$	$\mathbf{F}_5$	$\mathbf{F}_6$
[14]–[19]	✓				✓	
[21]		✓				
[22]		✓	✓			
[23], [24]	✓	✓	✓			
[26]		✓	✓		✓	
[27]	✓	✓		✓		
Our FTDS	✓	✓	✓	✓	✓	✓

Note: $\mathbf{F}_1$: Key-aggregate data sharing; $\mathbf{F}_2$: Keyword-based ciphertext retrieval; $\mathbf{F}_3$: Verifiable searching; $\mathbf{F}_4$: Data tamper resistance; $\mathbf{F}_5$: Secure construction; $\mathbf{F}_6$: VSNs environment.

III. PRELIMINARIES

This section briefly introduces basic knowledge used in our paper, including security assumption, blockchain, Bloom filter and digital signature.

A. Security Assumption

***Definition** 1 (Decisional BDHE Assumption):* Let $\mathsf{BG} = (p,\ g, \mathbb{G}_0, \mathbb{G}_1, e)$ be a bilinear group, where g is a generator of $\mathbb{G}_0$. Given the following sequence of group elements:

$$h, g, g^{\sigma^1}, \ldots, g^{\sigma^n}, g^{\sigma^{n+2}}, \ldots, g^{\sigma^{2n}}.$$

The problem of the decisional Bilinear Diffie-Hellman Exponents (BDHE) [14] is that it is intractable to distinguish $Z = e(h, g^{\sigma^{n+1}})$ or a random element of $\mathbb{G}_1$.

***Definition** 2 (Modified Decisional BDHE Assumption):* Let $\mathsf{BG} = (p, g, \mathbb{G}_0, \mathbb{G}_1, e)$ be a bilinear group, where g is a generator of $\mathbb{G}_0$. Given the following sequence of group elements:

$$h, g, g^{\sigma^1}, \ldots, g^{\sigma^n}, g^{\sigma^{n+2}}, \ldots, g^{\sigma^{2n}},$$
$$g^{\alpha_3\sigma^1}, \ldots, g^{\alpha_3\sigma^n}, g^{\alpha_3\sigma^{n+2}}, \ldots, g^{\alpha_3\sigma^{2n}},$$

the problem of the modified decisional Bilinear Diffie-Hellman Exponents (BDHE) is that it is intractable to distinguish $Z = e(h, g^{\alpha_3\sigma^{n+1}})$ or a random element of $\mathbb{G}_1$.

***Theorem** 1:* The proposed modified decisional BDHE Assumption is as hard as the decisional BDHE Assumption.

Proof: The detailed proof of this theorem can be found in Appendix **??**. ■

***Definition** 3 ($(2n, n, 1)$-MSE-DDH Assumption):* Let $\mathsf{BG} = (p, g, \mathbb{G}_0, \mathbb{G}_1, e)$ be a bilinear group, where g is a generator of $\mathbb{G}_0$. Given $v_1 = g^{\alpha_1}$, $v_2 = g^{\alpha_2}$ and the following sequence of group elements:

$$g, g^{\tau_1}, g^{\tau_2}, g^{\tau_2\theta^1}, \ldots, g^{\tau_2\theta^{n-\#}}, g^{\tau_2\theta^{n+2-\#}}, \ldots, g^{\tau_2\theta^n}$$
$$g^{\alpha_1\theta^1}, \ldots, g^{\alpha_1\theta^n}, g^{\alpha_2\theta^1}, \ldots, g^{\alpha_2\theta^{2n}}, g^{\tau_2(q(\theta)\mathsf{H}_1+t)},$$
$$g^{\alpha_1\tau_1 f}, g^{\tau_2(q(\theta)\mathsf{H}_2+t)}, g^{\alpha_1\tau_2 f}, g^{\tau_2(q(\theta)+\theta^{\#})\mathsf{H}_1+t)}, g^{\tau_1 t},$$

where $q(\theta) = \prod_{j\in S}\theta^{n+1-j}$ for any subset S, $\#$ denotes the random number selected from $\{1, \ldots, n\}$. The problem of multi-sequence of exponents decisional Diffie-Hellman (*MSE-DDH*) [26] is that it is intractable to distinguish $Z = g^{(\tau_2\mathsf{H}_2+\theta^{\#}\alpha_2)f}$ or a random element of $\mathbb{G}_0$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology

B. Blockchain

As the backbone of the Bitcoin system [28], blockchain technology creates a consistent ledger shared among distributed network nodes through consensus mechanisms. The ledger permanently records a gradually appending list of transactions blocks, where each block is linked with its previous block by the cryptographic hash function. Any changes in a previous block will vary the hash value of all later blocks and be easily detected. Accordingly, this design rationale guarantees the data stored in blockchain not to be privately tampered. This useful functionality makes blockchain widely deployed in security-aware applications, such as e-voting, crowdsourcing, and file systems.

C. Bloom Filter

A Bloom filter [24], [31] is a data structure that can be seen as a n-bit array, which are initialized as 0. In this structure, k different hash functions $\mathsf{H}'_1, \ldots, \mathsf{H}'_k$ of range $\{0, \ldots, n-1\}$ are used for verification. For every element $d \in S = \{d_1, \ldots, d_m\}$, the $\mathsf{H}'_i(d)$'s bits are set as 1, where $1 \leq i \leq k$. In the step of determining whether the element d is within the data set S, if any bits corresponding to $\mathsf{H}'_i(d)$ with $1 \leq i \leq k$ are 0, then $d \notin S$ with a high probability; otherwise, $d \in S$. Assuming that the k different hash functions $\mathsf{H}'_1, \ldots, \mathsf{H}_k$ are perfectly random and m elements are hashed into the n-bit Bloom filter, then the false positive rate equals to $(1-(1-1/n)^{kn})^k \approx (1-e^{-(km)/n})^k$. A n-bit Bloom filter contains two following algorithms:

- $\mathsf{BFGen}(\{\mathsf{H}'_1, \ldots, \mathsf{H}'_k\}, \{d_1, \ldots, d_m\}) \rightarrow \mathsf{BF}$: This algorithm produces a n-bits Bloom filter by hashing the data set $\{d_1, \ldots, d_m\}$ with these independent hash functions $\{\mathsf{H}'_1, \ldots, \mathsf{H}'_k\}$.
- $\mathsf{BFVerify}(\{\mathsf{H}'_1, \ldots, \mathsf{H}'_k\}, \mathsf{BF}, d) \rightarrow \{0, 1\}$: This algorithm outputs 1 if the element d is within the data set $\{d_1, \ldots, d_m\}$ and 0 otherwise.

D. Symmetric Encryption and Digital Signature

Let Π be a symmetric encryption scheme, like AES, DES, which contains two core algorithms: **Enc** and **Dec**, which uses a symmetric key K for encryption and decryption, respectively.

Let **Sig** be an unforgeable digital signature [38], which contains three algorithms: **KeyGen**, **Sign** and **Verify**, where the **KeyGen** algorithm is used for producing the public key pp and the secret key sk, the **Sign** algorithm is to produce a signature σ for the message M with the user's private key sk and the **Verify** algorithm is to verify whether the signature σ is valid for the message M with the public key pp.

IV. Problem Statement

In this section, the system and threat models, the security requirement, and the security model are shown in detail.

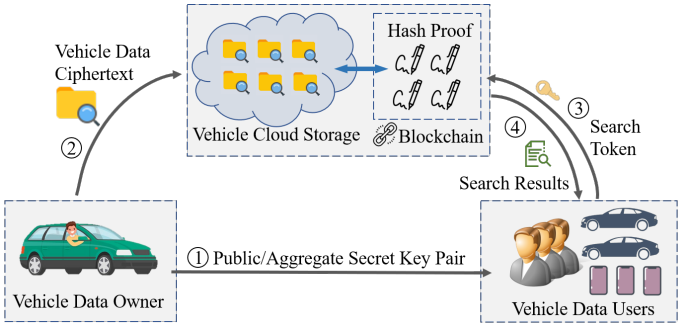


Fig. 2: System Model of the proposed FTDS system

A. System and Threat Models

The system model is as shown in Figure. 2, which consists of three entities: vehicle data owner (VDO) initializes system, issues the system public parameters to other entities and produces the credentials for vehicle data users (VDUs); Besides, it also outsources its encrypted vehicular sensory data and its index and takes charges of appending the block (hash proof) on the blockchain. The index of outsourced data (step ②) is produced by encrypted keywords that can be retrieved by the VDUs with the authorized credentials (step ①); the vehicle cloud server (VCS) renders massive cloud storage services for the VDOs and performs the retrieval tasks delegated by the VDUs (step ④); the data user VDU, who delegates the retrieval queries (step ③) to the VCS for searching the encrypted vehicular sensory data (i.e. keyword ciphertext and data ciphertext) of the VDO, and then accesses the target data. We suppose that the credentials are issued over authenticated private channels.

In the threat model, the VDO is considered as a semi-trusted entity, which means that (s)he honestly performs the encryption and uploads the encrypted sensory data but also tries to tamper his/her original data. The VCS is not fully trusted, meaning that it follows our scheme but may manipulate the vehicular sensory data and return the incorrect retrieval result to the delegated VDUs. The authorized VDUs are semi-trusted, which means that they may attempt to obtain some sensitive vehicular sensory data of interests. The malicious VDUs attempt to snoop the information from unauthorized sensory data and even collude with other VDUs. To prevent the corresponding manipulation attacks and collusion attacks from the semi-trusted VCS and VDUs in our threat model, the blockchain and cryptosystem are applied to check data integrity and ensure data confidentiality.

Remark: Blockchain is a public ledger maintained by a peer-to-peer network that provides immutable and transparent list of transaction records. It contains an increasing list of blocks of transactions shared by network peers. Network peers rely on consensus protocols to reach consistency on the shared public ledger. In this paper, a state-of-the-art PoW-based blockchain [28] is adopted because of its relatively high efficiency and stronger security guarantees. The maintenance process of PoW used in our system can be described in the following: Each node of the network in our system needs to calculate a hash value of the block header. To match a nonce

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology

Algorithm definitions in the KASE scheme

- **Setup**(1^λ): This algorithm is run by the VDO. Taking as input the security parameter 1^λ, it produces the public parameter pp and the master secret key msk.
- **sKeyGen**(pp): This algorithm is performed by the VDO. Based on the public parameter pp, it produces the public/ secret key pair (pk_s, sk_s) for the VCS.
- **Extract**$(\mathsf{pp}, S, \mathsf{msk})$: This algorithm is conducted by the VDO. Based on the public parameter pp, the file set S and the master secret key msk, this algorithm generates the aggregate retrieval key rk_{agg} and the aggregate secret key sk_{agg} for data users to conduct keyword retrieval and data access.
- **Encrypt**$(\mathsf{pp}, M, F_l, i, \omega_i)$: This algorithm is done by the VDO. Based on the public parameter pp, the message M of the l-th file F_l and i-th keyword ω_l, this algorithm produces the ciphertext C (data ciphertext and keyword ciphertext) and keyword signatures σ_{ω_i}, which will be uploaded to the VCS.
- **Trapdoor**$(rk_{agg}, \mathsf{pk}_s, \mathsf{pp}, \omega_i)$: This algorithm is executed by the VDU. This algorithm takes as input the aggregate retrieval key rk_{agg}, the public key pk_s of the server, the public parameter pp and the keyword ω_i, it produces the trapdoor $\mathsf{td} = (tr_1, tr_2)$.
- **Adjust**$(\mathsf{pp}, i, S, \mathsf{td})$: This algorithm is carried out by the VCS. Based on the public parameter pp, the file set S, the index i and the trapdoor, it generates the aggregate trapdoor $(tr_{1,i}, tr_2)$.
- **Retrieve**$(\mathsf{pp}, C, \sigma_{\omega_i}, S, i, sk_s, tr_{1,i}, tr_2)$: This algorithm is run by the VCS. Based on the public parameter pp, the ciphertext C associated with the keyword signature σ_{ω_i}, the file set S, the secret key sk_s of the VCS, the aggregate trapdoor $(tr_{1,i}, tr_2)$, if the algorithm retrieves the target keyword ciphertext, then it returns the search result and a proof Prf. Otherwise, it aborts and returns the symbol $\perp$.
- **Decrypt**$(\mathsf{pp}, S, i, \omega_i, \mathsf{Prf}, sk_{agg})$: This algorithm is executed by the VDU. Based on the public parameter pp, the file set S, the index i, the keyword ω_i, the proof Prf and the aggregate secret key sk_{agg}, it first checks data integrity and the correctness of the search results and if both are true, it then outputs 1 and recovers the plaintext M. Otherwise, it aborts and outputs the symbol $\perp$.

Fig. 3: Algorithms used in the our proposed FTDS system.

of the block header, miners (vehicle users) would frequently change the nonce to derive different hash values, such that the calculated value can reach the request of the consensus, i.e. the calculated value must be equal to or smaller than a certain given value. When the target value is reached by one node, the node would broadcast the block to other nodes and all other nodes must mutually confirm the correctness of the hash value. If the block is validated, other miners would append this new block to their own blockchains.

B. Security Requirements

The following security requirements are considered:

- **Data Privacy.** The ciphertext does not reveal any information about the plaintext to the attackers.
- **Keyword Privacy.** The keyword ciphertext does not leak any information about keyword privacy to the attackers who do not possess the authorized aggregate keys.
- **Verifiability.** The VCS can return an incorrect retrieval result to the delegated users and the effectiveness of this result can be also detected by the delegated VDU. Besides, the VDUs can verify the integrity of data from the blockchain.

Besides, **key aggregability**, **tampering-resistance** and **searchability** are also considered due to the basic technologies of key aggregate cryptosystem, blockchain and searchable encryption. As shown in Section IV-C and Section VI, the security model to satisfy the above listed requirements is defined and the corresponding security analysis is elaborated.

C. Security Model

We define the following security games for **Data Privacy**, **Keyword Privacy** and **Verifiability**:

Definition 4: (**Data Privacy**) We define the semantic security for data privacy by the following s-IND-CPA game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Data privacy game is secure against s-IND-CPA if for all attacks $|Pr[win_{\mathcal{A}}(r = r')] - 1/2| \leq \epsilon$, where ϵ denotes a negligible probability.

- **Init**: $\mathcal{A}$ gives $\mathcal{C}$ a challenge message set $S^* = \{1, \ldots, n\}$. After that, $\mathcal{C}$ selects a message class $i \in S^*$.
- **Setup**: The public parameter pp are produced from **Setup** algorithm and sent to $\mathcal{A}$.
- **Phases 1 & 2**: $\mathcal{A}$ issues the aggregate secret key sk_{agg} queries to $\mathcal{C}$, $\mathcal{C}$ then performs **Extract** algorithm to produce the sk_{agg} and sends it to $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ gives $\mathcal{C}$ two same length plaintexts M_0 and M_1, then $\mathcal{C}$ picks a random $r \in \{0, 1\}$ and encrypts M_r with **Encrypt** algorithm to produce a ciphertext.
- **Guess**: $\mathcal{A}$ outputs a guess $r' \in \{0, 1\}$, and if $r = r'$, then $\mathcal{C}$ outputs 1; otherwise, outputs 0.

Definition 5: (**Keyword Privacy**) We define the semantic security for keyword privacy by the following s-IND-CKA game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Keyword privacy game is secure against s-IND-CKA if for all attacks $|Pr[win_{\mathcal{A}}(r = r')] - 1/2| \leq \epsilon$, where ϵ denotes a negligible probability.

- **Init**: The file F^* is declared to be challenged by adversary $\mathcal{A}$.

- **Setup**: The public parameter pp is produced from **Setup** algorithm and sent to $\mathcal{A}$.
- **Phases 1 & 2**: $\mathcal{A}$ adaptively sends the trapdoor queries to $\mathcal{C}$, then $\mathcal{C}$ conducts **Trapdoor** algorithm to produce a trapdoor td for $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ gives $\mathcal{C}$ two same length keywords ω_0 and ω_1, then $\mathcal{C}$ picks a random $r \in \{0,1\}$ and encrypts ω_r with **Encrypt** algorithm to produce keyword ciphertext.
- **Guess**: $\mathcal{A}$ outputs a guess $r' \in \{0,1\}$, and if $r = r'$, then $\mathcal{C}$ outputs 1 to indicate that the keyword of trapdoor matches the keyword of ciphertext; otherwise, outputs 0.

***Definition** 6:* (**Data Verifiability**) We define the security for achieving verifiability by the following verifiability game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. This game can ensure the verifiability of return result if the signature system used in our scheme is unforgeable.

- **Setup**: Challenger $\mathcal{C}$ runs **Setup** algorithm to derive the public parameter pp and the master secret key msk, then sends pp to $\mathcal{A}$.
- **Phases 1 & 2**: $\mathcal{A}$ adaptively sends the aggregate (secret) key and the trapdoor queries to $\mathcal{C}$, $\mathcal{C}$ runs the **Extract** & **Trapdoor** algorithm to produce $(rk_{agg}, sk_{agg}, \mathsf{td}_\omega)$, where $rk_{agg}, sk_{agg}, \mathsf{td}_\omega$ denote the corresponding aggregate key, aggregate secret key and search token. Besides, $\mathcal{C}$ checks the returned proof Prf and outputs the checking result γ.
- **Challenge**: $\mathcal{A}$ gives a keyword ω^* to $\mathcal{C}$. $\mathcal{C}$ runs **Encrypt** algorithm to produce $C^* = (C^*_m, C^*_\omega)$, where C^*_m and C^*_ω refer to data ciphertext and keyword ciphertext, respectively. Next, $\mathcal{C}$ selects the aggregate retrieval key rk^*_{agg}, the aggregate secret key sk^*_{agg} such that the aggregate secret key or aggregate retrieval key matches the data ciphertext or the keyword ciphertext. Then, $\mathcal{C}$ returns $\mathcal{A}$ a trapdoor td^* by executing **Trapdoor** algorithm on the keyword ω^*.
- **Guess**: $\mathcal{A}$ outputs a proof Prf^* to $\mathcal{C}$. We say that $\mathcal{A}$ wins the game if the proof Prf^* can succeed in the process of verification.

V. The Proposed Secure Data Sharing System

This section first gives the overview of our system via blockchain and then presents our data sharing system based on the proposed cryptosystem.

A. Overview of Secure Data Sharing System via Blockchain

The proposed FTDS system consists of six phases: *System Initialization*, *User Authorization*, *Data Outsourcing*, *Trapdoor Outsourcing*, *Keyword Retrieval*, and *Data Verification and Recovery*. These phases involve the following algorithms defined in Fig. 3. In the proposed FTDS system, the setup and server's key generation algorithms perform the *System Initialization* including producing the public parameter, the public/secret key pair of the VCS and the master secret key. The extraction algorithm performs the *User Authorization* for generating an aggregate retrieval key and an aggregate secret key. When a VDO intends to share his/her vehicular sensory data with the vehicle data users (VDUs), (s)he executes the encryption algorithm to create the data ciphertext, keyword ciphertext and its signature, where the ciphertexts and keyword-related signature generated in *Data Outsourcing* phase are uploaded to the VCS for retrieval and access. Besides, in this phase, the VDO uses a hash function on ciphertext and symmetric key (hash proof), which will be uploaded on a blockchain to prevent data manipulation. In the *Trapdoor Outsourcing* phase, when a vehicle data user VDU desires to access his/her interest's data, (s)he performs the trapdoor algorithm to generate the trapdoor, which is then delegated to the VCS for conducting keyword retrieval. In the *Keyword Retrieval* phase, after receiving the trapdoor query of the VDU, the VCS first performs the adjustment algorithm to produce the aggregate trapdoor, then calls the retrieval algorithm to find the intended keyword ciphertext, and finally returns a search proof to the delegated VDU. In the *Data Verification and Recovery* phase, the VDU first verifies the data integrity and the proof validation. If the data integrity and the proof to be checked are true, the VDU then recovers the encrypted vehicular sensory data. Fig. 4 shows the detail designs.

In our FTDS system, it is unsuitable with the external authority to digitally sign the timestamped hash of the data although it is easier for the external authority maintaining the blockchain to digitally sign the timestamped hash of the data. The reasons are as follows: (1) It is impractical with the external authority to digitally sign the timestamped hash of the data. In most blockchain applications, the external authority commonly provides maintenance services of the blockchain for multiple users instead of a single data user. If using the external authority to perform signing operations on many hash of the data of different users, it means that the external authority must communicate with different data owners. In other words, the external authority must be always online, which obviously result in that the system communication cost will be prohibitive. Commonly speaking, the frequently used idea of designing a cryptosystem is to avoid the interaction between third-party authority (i.e. external authority) and other users as much as possible, which can not only reduce the communication overhead of the system, but also but also enhance the availability of the system. (2) It contradicts with the design of blockchain. If the external authority representing the data owner is to digitally sign the timestamped hash of the data, then the external authority is deemed as a trusted entity. However, in our paper, the external authority maintaining the blockchain is assumed to be not fully trusted, i.e. it can honestly perform the blockchain maintenance but still attempt to learn or tramper with the data on the blockchain. The untrusted assumption for the external authority mainly originates from two practical considerations: On the one hand, in the real-world applications, it is hard to ensure the external authority is completely honest. For example, the external authority may tamper with the data on the blockchain for some illegal economic benefits. On the other hand, if the external authority is entirely trusted, it contradicts with our design motivation that aims to prevent the data on the blockchain from being tampered or manipulated by external authority and users. That is to say, if the external authority is assumed to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology

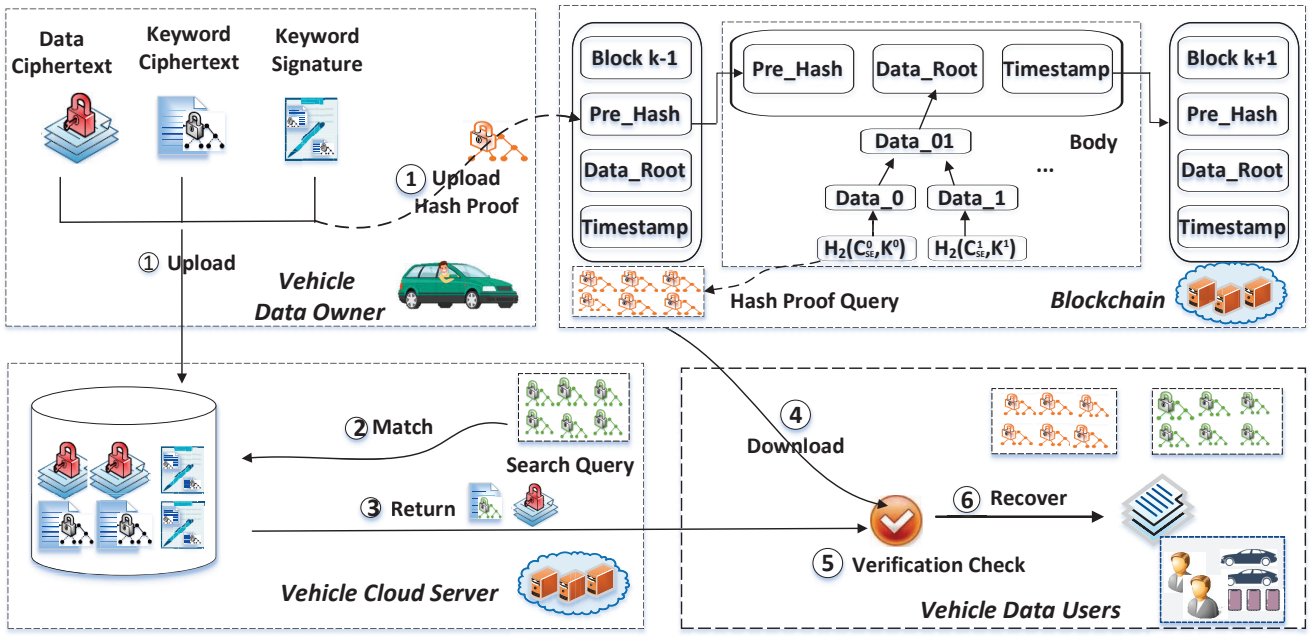


Fig. 4: The Detail Designs of Our System

be fully honest, then the blockchain technology used in our paper would not make any sense. In most practical blockchain applications, the external authority maintaining the blockchain are commonly deemed to be not completely trusted.

In our FTDS, the PoW-based (proof of work) blockchain technology [28] is adopted, this is because compared to other blockchain technologies, such as PoS (proof of stake) [33], DPoS (delegated proof of stake) [34], PBFT (practical byzantine fault tolerance) [35], Ripple [36] and Tendermint [37], the PoW-based blockchain technology can provide stronger security than others due to sacrifice of certain efficiency to enhance the safety of blockchain. The most important goal with the blockchain technology in our design is to resist the data tampering. Indeed, different blockchain technologies have various impacts on our design in terms of security, efficiency and scalability. For example, PoS, DPoS, PBFT, Ripple and Tendermint can largely improve the efficiency of blockchain. In a PoS/DPoS-based blockchain, since the search space is limited, which results in that the work of hashing the block header to retrieve the target value can be greatly reduced for each miner. In a PBFT/Ripple/Tendermint-based blockchain, the energy can be significantly saved due to no mining in the consensus process. Besides, since PBFT needs to know the identity of each miner in order to select a primary in every round while Tendermint needs to know the validators in order to select a proposer in each round, the PBFT/ Tendermint-based blockchain lack the scalability and flexibility. Compared to PBFT/Tendermint-based blockchain, nodes in the PoW/PoS/DPoS/ Ripple-based blockchain, nodes can join the network freely, which determines the flexibility and scalability of these blockchain technologies. As we all know, how to achieve a tradeoff between security and efficiency has always been a research hotspot in blockchain technologies. In our FTDS, we are inclined to choose PoW-based blockchain technology to construct our system due to its stronger security with relatively high efficiency.

B. Design of The Secure Data Sharing for VSNs

1) *System Initialization*: The VDO first runs the **Setup** algorithm to establish the system parameter. Then, (s)he also executes the **KeyGen** algorithm to produce the public parameter and the **sKeyGen** algorithm to create the public/secret key pair for the VCS.

- **Setup**(1^λ): Based on the security parameter 1^λ, it selects a bilinear group $\mathsf{BG} = (p, \mathbb{G}_0, \mathbb{G}_1, e)$ and sets the maximum number of files as n, which determines that the system allows users to encrypt n files at most using n different public keys at the same time. Then, it also chooses a collision resistant hash function $\mathsf{H}_0 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and a generator $g \in \mathbb{G}_0$. Besides, it also chooses a secure pseudo-random function $\mathsf{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^m$ and k independent universal hash functions $\mathsf{H}'_1, \ldots, \mathsf{H}'_k$ that are used to construct a m-bit Bloom filter. It also picks $\alpha_1, \alpha_2, \alpha_3, \theta, \sigma, \tau_2, \tau_3 \in \mathbb{Z}_p$, then computes $u_1 = g^{\tau_2}$, $u_2 = g^{\tau_3}$, $v_1 = g^{\alpha_1}$, $v_2 = g^{\alpha_2}$, $v_3 = g^{\alpha_3}$, $g_i = g^{\theta^i}$ and $g'_i = g^{\sigma^i}$ for $i = 1, \ldots, n$. Next, it computes the public parameters $v_{1,i} = v_1^{\theta^i}$, $v_{2,i} = v_2^{\theta^i}$, $v_{3,i} = v_3^{\sigma^i}$, $\mathsf{ek} = (ek_1 = g^{\alpha_1 \tau_1}, ek_2 = g^{\alpha_1 \tau_2})$ and destroys $v_1, v_2, v_3, g_i, \theta$. Finally, the public key $\mathsf{pp} = (\mathsf{ek}, \mathsf{BG}, \mathsf{H}_0, \mathsf{H}_1, n, g, \mathsf{H}'_1, \ldots, \mathsf{H}'_k, u_1, u_2, \{g'_i\}_{i\in[1,n]}, \{v_{1,i}\}_{i\in[1,n]}, \{v_{2,i}\}_{i\in[1,2n]}, \{v_{3,i}\}_{i\in[1,n]\cup[n+2,2n]})$ is published and the master secret key $\mathsf{msk} = (\alpha_1, \alpha_2, \tau_2, \tau_3, \{g_i\}_{i\in[1,n]})$ is retained secretly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology

- **sKeyGen**(pp): Based on the public parameter pp, it first uniformly chooses a random value τ_1 from $\mathbb{Z}_p$, sets the secret key of VCS as $sk_s = \tau_1$ and publishes the VCS's public key $pk_s = g^{\tau_1}$.

2) *User Authorization*: The VDO grants the access credential to the VDU based on the **Extract** algorithm.
   - **Extract**($\mathsf{pp}, S, \mathsf{msk}$): For given file set S, it creates an authorized aggregate retrieval key $rk_{agg} = \prod_{j\in S} g_{n+1-j}^{\tau_2}$ and an authorized aggregate secret key $sk_{agg} = \prod_{j\in S}(g'_{n+1-j})^{\tau_3}$, which would be securely sent to the authorized data user.

3) *Data Outsourcing*: The VDO performs the following **Encrypt** algorithm to generate the data ciphertext, the keyword ciphertext and the corresponding signatures. Then, (s)he sends the data ciphertext and the keyword ciphertext to the VCS for searching and sharing, and stores $\mathsf{H}_2(\Pi.CT, K)$ on the blockchain for checking data integrity (step ① in Fig. 4).
   - **Encrypt**($\mathsf{pp}, M, F_l, \omega_i$): It encrypts the i-th keyword ω_i and the message M of the l-th file. Then, the produced ciphertexts are uploaded to the server for data sharing and searching. Specifically, it first produces a Bloom filter for keyword set W_i of this file by computing: $\mathsf{BF}_i = \textbf{BFGen}(\{\mathsf{H}'_1, \ldots, \mathsf{H}'_k\}, W_i)$, then chooses a symmetric key K to encrypt the message M as $\Pi.CT \leftarrow \Pi.\textbf{Enc}(M, K)$ and chooses $r_{i,l} \in \mathbb{Z}_p$ for the symmetric key K and the keyword ω_i to compute $C = (C_{0,i,l}, C'_{0,i,l}, C_{1,i,l}, C_{2,i,l}, C_{3,i,l}, C_{4,i,l}, C_{5,i,l}, \sigma_{\omega_i})$ as follows:

$$C_{0,i,l} = K \cdot e(g'_1, v_{3,n})^{r_{i,l}}, C'_{0,i,l} = \mathsf{H}_1(K) \oplus \mathsf{BF}_i,$$
$$C_{1,i,l} = ek_1^{r_{i,l}} = g^{\alpha_1\tau_1 r_{i,l}}, C_{2,i,l} = ek_2^{r_{i,l}} = g^{\alpha_1\tau_2 r_{i,l}},$$
$$C_{3,i,l} = (u_1^{\mathsf{H}_0(\omega_i)} v_{2,i})^{r_{i,l}} = (g^{\tau_2 \mathsf{H}_0(\omega_l)} g_i^{\alpha_2})^{r_{i,l}},$$
$$C_{4,i,l} = g^{r_{i,l}}, C_{5,i,l} = (u_2 \cdot v_{3,i})^{r_{i,l}}.$$

     Next, it generates the keyword signature $\sigma_{\omega_i} = \textbf{Sig.Sign}(\textbf{Sig.sk}, C_{1,i,l}||\ C_{2,i,l}||C_{3,i,l})$ for keyword ciphertext. Finally, it submits $(\Pi.CT, C, \sigma_{\omega_i})$ to the VCS for searching, sharing and verification.

4) *Trapdoor Outsourcing*: Before the VDU accesses her/his target data, (s)he needs to locate the data of interests. So, the VDU performs the **Trapdoor** algorithm to generate the search token (trapdoor), which is subsequently delegated to the VCS for retrieval (step ② in Fig. 4).
   - **Trapdoor**($k_{agg}, \mathsf{pk}_s, \mathsf{pp}, \omega_i$): Based on the aggregate retrieval key rk_{agg}, the public key of the server pk_s, the public parameter pp and the keyword ω_i, it first randomly chooses $t \in \mathbb{Z}_p$, and then computes $tr_1 = rk_{agg}^{\mathsf{H}(\omega_i)} \cdot u_1^t$ and $tr_2 = pk_s^t = g^{\tau_1 t}$. After that, the user sends the trapdoor $\mathsf{td} = (tr_1, tr_2)$ with the intended subset S to the VCS for search query.

5) *Keyword Retrieval*: After receiving the search token from the delegated VDU, the VCS first performs the **Adjust** algorithm to produce the aggregate trapdoor and then runs the **Retrieve** algorithm to find the intended data ciphertext and return a search proof (step ③ in Fig. 4).
   - **Adjust**($\mathsf{pp}, i, S, \mathsf{td}$): After getting the trapdoor $\mathsf{td} = (tr_1, tr_2)$, this algorithm computes the aggregate trapdoor $tr_{1,i} = tr_1 \cdot \prod_{j\in S, j\neq i} v_{2,(n+1-j+i)} = tr_1 \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}^{\alpha_2}$.
   - **Retrieve**($\mathsf{pp}, C, \sigma_{\omega_i}, S, i, sk_s, tr_{1,i}, tr_2$): It first computes $pub = \prod_{j\in S} v_{1,(n+1-j)} = \prod_{j\in S} g_{n+1-j}^{\alpha_1}$ and then checks whether $e(pub, C_{3,i,l})^{\tau_1} \cdot e(tr_2, C_{2,i,l}) = e(v_{2,n+1} \cdot tr_{1,i}, C_{1,i,l})$ holds. If the equality does not hold, it aborts and returns 0; otherwise, it continues to compute $p_1 = C_{0,i,l} \cdot e(\prod_{j\in S, j\neq i} v_{3,n+1-j+i}, C_{4,i,l}) / e(\prod_{j\in S} g'_{n+1-j}, C_{5,i,l})$, then sets $p_2 = C_{4,i,l}$ and $p'_1 = C_{1,i,l}$, $p'_2 = C_{2,i,l}$, $p'_3 = C_{3,i,l}$. Finally, it outputs a proof $\mathsf{Prf} = ((p_1, p_2), (p'_1, p'_2, p'_3), \sigma_{\omega_i}, \Pi.CT)$.

6) *Data Verification and Recovery*: After getting the search proof from the VCS, the VDU first downloads hash proof from the blockchain and checks its data integrity (step ④ in Fig. 4), and then runs **Decrypt** algorithm to complete the integrity checking and retrieval verification (step ⑤ in Fig. 4). If both checking and verification are true, the VDU then recovers the cleartext (step ⑥ in Fig. 4). Otherwise, it aborts and outputs the symbol $\perp$.
   - **Decrypt**($\mathsf{pp}, S, i, \omega_l, \mathsf{Prf}, sk_{agg}$): Based on the public parameter pp, the file set S, the keyword ω_i, the proof Prf and the aggregate secret key sk_{agg}, it aborts and outputs 0 if $i \notin S$. Otherwise, it computes $K' = p_1 \cdot e(sk_{agg}, p_2)$ and then checks if $\mathsf{H}_2(\Pi.CT, K') = \mathsf{H}_2(\Pi.CT, K)$, where $\mathsf{H}_2(\Pi.CT, K)$ is obtained from the blockchain. If it does not hold, it aborts. Otherwise, it then continues to recover the i-th Bloom filter by calculating $\mathsf{BF}'_i = K' \oplus p_3$. Once there is any BF'_i that cannot be recovered, it aborts and returns 0; Otherwise, it verifies the existence of keyword ω_i by utilizing $acc_i = \textbf{BFVerify}(\{\mathsf{H}'_1, \ldots, \mathsf{H}'_k\}, \mathsf{BF}'_i, \omega_i)$. If $acc_i = 0$, it implies the keyword ω_l is not presented in the keyword group. Then, it continues to $i = i + 1$; otherwise, it runs $\eta \leftarrow \textbf{Sig.Verify}(\textbf{Sig.pk}, \sigma_i, p'_1||p'_2||p'_3)$. If $\eta = 0$, it aborts and then returns 0; otherwise, it also continues to $i = i + 1$. Finally, it outputs a set of acc_i and recovers the symmetric key K and then performs $M \leftarrow \Pi.\textbf{Dec}(\Pi.CT, K)$ to obtain the plaintext message M.

Remark: The proposed FTDS system can provide secure one-to-many key aggregate data sharing, verifiable data retrieval and data tampering resistance with the verifiable KASE and blockchain technologies. If with other verifiable KASE [23], [24] and blockchain technologies to achieve the same functionalities, the plaintext information cannot be secure against non-authorization users, which in fact contradicts data confidentiality. It is noted that it is almost impossible for an adversary with more than 51% of the computing power to manipulate the blockchain. The reason is that in a blockchain application, 51% of hash computational power is generally regarded as the

TABLE II: Theoretical analysis of computation and storage cost: A comparative conclusion

	LLL+ [23], [24]	NLG+ [27]	FTDS
Size	**Storage costs**		
Setup	(2n+1)$\|\mathbb{G}_0\| + +\|\mathsf{H}_0\| + \|\mathsf{H}_1\|$	$(2n+1)\|\mathbb{G}_0\| + \|\mathsf{H}_0\| + \|\mathsf{H}_1\| + \|\mathsf{H}_2\|$	$(5n+2)\|\mathbb{G}_0\| + \|\mathsf{H}'_1\| + \ldots + \|\mathsf{H}'_n\|$
Extract	$\|\mathbb{G}_0\|$	$2\|\mathbb{G}_0\|$	$2\|\mathbb{G}_0\|$
Encrypt	$2\ell \cdot \|\mathbb{G}_0\| + (\ell+1)\|\mathbb{G}_1\| + \|\mathsf{H}_1\|$	$(5\ell+1) \cdot \|\mathbb{G}_0\| + \|\mathbb{G}_1\| + \|\mathsf{H}_1\| + \|\sigma\|$	$5\ell \cdot \|\mathbb{G}_0\| + \|\mathbb{G}_1\| + \|\mathsf{H}_1\| + \|\sigma\|$
Trapdoor	$\|\mathbb{G}_0\|$	$\ell \cdot \|\mathbb{G}_0\|$	$2\|\mathbb{G}_0\|$
Retrieve	$3\ell \cdot \|\mathbb{G}_0\| + (\ell+1)\|\mathbb{G}_1\| + \|\mathsf{H}_1\|$	$7\ell \cdot \|\mathbb{G}_0\| + \|\mathbb{G}_1\| + \|\mathsf{H}_1\| + \|\sigma\|$	$6\ell \cdot \|\mathbb{G}_0\| + \|\mathbb{G}_1\| + \|\mathsf{H}_1\| + \|\sigma\|$
Decrypt	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$
Algorithm	**Computation costs**		
Setup	$(2n+1)\mathsf{e}_0$	$(2n+1)\mathsf{e}_0$	$(5n+5)\mathsf{e}_0$
Extract	$\|S\|\mathsf{e}_0$	$n \cdot \|S\|\mathsf{e}_0$	$2\|S\|\mathsf{e}_0$
Encrypt	$3\mathsf{p} + \ell \cdot (2\mathsf{e}_0 + 3\mathsf{e}_1 + \mathsf{H}_0) + \mathsf{H}_1$	$2\mathsf{p} + \ell \cdot ((2n+2)\mathsf{e}_0 + \mathsf{e}_1 + \mathsf{H}_0)$	$\mathsf{p} + \ell \cdot (6\mathsf{e}_0 + \mathsf{e}_1 + \mathsf{H}_0) + \mathsf{H}_1$
Trapdoor	e_0	e_0	$2\mathsf{e}_0$
Retrieve	$4\|S\|\mathsf{p}$	$(4\mathsf{p} + 2 \cdot n \cdot \mathsf{e}_1)\|S\|$	$(5\mathsf{p} + 1\mathsf{e}_1)\|S\|$
Decrypt	$\mathsf{p} + \mathsf{H}_1$	$2\mathsf{p} + \mathsf{H}_1$	$\mathsf{p} + \mathsf{H}_1$

threshold for one to gain control of the network. Once a party with 51% of hash computational power (called 51% attacks), then it can arbitrarily manipulate and modify the blockchain information. In this way, the blockchain makes no any sense. Hence, in all blockchain applications, the hash computational power of a party is limited within 50% of the computing power, such as the hash computational power of each miner in PoW-based blockchain is no more than 25% of the computing power.

VI. Security Analysis

In this section, we present the security analysis of the proposed data sharing system.

From the threat model presented in Section IV-A, our FTDS system security based on blockchain technology and our novel proposed cryptosystem. As a well-known secure technology, Blockchain can be resistant to data manipulation attacks. The FTDS cryptosystem incorporates hash function, symmetric encryption, and verifiable KASE. In which, the collision-resistant hash function is to prevent the adversary from breaching data integrity. Symmetric encryption like AES, DES can be used to secure against varieties of attacks. For the security of verifiable KASE, the strict security proof of selectively indistinguishable against chosen plaintext (keyword) attacks (s-IND-CP(K)A) is shown to demonstrate the cryptosystem security. Besides, the security analysis of verifiability is also shown to ensure the correctness of returned search results. The following theorems to achieve data privacy, keyword privacy and verifiability are presented. The further details of security proof can be referred to the Appendix. A.

Theorem 2: Suppose that the modified decisional n-BDHE assumption holds, then our cryptosystem can achieve selective s-IND-CPA security.

Theorem 3: Our cryptosystem achieves semantic security on the s-IND-CKA game under the random oracle model if the $(2n, n, 1)$-*MSE-DDH* assumption holds.

Theorem 4: If **Sig** is an unforgeable signature system, our cryptosystem achieves the verifiability.

VII. Implementation and Evaluation

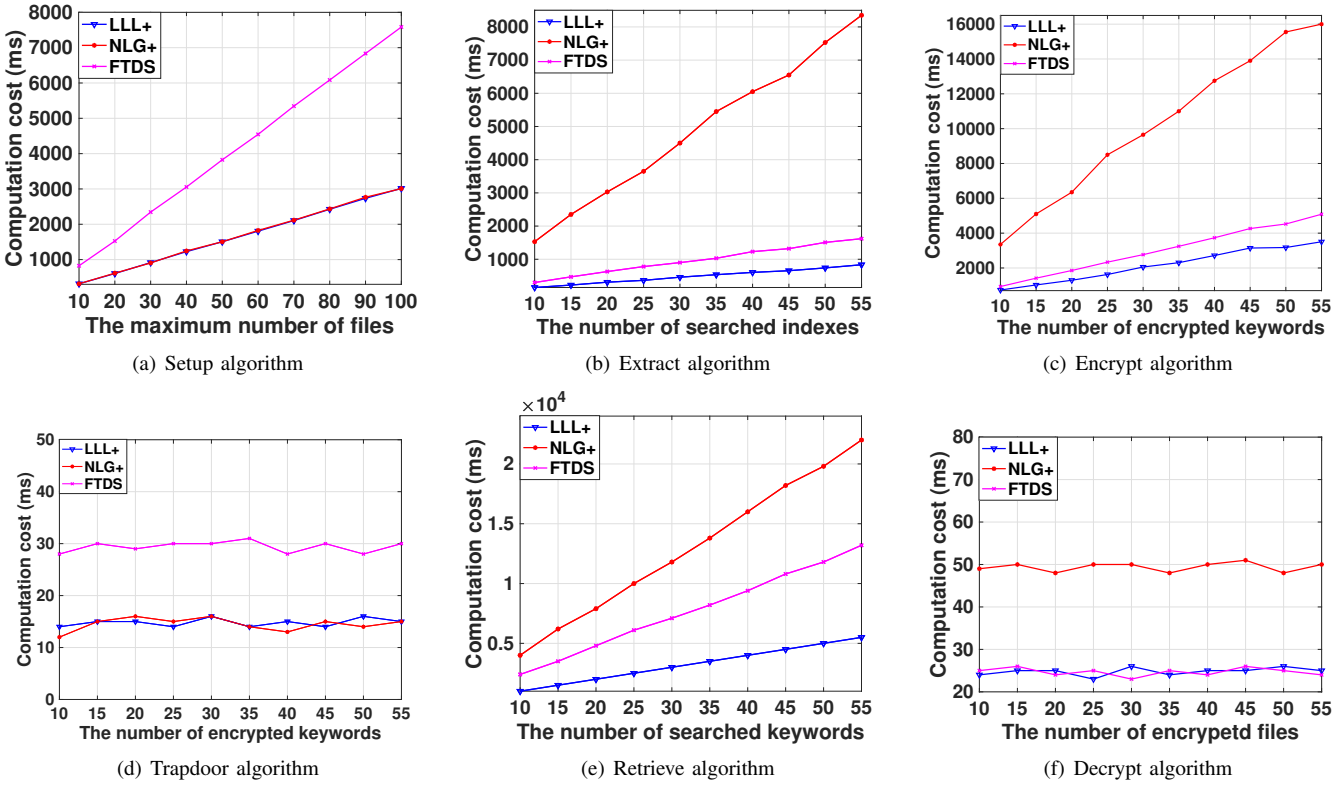
In this section, we compare our verifiable KASE scheme used for constructing the secure data sharing system with other existing verifiable KASE schemes [23], [24], [27] in terms of computation overhead and storage cost for various algorithms. In addition, we also conduct the experimental comparisons between our FTDS scheme and other related schemes.

A. Theoretical Evaluation

To analyze the theoretical computation cost, we consider several main operations: modular exponentiation operation e_0 (or e_1) in $\mathbb{G}_0$ (or $\mathbb{G}_1$), pair operation p, hash operation H_0 (or H_1) which maps the arbitrary string into $\mathbb{Z}_p$ (or $\{0,1\}^m$) and signature verification σ. As for the theoretical storage cost, we define the element lengths in $\mathbb{G}_0$ and $\mathbb{G}_1$ as $|\mathbb{G}_0|$ and $|\mathbb{G}_1|$, respectively and we denote an element with $\mathsf{H}_1(\cdot)$ as $|\mathsf{H}_1|$ and a signature length as $|\sigma|$. S and ℓ represent the number of searched indexes and encrypted/queried keywords, respectively. The theoretical analysis of the schemes [23], [24] and our FTDS scheme are illustrated in TABLE II.

From TABLE II, we can notice that the storage and computation costs of the scheme [23] are the same as that of the scheme [24] due to that the work [24] is an extension of the work [23]. Besides, we see that the storage and computation costs of our proposed FTDS construction are slightly more than those of other two schemes [23], [24] in **Extract**, **Encrypt**, **Trapdoor** and **Retrieve** and our storage cost are the same as that of the schemes [23], [24] in **Decrypt**. The reason leading to this difference result originates from that guaranteeing more stronger security of the proposed FTDS scheme needs to generate more computation and storage costs. Compared to the work [27], the storage cost of FTDS is lower than that of the work [27] in terms of **Trapdoor**, **Encrypt** and **Retrieve** algorithm. The storage cost of **Setup** in the work [27] is a bit lower than that of FTDS, and the storage cost of **Extract** and **Decrypt** in [27] are almost same as that in FTDS. The reason resulting in relatively high storage cost of **Setup** is that requiring to produce more public parameters to achieve verifiability of searching. For computation cost, our FTDS has lower computation cost than [27] in therm of **Setup**, **Extract**, **Encrypt**, **Retrieve** and **Decrypt** algorithm. We can also find that the storage costs of these works in **Encrypt** and **Retrieve** are linear increasingly with the number of encrypted keywords while those of these works in **Extract** while **Trapdoor** and **Decrypt** are constant. The computation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology



(a) Setup algorithm (b) Extract algorithm (c) Encrypt algorithm

(d) Trapdoor algorithm (e) Retrieve algorithm (f) Decrypt algorithm

Fig. 5: Computation cost comparison for each algorithm in related schemes.

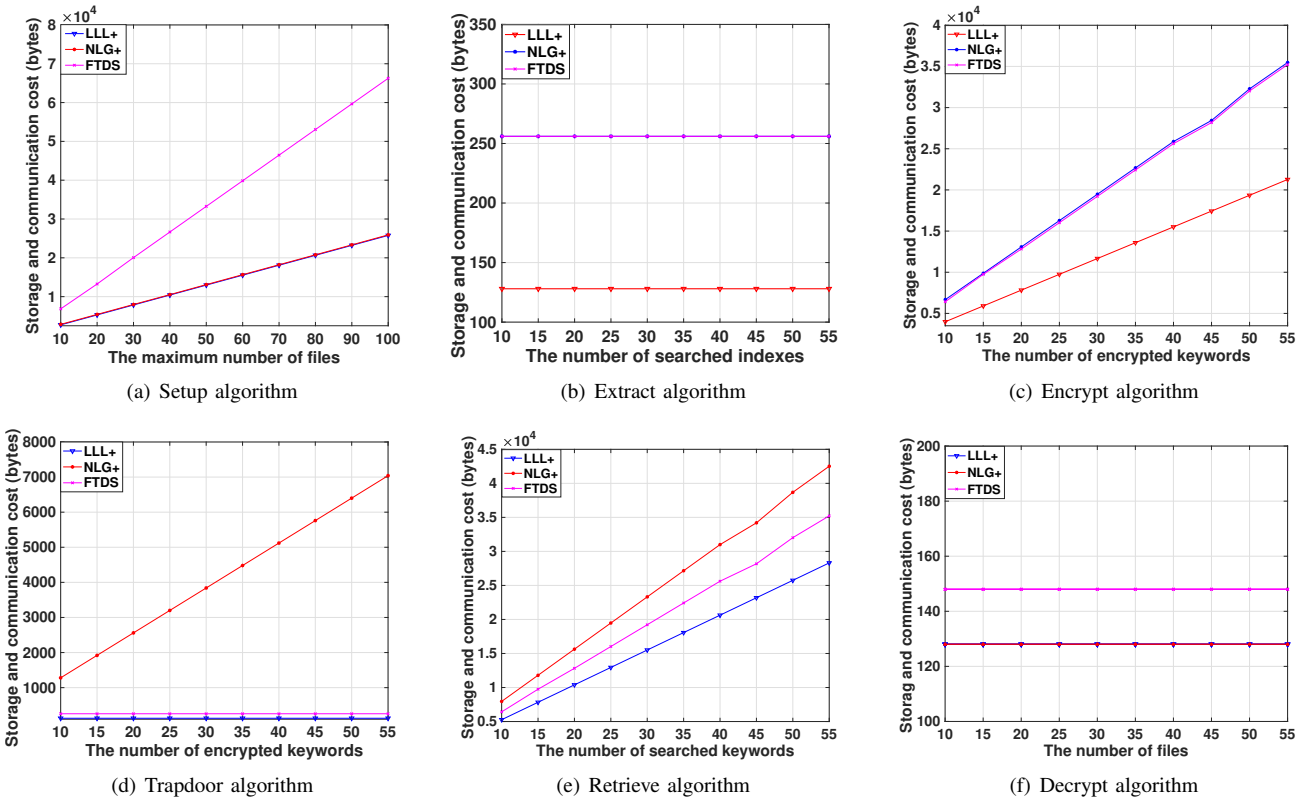
costs of these schemes in **Extract**, **Encrypt** and **Retrieve** follow the linear relationship with the number of searched indexes and encrypted keywords while those of these schemes in **Trapdoor** and **Decrypt** are constant. Note that in FTDS the storage and computation efficiency is sacrificed to ensure stronger security and more functionalities while the others even cannot ensure the security of the encrypted plaintext-see the functionality comparison in TABLE I.

B. Experimental Evaluation

To practically evaluate the performance of the FTDS scheme and other related works, we implement the simulations on a window 10 64 bits operation system with Intel(R) 8 Core(TM) i7-7820HK CPU @2.9 GHz and 16GB RAM. We use the version of Intellij IDEA-2018.2.5, Java 8 and install the latest JPBC library [39] for underlying cryptographic operations. This cryptographic function called Type-A is defined as $E(F_q)$: $y^2 = x^3 + x$, and $\mathbb{G}_0, \mathbb{G}_1$ of order p are the subgroups of $E(F_q)$, where the lengths of p and q are 160 bits and 512 bits, respectively. Note that $|\mathbb{Z}_p| = 160$ bits, $|\mathbb{G}_0| = |\mathbb{G}_1| = 1024$ bits. We also use secure hash function SHA-1 and the elliptic curve digital signature algorithm (ECDSA) [38] as the hash function H_1 and digital signature used in our scheme. All our experimental results are based on the average time of 100 times.

Fig. 5 shows the computation cost comparisons for each algorithm in related similar works (i.e. [23], [24], [27] and FTDS) under the different number of encrypted keywords and accessed indexes, where computation cost refers to the execution time of the cryptographic algorithms. In each sub-figure of Fig. 5, we vary the number of encrypted keywords (or searched indexes) l (or $|S|$) from 10 to 55 and the maximum number of files n from 10 to 100, such that the chosen number is enough for experimental performance evaluation. We can see that the computation costs for the setup algorithm of the works [23], [24], [27] and our FTDS shown in Fig. 5(a) are almost linear increasingly with the number of files. We can also observe that our computation cost in setup phase is indeed higher than that in other works [23], [24], [27]. This is because more public parameters need to be generated in our setup stage in order to achieve the verifiability and data tamper resistance while the other works have not considered achieving these properties simultaneously. From Fig. 5(b), we can learn that the computation costs of Extract algorithm in these works are all linearly increasing with the number of searched indexes. Compared to the works [23], [24], [27], our computation cost of Extract algorithm is much lower than that of NLG+ [27] and slightly higher than that of LLL+ [23], [24]. From Fig. 5(c), it is easy to find that the computation costs of encryption algorithm in all these schemes are also scaling linearly with the number of encrypted keywords. By comparison to the works [23], [24], [27], the computation cost of our Encrypt algorithm is clearly lower than that of NLG+ [27] and a bit higher than that of LLL+ [23], [24]. By the observation of Fig. 5(d), it is not hard to get that the computation costs of trapdoor generation algorithm in these schemes are almost stable regardless of the number of encrypted keywords. We can also easily get that the computation cost of our Trapdoor

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.3015916, IEEE Transactions on Vehicular Technology



(a) Setup algorithm (b) Extract algorithm (c) Encrypt algorithm

(d) Trapdoor algorithm (e) Retrieve algorithm (f) Decrypt algorithm

Fig. 6: Storage and communication cost comparison for each algorithm in related schemes.

algorithm is indeed higher than that of the others [23], [24], [27]. The reason originates from that some exponentiation operations in our FTDS are required for trapdoor generation while the others in the trapdoor generation phase just need to perform some simple multiplication operations. From Fig. 5(e), we can obtain that the computation costs of encryption algorithm in these schemes follow the linear relationship with the number of searched keywords. Compared to the works [23], [24], [27], the computation cost of our Retrieve algorithm is relatively higher than that of LLL+ [23], [24], but obviously lower than that of NLG+ [27]. As well, we can easily conclude from Fig.5(f) that the computation costs of Decrypt algorithm in these schemes almost have no relationship with the number of encrypted files. We can also find that the computation cost of Decrypt algorithm in NLG+ [27] is much higher than that in LLL+ [23], [24] and our FTDS. From the above illustrations, we can conclude that our FTDS can completely achieve better efficiency than the work [27] in all algorithms. Compared to the works [23], [24], our work indeed has a slightly higher computation cost. However, our work can ensure the security of the proposed scheme (the works [23], [24] are proven insecure in [25]) and achieve more desirable functionalities (such as data tamper resistance and secure verifiability). In summary, our FTDS scheme can be more feasible and practical for real-world applications.

Fig. 6 presents the storage and communication cost comparisons for each algorithm in related similar works (i.e. [23], [24], [27] and FTDS) under the different number of encrypted keywords and accessed indexes, where storage and communication cost refers to the storage cost to store the communicated parameter during the running time of each algorithm. For ease of comparisons, we also set the number of encrypted keywords (or searched indexes) l (or $|S|$) from 10 to 55 and the maximum number of files n from 10 to 100. As shown in Fig. 6(a), we can find that the storage and communication costs of Setup algorithm in these works are growing linearly with the maximum number of files, and compared to LLL+ [23], [24] and NLG+ [27], the storage and communication cost of Setup algorithm in our FTDS is higher than that in the works [23], [24], [27]. From Fig. 6(b), it is easily observed that the storage and communication costs of Extract algorithm in all these works are constant irrespective of the number of the searched indexes. Besides, the storage and communication cost of Extract algorithm in LLL+ [23], [24] is relatively lower than that in NLG+ [27] and our FTDS. From Fig. 6(c), it is easy to see that the storage and communication costs of Encrypt algorithm in these works are linear increasingly with the number of encrypted keywords. In addition, the storage and communication cost of Encrypt algorithm in our FTDS is slightly lower than that in NLG+ [27], but a bit higher than that in LLL+ [23], [24]. From Fig. 6(d), we can observe that, the storage and communication cost of Trapdoor algorithm in NLG+ [27] is linear increasingly with the number of encrypted keywords while the storage and communication costs of Trapdoor algorithm in LLL+ [23], [24] and our FTDS follow no relationship with the number of encrypted keywords. Besides, we can see that the storage and communication cost of Trapdoor algorithm in our FTDS

is almost the same as that in LLL+ [23], [24], but much lower than that in NLG+ [27]. By the observation of Fig. 6(e), we can conclude that the storage and communication costs of Retrieve algorithm in all these works follow a linear relationship with the number of encrypted keywords. Besides, we can find that the storage and communication cost of Retrieve algorithm in our FTDS is a bit higher than that in LLL+ [23], [24] but relatively lower than that in NLG+ [27]. From Fig. 6(f), it is easy to learn that the storage and communication costs of Decrypt algorithm in all these works are constant regardless of the number of the encrypted files. Besides, we can easily derive that storage and communication cost of Decrypt algorithm in our FTDS is indeed higher than that in other works [23], [24], [27]. Overall, our storage and communication costs are indeed relatively higher than that of other works [23], [24], [27]. However, the communication costs can still be desirable due to the fact the communication and storage costs are relatively low within a few MBytes.

According to the above analysis of computation and storage cost comparisons, we can learn that the computation and storage costs of decryption and trapdoor algorithm in our FTDS are always stable and relatively low, which makes our FTDS practical. This is because, for users, small constant computation and storage costs mean stably low energy consuming and storage space.

VIII. Conclusion

In this paper, we presented a practical blockchain-based data sharing system that is designed for the VSNs. Specifically, our FTDS system provides secure one-to-many key-aggregate encryption, data tampering resistance, verifiable keyword retrieval and constant search token (or aggregate secret key) generation capabilities for VSNs. We also showed the detailed security analysis and evaluation performance to demonstrate that the proposed system is secure and practical in actual applications. Future work includes finding solutions to solve trapdoor privacy leakage and enriching other features as required in various scenarios.

Acknowledgment

We would like to thank anonymous reviewers for their efforts and insight comments. This work was supported in part by the 13th Five-Year Plan of National Cryptography Development Fund for Cryptographic Theory of China under Grant MMJJ20170204. This work is also partially supported by the International Scientific and Technological Innovation Cooperation Project in Sichuan Province (2020YFH0062).

References

[1] Z. Ning, F. Xia, N. Ullah, et al., "Vehicular social networks: Enabling smart mobility", *IEEE Communications Magazine,* vol. 55, pp. 16-55, 2017.

[2] Y. Li, N. Kato, et al., "TSP security in intelligent and connected vehicles: Challenges and solutions", *IEEE Wireless Communications,* DOI: 10.1109/MWC.2019.1800289, IEEE, 2019.

[3] T. Reuters, "Connected Car Market Size, Share, Report, Analysis, Trends & Forecast to 2026", https://www.reuters.com/brandfeatures/venture-capital/article?id=37432.

[4] Y. Kawamoto, N. Kato, et al., "Toward Future Unmanned Aerial Vehicle Networks: Architecture, Resource Allocation and Field Experiments", *IEEE Wireless Communications,* vol. 26, no. 1, pp. 94-99, 2018.

[5] D. Chen, N. Zhang, R. Lu, et al., "An LDPC code based physical layer message authentication scheme with prefect security", *IEEE Journal on Selected Areas in Communications,* vol. 36, no. 4, pp. 748-761, 2018.

[6] J. Sun, H. Xiong, et al., "Mobile access and flexible search over encrypted cloud data in heterogeneous systems", *Information Sciences,* vol. 507, pp. 1-15, 2020.

[7] A. Rahim, X. Kong, F. Xia, Z. Ning, et al. ,"Vehicular social networks: A survey", *Pervasive and Mobile Computing,* vol. 43, pp. 96–113, 2018.

[8] A. Vegni and V. Loscri, "A survey on vehicular social networks", *IEEE Communications Surveys & Tutorials,* vol. 17, no. 4, pp. 2397–2419, 2015.

[9] J. Ning, N. Kato, et al., "Attacker Identification and Intrusion Detection for In-vehicle Networks", *IEEE Communications Letters,* DOI: 10.1109/LCOMM.2019.2937097, 2019.

[10] J. Sun, H. Xiong, X. Liu, et al., "Lightweight and Privacy-aware Fine-grained Access Control for IoT-oriented Smart Health", *IEEE Internet of Things,* vol. 7, no. 7, pp. 6566-6575, 2020.

[11] G. Xu, H. Li, S. Liu S, et al., "Efficient and Privacy-Preserving Truth Discovery in Mobile Crowd Sensing Systems", *IEEE Transactions on Vehicular Technology,* vol. 68, no. 4, pp. 3854-3865, 2019.

[12] X. Liu, K. Choo, R. Deng, et al., "Efficient and privacy-preserving outsourced calculation of rational numbers", *IEEE Transactions on Dependable and Secure Computing,* vol. 15, no. 1, pp. 27-39, 2016.

[13] G. Xu, H. Li, S. Liu, et al., "VerifyNet: Secure and Verifiable Federated Learning", *IEEE Transactions on Information Forensics and Security,* DOI: 10.1109/TIFS.2019.2929409, 2019.

[14] C. Chu, S. Chow, W. Tzeng et al., "Key-aggregate cryptosystem for scalable data sharing in cloud storage", *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 2, pp. 468-477, 2013.

[15] S. Patranabis, Y. Shrivastava, D. Mukhopadhyay, "Dynamic key-aggregate cryptosystem on elliptic curves for online data sharing", *International Conference on Cryptology in India*, Springer, Cham, LNCS 9462, pp. 25-44, 2015.

[16] S. Patranabis, Y. Shrivastava, D. Mukhopadhyay, "Provably secure key-aggregate cryptosystems with broadcast aggregate keys for online data sharing on the cloud", *IEEE Transactions on Computers,* vol. 66, no. 5, pp. 891-904, 2016.

[17] C. Guo, N. Luo, M. Bhuiyan, et al., "Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage", *Future Generation Computer Systems,* vol. 84, pp. 190-199, 2018.

[18] Q. Gan, X. Wang, D. Wu, "Revocable key-aggregate cryptosystem for data sharing in cloud", *Security and Communication Networks,* vol. 2017, 2017.

[19] Z. Wang, "Provably secure key-aggregate cryptosystems with auxiliary inputs for data sharing on the cloud", *Future Generation Computer Systems,* vol. 93, pp. 770-776, 2019.

[20] D. Boneh, G. Crescenzo, R. Ostrovsky, et al., "Public key encryption with keyword search", *Eurocrypt-2004,* Springer, LNCS 3027, pp. 506-522, 2004.

[21] B. Cui, Z. Liu, L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage", *IEEE Transactions on computers,* vol. 65, no. 8, pp. 2374-2385, 2015.

[22] Z. Liu, Y. Liu, "Verifiable and authenticated searchable encryption scheme with aggregate key in cloud storage", *International Conference on Computational Intelligence and Security (IEEE CIS)*, pp. 421-425, 2018.

[23] T. Li, Z. Liu, P. Li, et al., "Verifiable searchable encryption with aggregate keys for data sharing in outsourcing storage", *Australasian Conference on Information Security and Privacy*, Springer, Cham, LNCS 9723, pp. 153-169, 2016.

[24] Z. Liu, T. Li T, P. Li, et al., "Verifiable searchable encryption with aggregate keys for data sharing system", *Future Generation Computer Systems,* vol. 78, pp. 778-788, 2018.

[25] A. Kiayias, O. Oksuz, A. Russell, et al., "Efficient encrypted keyword search for multi-user data sharing", *European symposium on research in computer security,* Springer, Cham, LNCS 9878, pp. 173-195, 2016.

[26] R. Zhou, X. Zhang, X. Du, et al., "File-centric multi-key aggregate keyword searchable encryption for industrial internet of things", *IEEE Transactions on Industrial Informatics,* vol. 14, no. 8, pp. 3648-3658, 2018.

[27] J. Niu, X. Li, J. Gao, et al., "Blockchain-based Anti-Key-Leakage Key Aggregation Searchable Encryption for IoT", *IEEE Internet of Things Journal,* DOI: 10.1109/JIOT.2019.2956322, 2019.

[28] S. Nakamotoet, "Bitcoin: A peer-to-peer electronic cash system", 2008.

[29] M. Abdalla, M. Bellare, et al., "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions", *Crypto-2005,* Springer, LNCS 3621, pp. 205-222, 2005.
[30] J. Sun, S. Hu, X. Nie, et al., "Efficient Ranked Multi-Keyword Retrieval With Privacy Protection for Multiple Data Owners in Cloud Computing", *IEEE Systems Journal,* vol. 14, no. 2, pp. 1728-1739, 2020.
[31] Q. Zheng, S. Xu, G. Ateniese, "VABKS: verifiable attribute-based keyword search over outsourced encrypted data", *IEEE INFOCOM 2014*, IEEE, pp. 522-530, 2014.
[32] H. Cui, Z. Wan, R.H. Deng, et al., "Efficient and Expressive Keyword Search Over Encrypted Data in the Cloud", *IEEE Transactions on Dependable and Secure Computing,* vol. 15, no. 3, pp. 409-422, 2016.
[33] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," Self-Published Paper, August, vol. 19, 2012.
[34] "Bitshares-your share in the decentralized exchange." 2020. [Online]. Available: https://bitshares.org/.
[35] "Hyperledger project," 2020. [Online]. Available: https://www. hyperledger.org/.
[36] D. Schwartz, N. Youngs, and A. Britto,"The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, 2014.
[37] J. Kwon, "Tendermint: Consensus without mining", Draft v. 0.6, fall, vol. 1, no. 11, 2014.
[38] D. Johnson, A. Menezes, S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)", *International journal of information security*, vol. 1, no. 1, pp. 36-63, 2001.
[39] B. Lynn, JPBC library, Online: http://crypto. stanford. edu/jpbc, 2006.

Jianfei Sun is currently pursuing the Ph.D. degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China. His research interests include public key cryptography and network security. He was also a joint training Ph.D student under the supervisor Prof. Robert H. Deng with the School of Information and Systems, Singapore Management University.

Hu Xiong received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC) in 2009. He is currently a full professor with the School of Information and Software Engineering, UESTC. His research interests include applied cryptography and cypberspace security. He is a member of IEEE.

Shufan Zhang received the B. E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in the year of 2020. He is pursuing the M.Math degree with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, in the year of 2020. His research interests are data security and privacy, with current focus on differential privacy, applied cryptography with applications.

Ximeng Liu received Ph.D. degrees from Xidian University, China, in 2015. Now, he is a full professor at College of Mathematics and Computer Science, Fuzhou University, China. He has published over 100 research articles include IEEE TIFS, IEEE TDSC, IEEE TC, IEEE TII, IEEE TSC, IEEE TVT and IEEE TCC. His research interests include cloud security, applied cryptography and big data security.

Jiaming Yuan is currently pursuing the Master degree with the School of Information and Systems, Singapore Management University. His research interests include network security and Software Engineering.

Robert H. Deng is AXA Chair Professor of Cybersecurity and Professor of Information Systems in the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017). He is a fellow of the IEEE.