Information Sciences 635 (2023) 72–85

Contents lists available at ScienceDirect

Information Sciences

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

Privacy-preserving and efficient data sharing for blockchain-based intelligent transportation systems

Shan Jiang [a], Jiannong Cao [a,*], Hanqing Wu [a], Kongyang Chen [b,c,*], Xiulong Liu [d,*]

[a] *Department of Computing, The Hong Kong Polytechnic University, Hong Kong Special Administrative Region of China*
[b] *Institutes of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou, China*
[c] *Pazhou Lab, Guangzhou, China*
[d] *College of Intelligence and Computing, Tianjin University, Tianjin, China*

ARTICLE INFO

Keywords:
Blockchain
Intelligent transportation systems
Searchable symmetric encryption
Smart contract
Privacy preservation

ABSTRACT

Recent years have witnessed the development and adoption of blockchain technology in intelligent transportation systems (ITS) because of its authenticity and traceability. However, increasing ITS devices impose grand challenges in privacy-preserving and efficient data sharing. Recent research has demonstrated that integrating searchable symmetric encryption in blockchain enables privacy-preserving data sharing among ITS devices. However, existing solutions focus only on single-keyword searches over encrypted ITS data on the blockchain and suffer from privacy and efficiency issues when extended to multi-keyword scenarios. This work proposes a bloom filter-based multi-keyword search protocol for ITS data with enhanced efficiency and privacy preservation. We design a bloom filter to select a low-frequency keyword from the multiple keywords input by the ITS data owner. The low-frequency keyword can filter out a large portion of the ITS data from the search result, thus significantly reducing the computational cost. Furthermore, each identifier-keyword pair is attached with a pseudorandom tag that enables the completion of a search operation in only one round. In this manner, privacy is preserved because there are no intermediate rounds and results. In addition to the multi-keyword search protocol, we specify the addition and deletion protocols to enable dynamic updates of data records. We conducted a comprehensive performance evaluation of the protocols. The experimental results indicate that the proposed multi-keyword search protocol saves 14.67% query time and 59.96% financial cost.

1. Introduction

In the past decades, intelligent transportation systems (ITS) [46] have emerged, integrating the technologies of computer vision, sensing, wireless communication, etc., to reduce traffic accidents and improve transportation efficiency. The total market value of ITS is expected to hit 42.8 billion US dollars by 2028 [2]. In ITS, autonomous vehicles need to share a large amount of data with each other and the infrastructure (i.e., roadside units and base stations) to facilitate secure and fast decision-making [14]. For example, the incident information can be quickly delivered to the surrounding vehicles for accurate arrival time estimation as well as better

* Corresponding authors.
E-mail addresses: cs-shan.jiang@polyu.edu.hk (S. Jiang), jiannong.cao@polyu.edu.hk (J. Cao), hanqing.91.wu@connect.polyu.hk (H. Wu), kychen@gzhu.edu.cn (K. Chen), xiulong_liu@tju.edu.cn (X. Liu).

https://doi.org/10.1016/j.ins.2023.03.121
Received 18 September 2022; Received in revised form 10 March 2023; Accepted 19 March 2023

Available online 27 March 2023

path planning [40]. Moreover, the traffic lights can be carefully scheduled based on the road traffic information to maximize the traffic flow [45].

The traditional ITS data-sharing approaches can be categorized into data hosting-based [19] and data aggregation-based [32]. In data hosting-based approaches, the roadside units and base stations serve as the data hosting center storing data and responding to the search queries from the vehicles [30]. Because the upload data can be sensitive and numerous, such approaches incur privacy and efficiency concerns. Concerning data aggregation-based approaches, the vehicles only upload the metadata (descriptions) instead of the raw data to the roadside units and base stations [20]. In such circumstances, the existence of raw data corresponding to the metadata can hardly be ensured, resulting in authenticity issues.

The fundamental inadequacies of data hosting-based and data aggregation-based approaches lie in the reliance on a centralized server, either a roadside unit or a base station. To tackle the centralization issue, the research community has identified blockchain technology as a critical enabling solution for secure ITS data sharing [31]. More specifically, the vehicles, roadside units, and base stations are blockchain nodes and users simultaneously. They form a peer-to-peer network and maintain a blockchain. The data generated from vehicles are encrypted and securely stored on the blockchain. Moreover, these data can be queried by other vehicles, roadside units, and base stations via smart contracts in a privacy-preserving manner [28].

Searchable symmetric encryption is widely adopted as the encryption technique for blockchain-based ITS data-sharing system, where ITS data and search queries are encrypted, stored, and executed on blockchain [47]. On the one hand, the querier knows nearly nothing about the data, so privacy is preserved. On the other hand, the search queries are executed via smart contracts by the whole blockchain network, ensuring the soundness and completeness of the query results. Therefore, integrating blockchain and searchable symmetric encryption enables secure and privacy-preserving sharing among ITS devices.

Despite the significance, the state-of-the-art searchable encryption approaches over blockchains [48,3] merely consider a single keyword. Indeed, the approaches can be extended to serve multiple keywords by undertaking the keywords individually and computing the intersection [17]. However, such extensions intrinsically suffer from security and efficiency issues. In particular, the intermediate results, i.e., the ITS data associated with each keyword, will be leaked to the blockchain nodes. Such data leakage raises security and privacy concerns [12]. Moreover, the blockchain nodes must proceed with the single-keyword search queries sequentially. Note that some keywords are associated with most and even all the identifiers. Therefore, dealing with the numerous keywords incurs a relatively high computation cost and time overhead. The intermediate results need to be stored in the search smart contract, resulting in a high financial cost.

In this work, we design a secure and efficient ITS data-sharing system with database setup, dynamic update, and multi-keyword search services. We regard the offloaded database as multiple pairs of identifiers and keywords. That is, there are several keywords associated with each of the identifiers. After setup, the ITS data owner can update the data by dynamically adding or deleting specific identifier-keyword pairs. Meanwhile, the ITS data owners can perform multi-keyword search queries to find the identifiers linked to some given keywords. The multi-keyword search queries can be executed with a low time and financial overhead owing to the novel bloom filter-enabled approach proposed in this work. The ITS devices act as blockchain nodes that hire data and computation resources to reap economic rewards. The protocols of database setup, dynamic update, and multi-keyword search are implemented with the help of the smart contracts running on the blockchain.

The main contributions of this work are as follows:

- We propose a bloom filter-enabled multi-keyword search protocol for blockchain-based ITS data sharing, significantly saving the time overhead and financial cost.
- We design a cost estimation method for various blockchain operations so that the cost limit role in smart contracts will not be violated.
- We conduct extensive experiments in real-world ITS datasets, validating the proposed protocols' practicability and effectiveness.

The remainder of this work is organized as follows. Sec. 2 summarizes the related work and articulates the motivations of this work. Sec. 3 elaborates on the system design and proposed protocols of setup, addition, deletion, and multi-keyword search. Sec. 4 shows the experimental settings and extensive performance evaluation results regarding time overhead and financial cost. Finally, Sec. 5 concludes this work and suggests future research directions.

2. Related work

This section presents the related work of blockchain-based ITS, data sharing in ITS, and high-performance search over blockchains. We show that secure and efficient data sharing for blockchain-based ITS is important and inadequately addressed.

2.1. Blockchain-based intelligent transportation systems

With the popularity of blockchain technology, its applications in intelligent transportation systems have been attracting intensive attention from academia and industry. There are a bunch of survey papers about blockchain-enabled intelligent transportation systems [43,31,7]. As early as 2016, Yuan and Wang conducted a pioneering study of blockchain-based ITS and designed an ITS-oriented seven-layer blockchain conceptual model [43]. Mollah et al. presented the first comprehensive survey of blockchain-based ITS, covering the applications, architectures and frameworks, challenges, and future research opportunities [31]. More recently, Dibaei et al. focused on the security of vehicular networks and surveyed the blockchain and machine learning-based approaches to tackle the security issues [7].

The research community has identified a bunch of critical applications of blockchain-based ITS, including data protection and trading, resource sharing, content broadcasting, and traffic control [13]. Many challenging issues arise and remain to be addressed, such as incentive mechanism design [28,49] and high-performance consensus [26]. For example, Li et al. proposed CreditCoin facilitating content broadcasting [28]. In particular, CreditCoin is a blockchain-based ITS incentivizing vehicles to share surrounding traffic information while guaranteeing its anonymity and authenticity.

Besides the challenges above, security and privacy are also significant challenges of blockchain-based ITS because insecurity leads to accidents and even fatalities, and privacy leakage makes data owners lose great value [34]. For example, Lei et al. discussed the limitations of traditional authentication mechanisms in cloud-based ITS systems. They introduced a blockchain-based solution for vehicle authentication as well as key exchange [27]. However, current work mainly considers the security of vehicles from the perspective of communication or authentication [27].

2.2. Data sharing in intelligent transportation systems

Data sharing is essential for ITS because it enriches individual vehicles' knowledge, achieving higher road safety and better path planning [35]. Most existing data-sharing approaches derive from the requirement of ITS applications, e.g., content broadcasting and dissemination [22] and vehicular mobile social networks [36]. For example, Ko et al. proposed employing vehicle-to-vehicle data sharing to facilitate the roadside units to schedule data services in bidirectional road scenarios [22]. Sun et al. developed secure and flexible vehicular social networks by designing a blockchain-based tampering-resistant data sharing mechanism [36]. Similarly, Kong et al. proposed an effective and efficient approach to allowing vehicles to share multi-dimensional sensory data with privacy preservation [23]. Furthermore, Gosman et al. considered the trustworthiness of information shared by vehicles and devised an approach to quality assessment concerning spatial accuracy, temporal closeness, and vehicle reputation [10].

Recently, emerging studies have considered the challenges of ITS data sharing, including security, efficiency, and flexibility. Sun et al. proposed a flexible cross-domain authorization mechanism for secure ITS big data sharing [37]. In particular, a ciphertext conversion technique was designed to achieve high-fidelity data communication between two vehicles in different platoons. Regarding ITS data sharing, blockchain is a promising solution due to its distinctive properties of security and decentralization, and there are many related studies [6]. Kang et al. designed a blockchain-based edge computing architecture integrating autonomous vehicles, roadside units, and base stations, in which the vehicles can share data securely and efficiently [20]. To the best of our knowledge, it is the first to consider blockchain-based ITS data sharing. Javaid et al. further considered trust management in blockchain-based ITS inter- and intra-network communication [16]. Cui et al. reduced the dependence on roadside units and enhanced the traceability and anonymity during ITS data sharing [5].

2.3. High-performance search over blockchain

Searching over a blockchain has been a hot topic in recent years [41]. The naive approach is that the user sends a search query to a blockchain full node, and the full node proceeds block by block and transaction by transaction. However, such an approach incurs low efficiency, integrity, and efficiency. Notably, it is time-consuming to scan the whole blockchain, a single full node can be malicious, and the data and search queries are transparent to the public. To this end, the researchers have been designing blockchain search protocols guaranteeing integrity and privacy and enhancing efficiency.

Smart contracts and verifiable computation are two mainstream approaches to enhancing search integrity. In smart contract-based methods, the requests from users are executed by all the blockchain nodes rather than a single one [17]. As long as a blockchain system is secure, the search results will be honorable because most blockchain nodes agree on the results. Such methods are convenient to be adapted to all kinds of blockchain data but are time- and computation-intensive. Regarding verifiable computation, the full node returns not only the results but proof as well [42]. The returned proof is essential to verify the search results' integrity, i.e., soundness and completeness. Such an approach enjoys high efficiency owing to the design of subtle data structure for verification [44]. However, no universal data structure can handle every kind of data.

Searchable symmetric encryption is one of the emerging techniques concerning privacy protection [15,38,11]. The data, search queries, and results are encrypted throughout the search procedures over blockchain so that the users know nearly nothing about the data stored by the owners. Hu et al. proposed the first protocol integrating searchable symmetric encryption in blockchain search, supporting keyword-based search queries [15]. Cai et al. further devised a fair protocol to handle disputes and issue fair payments between data users and owners [3]. Chen et al. integrated blockchain and searchable encryption technologies for vehicular mobile social networks comprehensively considering both the backward and forward privacy [4]. The techniques are widely employed in sharing electronic health records [4,18,33]. Searchable symmetric encryption can also be implemented in smart contracts providing privacy and integrity simultaneously [15,17].

In a nutshell, existing studies have considered data sharing a critical application of blockchain-based ITS. However, they only focus on real-world implementation or are conceptual works while neglecting security and efficiency issues.

3. Secure and efficient ITS data sharing

In this section, we present the overall system design of the proposed ITS data sharing scheme in Sec. 3.1 and explain the detailed protocols of database setup, dynamic update, and multi-keyword search in Sec. 3.2, Sec. 3.3, and Sec. 3.4, respectively.

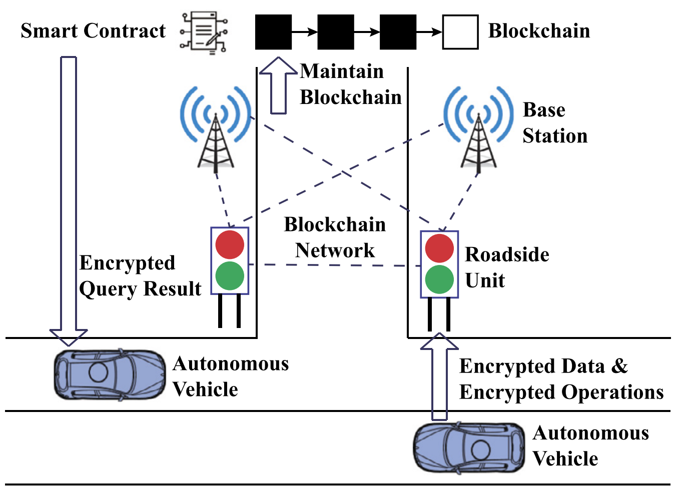

Fig. 1. System overview of the blockchain-based ITS data-sharing system. On the one hand, autonomous vehicles enjoy privacy-preserving and efficient data-sharing services by sending the encrypted data and operations to and receiving encrypted query results from the roadside infrastructure. On the other hand, the roadside infrastructure, including roadside units and base stations, maintain a blockchain and responds to the vehicles' operations by invoking pre-defined smart contracts.

3.1. System overview

As shown in Fig. 1, there are three entities in the blockchain-enabled ITS data management and sharing system, i.e., *autonomous vehicles*, *roadside units*, and *base stations*. The roadside units and base stations are the service providers which form a blockchain network and rent out computational resources to provide data-sharing services and earn monetary benefits. Autonomous vehicles are service users who aim to offload their TIS data and enjoy content update and search services.

There are four kinds of actions that can happen between the blockchain nodes and data owners, i.e., *setup*, *addition*, *deletion*, and *search*. The formal definitions of the four actions are as follows.

- *Setup*. The ITS data owner offloads a database $W = \{(id_i, w_i) \mid i = 1, 2, \cdots, l\}$, consisting of a sequence of l identifier-keyword pairs to the blockchain nodes. Inside, $id_i \in \{0,1\}^{\mu}$ are the identifiers represented in strings with a certain length; $w_i \in \{0,1\}^*$ are the keywords represented in strings with a uncertain length; (id_i, w_i) means the identifier id_i is associated with the keyword w_i. In total, there are n identifiers, m keywords, and l identifier-keyword pairs.
- *Addition*. The ITS data owner aims to update the offloaded database W by associating a set of keywords W_a to an identifier id.
- *Deletion*. The ITS data owner aims to update the offloaded database W by disassociating a set of keywords W_a from an identifier id.
- *Search*. The ITS data owner sends $W_s = \{w_1, w_2, \cdots, w_k\}$ to the blockchain nodes to obtain all the identifiers id s such that there exists $w \in W_s$ and $(id, w) \in W$.

The flowchart of the actions is demonstrated in Fig. 1. When autonomous vehicles, i.e., ITS data owners, conduct specific operations, they run the protocols at the side of ITS data owners to pack the operations and encrypted data into several blockchain transactions and send them to the roadside infrastructure. As for the roadside infrastructure, i.e., blockchain nodes, they run the protocols at the side of blockchain nodes to proceed with the transactions through predefined smart contracts and confirm the transactions to include them in the blockchain. To this end, the ITS data owners can obtain the search results by looking up the states in the smart contracts. The system is designed so for several reasons. On the one hand, autonomous vehicles are generally mobile and equipped with limited resources, making themselves unsuitable for maintaining blockchain and running smart contracts. To this end, the vehicles are designed to perform affordable data encryption tasks only. On the other hand, the roadside units and base stations are immobile and can afford computation-intensive blockchain tasks.

The security model affects the design of communication protocols. Generally, a strict security model fits the real world more appropriately while requiring a more subtle protocol design. Our security model is strict because we assume both the blockchain nodes and the ITS data owners in our system model can be malicious. In particular, the security model in this work is as follows.

- *Standard public blockchain model*. We follow the standard and most popular blockchain threat model [24] in this work. More specifically, the blockchain nodes accept transactions from the ITS data owners and make a consensus on validating blocks. The blockchain nodes together form the blockchain network. The whole blockchain system works correctly as long as there are no more than 50% malicious blockchain nodes [39].
- *Dishonest data owner*. The ITS data owner may refuse to pay the fee after enjoying the data services of setup, addition, deletion, and search from the blockchain nodes.

Algorithm 1 Smart contract initialization protocol at blockchain nodes.

```
1: Allocate a dictionary $D_{ori}$    ▷ To store the encrypted keyword-identifier pairs
2: Allocate two sets $S_{del}$ and $S_{tag}$    ▷ $S_{del}$ is to cache the deleted identifier-keyword pairs and $S_{tag}$ is to store all the identifier-keyword pairs
3: Allocate two lists $Flag$ and $Result$    ▷ $Flag$ is to facilitate the addition protocol and $Result$ is to store the keyword search result
4: Set the balance as $B$, which is the money deposited by ITS data owner
```

Algorithm 2 Setup protocol at ITS data owner on storing W.

```
1: $K, K^+, K^-, K^T \leftarrow \{0,1\}^\lambda$    ▷ Four secret keys of the same and fixed length $\lambda$ for data encryption
2: Allocate two lists $L$ and $L_{tag}$, and a counter dictionary $D_{cntr}$    ▷ The two lists will be offloaded while the dictionary is for local usage
3: $count \leftarrow 0$    ▷ The counter of the accumulated transactions
4: for each keyword $w \in W$ do    ▷ Iterate over the keywords
5:     $K_1||K_2 \leftarrow F(K, w)$    ▷ Generate keyword-specific keys using the secret key $K$
6:     $K_w^T \leftarrow F(K^T, w)$    ▷ Generate the key to encrypt the keyword-identifier pairs
7:     $c \leftarrow 0$    ▷ The counter of the identifiers associated with the keyword $w$
8:     for each $id \in W_w$ do    ▷ Iterate over the identifiers associated with $w$
9:         $l \leftarrow F(K_1, c)$    ▷ Encrypt the counter as the encrypted keyword
10:        $d \leftarrow Enc(K_2, id)$    ▷ Encrypt the identifier
11:        $c \leftarrow c + 1$    ▷ Increment $c$ by 1
12:        $tag \leftarrow F(K_w^T, id)$    ▷ Encrypt the identifier-keyword pair
13:        Append $(l, d)$ to $L$    ▷ The pair $(l, d)$ will be offloaded
14:        Append $tag$ to $L_{tag}$    ▷ The $tag$ will be offloaded
15:        $count \leftarrow count + 1$    ▷ Increment $count$ by 1
16:        if $count \geq \mu$ then    ▷ Pack the data into a transaction if its volume reaches the limit
17:            Shuffle $L$ and $L_{tag}$ randomly    ▷ Shuffle the lists to avoid data leakage
18:            Send $(\text{SETUP}, L, L_{tag})$ to the blockchain nodes    ▷ Send out the transaction
19:            $count \leftarrow 0$    ▷ Reset the counter $count$
20:            Empty $L$ and $L_{tag}$
21:        end if
22:    end for
23:    $Set(D_{cntr}, w, c)$    ▷ Set the next counter of keyword $w$ as $c$
24: end for
25: Send $(\text{SETUP}, L, L_{tag})$ to the blockchain nodes    ▷ Send out the remaining data
26: Sort the keywords $w \in W$ with $Get(D_{cntr}, w)$ in a non-increasing order to get a list of keywords $W_s$
27: Allocate a list $bf$ of $\alpha$ 0/1 bits initialized to be all 0    ▷ Initialize the bloom filter with $\alpha$ bits
28: for each keyword $w$ in the first $\beta$ percentage of $W_s$ do    ▷ Iterate over the high-frequency keywords
29:     $bf \leftarrow (\mathcal{H}(w) \mid bf)$    ▷ Update the bloom filter using each high-frequency keyword
30: end for
31: Store $K, K^+, K^-, K^T, bf$, and $D_{cntr}$ locally    ▷ Keep the secrete keys, bloom filter, and counter dictionary for local usage
```

We design privacy-preserving and efficient protocols based on the security model to fulfill the four actions described in the following subsections. Before elaborating on the actions, we first explain the steps of initializing the smart contract as shown in Algo. 1. The smart contract stores five variables, which are a dictionary D_{ori}, two sets S_{del} and S_{tag}, and two lists $Flag$ and $Result$. Inside, D_{ori} keeps the encrypted pairs of keywords and identifiers, S_{del} and $Flag$ facilitate high-efficiency addition and deletion of the data records, S_{tag} sustains the storage of search-related tags, and $Result$ stores the results of search queries. Finally, the ITS data owner has a surplus of B units of tokens in the smart contract. The operations can cost no more than B in the future, which we assume is true in this work.

3.2. Database setup

Algo. 2 elaborates on the steps the ITS data owner needs to take to set up and offload the database. The ITS data owner targets to store numerous identifier-keyword pairs on the blockchain. First, the ITS data owner generates four secret keys K, K^+, K^-, and K^T, all of the size λ, an adjustable security parameter. Specifically, K is for the encryption of the identifiers and keywords, K^+ is to support the addition operation, K^- is to support the deletion operation, and K^T is to encrypt the keyword-identifier pairs.

We leverage a pseudorandom function (PRF) [9] F and the secret key K to generate two keyword-specific keys K_1 and K_2 for each keyword w. In particular, the key K_1 takes the counter of identifiers associated with the keyword w to generate the pseudorandom labels of the keywords. The key K_2 symmetrically encrypts the identifiers. Given a particular keyword w, we can quickly compute the set of identifiers with w as one of the keywords. The set is annotated as W_w. Afterward, we iterate the identifiers id over W_w and automatically increment the counter c starting from 0. To this end, a sequence of keywords (w), identifiers $(id \in W_w)$, and counters c are obtained. On the one hand, we apply the PRF F for the counter c using the key F_1 to get a pseudorandom label l black representing the encrypted keyword. On the other hand, K_2 is utilized to symmetrically encrypt the identifier id to obtain the encrypted identifier d. Then, the encrypted keyword-identifier pair is pushed into the list L. In addition, we sequentially take w and id as input to use the tag key K^T and PRF F to get a unique tag for each keyword-identifier pair. The tags are accumulated into the list L_{tag}. The tags help boost the efficiency of search queries over multiple keywords. In a nutshell, we generate an encrypted keyword-identifier pair and a unique tag for each identifier-keyword pair leveraging the secret keys K and K^T.

The ITS data owner sends the encrypted data, i.e., L and L_tag, to the blockchain nodes. However, the sizes of the two lists can be too large to be contained in a single transaction because of the limit of transaction size and smart contract cost. In this work, we

Algorithm 3 Setup protocol at blockchain nodes on receiving $(\textsc{Setup}, L, L_{tag})$.

```
1: Add all the elements $(l_i, d_i)$ in $L$ to the dictionary $D_{ori}$ with $l$ as the key and $d$ as the value
2: Add all the elements in $L_{tag}$ to the set $S_{tag}$
```

Algorithm 4 Addition protocol at ITS data owner on adding (id, W_a) to W.

```
1: Allocate two lists $L$ and $L_{tag}$    ▷ The two lists will be offloaded
2: for each $w \in W_a$ do    ▷ Iterate over the keywords to be associated with the identifier $id$
3:     $K_1^+||K_2^+ \leftarrow F(K^+, w)$    ▷ Encrypt the keyword $w$ using the addition-related secrete key $K^+$
4:     $c \leftarrow Get(D_{cntr}, w)$    ▷ Get the next counter of the keyword $w$ from the dictionary $D_{cntr}$
5:     if $c = \perp$ then $c \leftarrow 0$ end if    ▷ If there is no record, initialize the counter as 0
6:     $l \leftarrow F(K_1^+, c)$    ▷ Encrypt the counter as the encrypted keyword
7:     $d \leftarrow Enc(K_2^+, id)$    ▷ Encrypt the identifier
8:     $delid \leftarrow F(F(K^-, w), id)$    ▷ Encrypt the keyword-identifier pair using the deletion secret key $K^-$
9:     $tag \leftarrow F(F(K^T, w), id)$    ▷ Encrypt the keyword-identifier pair using the tag secret key $K^T$
10:    Append $(l, d, delid)$ to $L$    ▷ The tuple $(l, d, delid)$ will be offload
11:    Append $tag$ to $L_{tag}$    ▷ The $tag$ will be offloaded
12: end for
13: Shuffle $L$ and $L_{tag}$ randomly    ▷ Shuffle the lists to avoid data leakage
14: Send $(\textsc{Add}, L, L_{tag})$ to the blockchain nodes    ▷ Send out the transaction
15: Wait for value change of $Flag$    ▷ Wait for the execution of the addition smart contract
16: $i \leftarrow 1$
17: for each $w \in W_a$ do    ▷ Iterate over the keyword to be associated with the identifier $id$
18:    if the $i$-th bit of $Flag$ is 0 then    ▷ It means the keyword is not cached for deletion
19:        $c \leftarrow Get(D_{cntr}, w) + 1$    ▷ Get the next counter of the keyword $w$ using the dictionary $D_{cntr}$
20:        $Set(D_{cntr}, w, c)$    ▷ Update the next counter of the keyword $w$
21:    end if
22:    $i \leftarrow i + 1$
23: end for
```

propose to slice the lists into pieces if they exceed the limit. More specifically, in the database setup protocol, the data arising from each identifier-keyword pair is deterministic, i.e., only an encrypted keyword-identifier pair and a tag. Furthermore, the volume of the encrypted keyword-identifier pair and tag are regular, given a PRF. For example, if the PRF is HMAC-SHA256 [25], each keyword-identifier pair will result in an encrypted keyword-identifier pair in 512 bits and a tag in 256 bits.

Therefore, we can compute the computation capacity of every transaction, i.e., the number μ of identifier-keyword pairs in a transaction. Suppose that the PRF F digests any amount of data outputting μ_f bits, and every single transaction can store data with at most μ_t bits. Then, the value of μ should be $\lfloor \mu_t/(3\mu_f) \rfloor$. In this work, we set the data volume limit of each transaction as 10 KB ($\sim$ 8000 bits) and employ HMAC-SHA256 as the PRF, i.e., μ_t equals $80{,}000$, μ_f equals 256, and μ is calculated to be 104. When the counter reaches the limit μ, we shuffle the lists L and L_{tag} and send a transaction of setup containing them to the blockchain nodes. Shuffling L and L_{tag} is to prevent the blockchain nodes from inferring any information related to the data. For instance, L and L_{tag} are in the same order concerning each identifier-keyword pair. Once a transaction is sent to blockchain nodes, the counter $count$ is reset as 0, and the two lists are cleared. Then, we will sort the keywords according to their appearance times in the database to initialize the bloom filter. It will be described in Sec. 3.4 in detail.

Regarding the blockchain nodes, they receive a sequence of transactions of setup operations accompanied by two lists L and L_{tag}. Algo. 3 shows how the blockchain nodes deal with the transactions. Regarding each transaction, blockchain nodes iterate the items (l_i, d_i) in L and add elements into D_{ori} with l_i and d_i as the key and value. Then, the blockchain nodes copy all the variables in L_{tag} into the set S_{tag}. It is noticeable that the data storage in the smart contract is enabled with dictionaries and sets, which are *unordered* data structures. It implies that the database setup protocol can resist the attacks leveraging the transaction order in nature.

3.3. Dynamic update

After the database setup, the ITS data owner can update the data records from time to time. In our protocol, we consider two update operations, i.e., addition and deletion. The smart contract keeps S_{del}, a set of keyword-identifier pairs cached for deletion. In each update operation, the ITS data owner will inform the blockchain nodes of the way to update S_{del}. The addition and deletion protocols are presented in Algo. 4, Algo. 5, Algo. 6, and Algo. 7.

In the addition protocol, the ITS data owner aims to add a set of keywords W_a to an identifier id. Note that in Algo. 2, the ITS data owner increases a dictionary D_{cntr} to maintain the latest counter of each keyword, i.e., the number of identifiers associated with each keyword. Algo. 4 iterates over the keywords $w \in W_a$ to be associated with id. The ITS data owner fetches the counter c of the keyword w. The counter will be encrypted as l representing the encrypted keyword. Regarding the identifier id, it will be symmetrically encrypted to d. The encryption of the keywords and identifiers uses the addition key K^+. Furthermore, $delid$ and tag will be generated using the deletion key K^- and tag key K^T to go through the keyword w and identifier id, respectively. To this end, each keyword to be associated with the identifier will generate a tuple of l, d, $delid$, and tag. We append the tuple of $(l, d, delid)$ s to the list L and the tag s to the list L_{tag}. The ITS data owner will send the two lists L and L_{tag} to the blockchain nodes for the addition of data records. Here, we assume for simplicity that a single transaction is large enough for the two lists. If too many elements are

Algorithm 5 Addition protocol at blockchain nodes on receiving $(\textsc{Add}, L, L_{tag})$.

```
1: Add all the elements in $L_{tag}$ to $S_{tag}$    ▷ Add all the keyword-identifier pairs
2: Allocate a list $F$ of $|L|$ bits    ▷ A temporary variable that will be assigned to $Flag$ in the end
3: $i \leftarrow 1$
4: for each $(l, d, delid) \in L$ do    ▷ Iterate over the list $L$
5:     if $delid \in S_{del}$ then    ▷ The keyword-identifier pair was deleted and cached
6:         Set the $i$-th bit of $F$ to be 1    ▷ Mark the bit as cached for deletion
7:         $S_{del} \leftarrow S_{del} \setminus \{delid\}$    ▷ Remove the keyword-identifier pair from the set $S_{del}$
8:     else
9:         Set the $i$-th bit of $F$ to be 0    ▷ Mark the bit as not cached for deletion
10:        $D_{ori} \leftarrow D_{ori} \cup \{(l, d)\}$    ▷ Update the keyword-identifier dictionary $D_{ori}$
11:    end if
12:    $i \leftarrow i + 1$
13: end for
14: Flag ← F    ▷ Assign $F$ to $Flag$ to notify the completion of the addition smart contract
```

Algorithm 6 Deletion protocol at ITS data owner on deleting (id, W_d) from W.

```
1: Allocate two lists $L_{del}$ and $L_{tag}$    ▷ The two lists will be offloaded
2: for each $w \in W_d$ do    ▷ Iterate over the keywords to be disassociated from the identifier $id$
3:     $delid \leftarrow F(F(K^-, w), id)$    ▷ Encrypt the keyword-identifier pair using the deletion secret key $K^-$
4:     $tag \leftarrow F(F(K^T, w), id)$    ▷ Encrypt the keyword-identifier pair using the tag secret key $K^T$
5:     Append $delid$ to $L_{del}$    ▷ The $delid$ will be offloaded
6:     Append $tag$ to $L_{tag}$    ▷ The $tag$ will be offloaded
7: end for
8: Shuffle $L_{del}$ and $L_{tag}$ randomly    ▷ Shuffle the lists to avoid data leakage
9: Send $(\textsc{Delete}, L_{del}, L_{tag})$ to the blockchain nodes    ▷ Send out the transaction
```

Algorithm 7 Deletion protocol at blockchain nodes on receiving $(\textsc{Delete}, L_{del}, L_{tag})$.

```
1: Add all the elements in $L_{del}$ to $S_{del}$
2: Remove all the elements in $L_{tag}$ from $S_{tag}$
```

added, we can split L and L_{tag} similar to Algo. 2. The same assumption is made for the deletion protocol as well. Note that the lists L and L_{tag} are shuffled before sending to the blockchain nodes to take care of privacy preservation.

Algo. 5 depicts the steps the blockchain nodes need to take when receiving an addition operation together with L and L_{tag}. They proceed as follows. First of all, the nodes insert the elements received in L_{tag} to the dictionary S_{tag} of the smart contract. Regarding each element $(l, d, delid)$ in L_{tag}, the nodes will examine if the set S_{del} contains the element $delid$. If yes, then (l, d) is a cached element for deletion and is supposed to be inserted back. In this circumstance, the element $delid$ will be deleted from the deletion set S_{del}, meaning the revocation of deleting (l, d). Otherwise, $delid$ is not in the set S_{del}. It means (l, d) is a newly added keyword-identifier pair for the database. Under this circumstance, a new data record with l as the key and d as the value is inserted into the dictionary D_{ori}. Meanwhile, the blockchain nodes will notify the ITS data owner to renew the dictionary D_{cntr}. More specifically, the blockchain nodes save a list of $|L|$ 0/1 bits to the variable $Flag$ in the smart contract, in which 0 means (l, d) is a new keyword-identifier pair and 1 means (l, d) is previously deleted.

The data owner will update the counter dictionary D_{cntr} when the change of $Flag$ in the smart contract is observed. Each 0 bit in $Flag$ means the keyword w is associated with a newly added keyword-identifier pair. In this case, the data owner will fetch the counter of w in D_{cntr}, increase the counter by 1, and add the keyword-counter pair to D_{cntr}.

In the deletion protocol, the ITS data owner targets to disassociate a set of keywords W_d from an identifier id. The protocol of deletion is more concise than the one of addition. In particular, the data owner calculates $delid$ using the deletion key K^- and tag using the tag key K^T for each $(id, w) \in W_d$. Then, the ITS data owner packs the $delid$ s into the deletion list L_{del} and the tag s into the tag list L_{tag}. The two lists L_{del} and L_{tag} will be shuffled randomly and sent to the blockchain nodes for deletion purposes.

On receiving L_{del} and S_{del} from the data owner, the blockchain nodes add all the elements in L_{del} to S_{del} and remove all the elements in L_{tag} from S_{tag}. Note that S_{del} is to cache the deletion of the identifier-keyword pair rather than direct execution to save time. As for S_{tag}, it stores the pseudorandom labels of all the identifier-keyword pairs to facilitate the process of multi-keyword search.

3.4. Multi-keyword search

The above protocols enable the ITS data owners to store and update the data records in a dynamic, secure, and privacy-preserving manner. In the following, we present the protocol of multi-keyword search to enable the ITS data owners to query over the encrypted data records.

Fig. 2 depicts how the proposed multi-keyword search protocol works. The k keywords submitted by the ITS data owner will go through a bloom filter, outputting a low-frequency keyword w_1 and $k-1$ other keywords $w_2, w_3, \cdots, w_k$. Note that w_1 is "low-frequency" if it "rarely" appears in the database W. Then, the protocol performs a single-keyword search to filter the whole database W using w_1. Because w_1 is low-frequency, the output set containing candidate identifiers will be small in size. The candidate

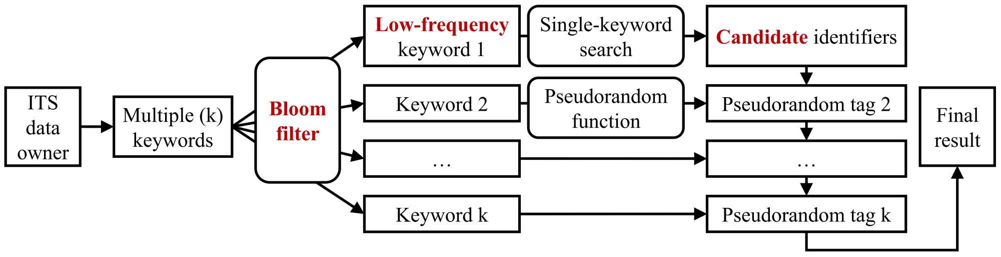

Fig. 2. Flowchart of bloom filter-enabled search protocol. First, a low-frequency keyword is selected from the k submitted keywords using a bloom filter. Then, the whole database is filtered using the low-frequency keyword to get a set of candidate identifiers. Finally, each candidate identifier is examined by the other $k-1$ keywords one by one to compute the final results.

Algorithm 8 Search protocol at ITS data owner on searching $(w_1, w_2, \cdots, w_k)$ in W.

```
1: for $i \leftarrow 1$ to $k$ do                    ▷ Iterate over the indexes of the keywords in the search query
2:     $h \leftarrow \mathcal{H}(w_i)$                ▷ Hash the keyword $w_i$
3:     if $(h \& bf) \neq h$ then                     ▷ Check whether $h$ is in the bloom filter
4:         Swap $w_1$ and $w_i$                       ▷ If yes, then $w_i$ is a low-frequency keyword; switch the found low-frequency keyword $w_i$ with the first one
5:         Break                                      ▷ A low-frequency keyword is found, so break the for-loop
6:     end if
7: end for
8: $K_1||K_2 \leftarrow F(K, w_1)$                    ▷ Encrypt the low-frequency keyword using the secrete key $K$
9: $K_1^+||K_2^+ \leftarrow F(K^+, w_1)$              ▷ Encrypt the low-frequency keyword using the addition secrete key $K^+$
10: $K_1^- \leftarrow F(K^-, w_1)$                   ▷ Encrypt the low-frequency keyword using the deletion secrete key $K^-$
11: for $i \leftarrow 2$ to $k$ do                   ▷ Iterate over the other keywords
12:     $K_i^T \leftarrow F(K^T, w_i)$                ▷ Encrypt the other keywords using the tag secret key $K^T$
13: end for
14: Send $(\textsc{Search}, K_1, K_2, K_1^+, K_2^+, K_1^-, K_2^T, \cdots, K_k^T)$ to the blockchain nodes    ▷ Send out the search transaction
```

identifiers are checked through $w_2, w_3, \cdots, w_k$ in sequence using pseudorandom tags to obtain the final result. Note that the checking will not incur high overhead due to the small number of candidate identifiers.

First of all, we introduce the concept of high-frequency keywords. In Algo. 2, a dictionary D_{cntr} is declared to count the number of identifiers associated with each keyword, which is the appearance time defined as follows:

$$F(w, W) = |\{id \mid (id, w) \in W\}|$$

The set $\{id \mid (id, w) \in W\}$ represents all the identifiers associated with the keyword w. The appearance time of w in W, i.e., $F(w, W)$, is the cardinality of the set. We define that a keyword $w \in W$ is high-frequency if after sorting $\{w \mid w \in W\}$ in a non-increasing order according to $F(w, W)$, the keyword w appears in the first β percentage. A keyword is defined as low-frequency in W if it is not high-frequency in W.

Second, we present the bloom filter method to find the low-frequency keywords. On the opposite side, we construct the bloom filter bf for high-frequency keywords because the amount of high-frequency keywords is relatively small. The bloom filter is initialized as an α-bit string with 0 s. A hash function $\mathcal{H}$ is applied for each high-frequency keyword w to get an α-bit 0/1 string, i.e., $\mathcal{H}(w)$. We perform bitwise OR operation between bf and $\mathcal{H}(w)$ to update bf. After processing all the low-frequency keywords, bf will be a bloom filter that can filter high-frequency keywords. Note that α and β are parameters that should be fine-tuned to make the bloom filter efficient. In particular, when α is large, the storage overhead will increase from the perspective of the ITS data owners. In contrast, it will decrease the false positive ratio of asserting high-frequency keywords. Regarding β, a large value of β increases the false positive rate while reducing the true positive cost. In this work, we set α and β to be $8,000$ and 10%, respectively, which is enough to handle a database of up to 9.1 M identifier-keyword pairs.

As shown in Algo. 8, the bloom filter can be used to efficiently find out a low-frequency keyword in the multi-keyword search query. More specifically, for a keyword w, if its hash value, i.e., $\mathcal{H}(w)$ cannot pass the bloom filter, i.e., $\mathcal{H}(w) \,\&\, bf \neq \mathcal{H}(w)$, then we can make sure that w is low-frequency. We switch the places of the found low-frequency keyword and the first keyword in the multi-keyword search query.

Now, it comes to the phase of generating encrypted search queries by the data owner. The ITS data owner leverages the regular key K, addition key K^+, and deletion key K^- to generate three pseudorandom labels, i.e., regular label K_1, addition label K_1^+, and deletion label K_1^- and two symmetric keys, i.e., regular key K_2 and addition key K_2^+ for the first keyword w_1 that is of low frequency. In addition, a pseudorandom tag will be calculated for each other keywords using the tag key K^T. To this end, an encrypted search query containing three pseudorandom labels, two symmetric keys, and $k-1$ tags is successfully composed and sent to the blockchain nodes.

On receiving a search query from the data owner, the blockchain nodes will start with the first keyword and get a set of candidate identifiers as in Algo. 9. More specifically, the nodes go through D_{ori} using the pseudorandom keys, i.e., regular key K_1 and addition key K_1^+. Then, the blockchain nodes accumulate all the counters in the key field of D_{ori} after encryption using K_1 or K_1^+. Furthermore, the blockchain nodes add the identifiers after decryption corresponding to each accumulated counter using the corresponding

Algorithm 9 Search protocol at blockchain nodes on receiving ($\textsc{Search}, K_1, K_2, K_1^+, K_2^+, K_1^-, K_2^T, \cdots, K_k^T$).

```
1:  $res \leftarrow \emptyset$                                  ▷ A temporary variable that stores the search result
2:  for $c \leftarrow 0$ to $\infty$ do                          ▷ Iterate the counter for the dictionary $D_{ori}$ using $K_1$ to get the identifiers associated with the first keyword
3:      $l \leftarrow F(K_1, c)$                                 ▷ Encrypt the counter as the encrypted keyword using the regular key
4:      $d \leftarrow Get(D_{ori}, l)$                           ▷ Check whether the encrypted keyword has an associated identifier
5:      if $d = \perp$ then break end if                         ▷ If no associated identifier, then break the for-loop
6:      $res \leftarrow res \cup \{Dec(K_2, d)\}$                ▷ Decrypt the associated identifier and add it to the search result
7:  end for
8:  for $c \leftarrow 0$ to $\infty$ do                          ▷ Iterate the counter for the dictionary $D_{ori}$ using $K_1^+$ to get the identifiers associated with the first keyword
9:      $l \leftarrow F(K_1^+, c)$                               ▷ Encrypt the counter as the encrypted keyword using the addition secrete key
10:     $d \leftarrow Get(D_{ori}, l)$                           ▷ Check whether the encrypted keyword has an associated identifier
11:     if $d = \perp$ then break end if                         ▷ If no associated identifier, then break the for-loop
12:     $res \leftarrow res \cup \{Dec(K_2^+, d)\}$              ▷ Decrypt the associated identifier and add it to the search result
13: end for
14: for each $id \in res$ do                                     ▷ Filter out those cached for deletion or not associated with the other keywords
15:     $delid \leftarrow F(K_1^-, id)$                          ▷ Compute the $delid$ using the deletion secret key
16:     if $delid \in S_{del}$ then                              ▷ Check whether $delid$ is in $S_{del}$
17:         $res \leftarrow res \setminus \{id\}$                ▷ if $delis$ is cached for deletion, then remove it from the search result
18:     else
19:         for $i \leftarrow 2$ to $k$ do                       ▷ Iterate over the other keywords
20:             if $F(K_i^T, id) \notin S_{tag}$ then $res \leftarrow res \setminus \{id\}$ end if   ▷ If the potential result is not associated with any keyword, then remove it from the search result
21:         end for
22:     end if
23: end for
24: $Result \leftarrow res$                                      ▷ Assign $res$ to $Result$ as the final multi-keyword search result
```

symmetric key. Finally, we simultaneously deal with the deletion set and the other $k-1$ keywords. For each of the identifiers id after decryption, we check whether the encrypted result of id using the pseudorandom deletion label K_1^- is in the set of S_{del} and whether any of the $k-1$ tags after applying PRF F to id is not in the tag list S_{tag}. If either of the two conditions holds, id will be excluded from the result.

Finally, we analyze the time delay and financial cost and compare the traditional and our methods. The traditional method takes $O(l)$ time and generates $O(n)$ identifiers for each single-keyword search query. Then, it takes an extra $O(k \cdot n \cdot \log n)$ time to calculate the intersection of k sets, each of which is of size $O(n)$. Since the writing operations dominate the financial cost for a smart contract [29], we approximate the financial cost as the number of identifiers in the intermediate and final results. Hence, the time delay and financial overhead are $O(k \cdot l + k \cdot n \cdot \log n)$ and $O(k \cdot n)$, respectively. Our method takes $O(l)$ time to use the first keyword to filter the database. Then, $\beta \cdot n$ identifiers will be generated and verified through $k-1$ tags, which takes $O(k \cdot \beta \cdot n)$ time. Therefore, the time overhead for our approach is $O(l + k \cdot \beta \cdot n)$. In the experiments, we figure that β can be as small as 10%, remarkably reducing time complexity. The financial cost overhead is $O(n)$ since only the final result will be written to the smart contract.

4. Performance evaluation

In this section, we conduct extensive experiments on three public datasets to evaluate the proposed protocols of setup, addition, deletion, single-keyword search, and multi-keyword search.

4.1. Experimental settings

We implement the setup, addition, deletion, and search protocols in Python 3.9, using the PyCryptodome and pybloom [1] packages to fulfill the pseudorandom HMAC-SHA256 function and bloom filter. We run the ITS data owner and the blockchain nodes on workstations running Ubuntu 20.04.3 with 32 GB RAM and Intel Core i 9-10900 CPU. The blockchain nodes form a peer-to-peer network through the local area network and employ the proof-of-work [8] as the blockchain consensus protocol. The nodes respond to the setup, addition, deletion, and search requests from the ITS data owner by running the corresponding smart contracts. We employ proof of work as the blockchain consensus protocol and set the difficulties low so as to neglect the influence of the blockchain performance while focusing on the proposed data-sharing protocols only.

After setting up the experimental environment, we have conducted extensive experiments on three datasets: Enron Email Dataset [21], Earth Surface Temperature Dataset,[1] and New York City Bus Dataset[2]. The summary of the datasets is as follows:

- *Enron Email Dataset.* The dataset contains as many as 517 thousand emails. Each email is regarded as a new identifier id. The words inside each email after lowercase are the associated keywords regarding id. The lowercase operations can reduce the number of keywords.
- *Earth Surface Temperature Dataset.* It is a temperature dataset all around the world starting from 1750. Each row is treated as a new identifier id, and each attribute in the row, e.g., average temperature, city, country, latitude, and longitude, is treated as a

[1] https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data.

[2] https://www.kaggle.com/stoney71/new-york-city-transport-statistics.

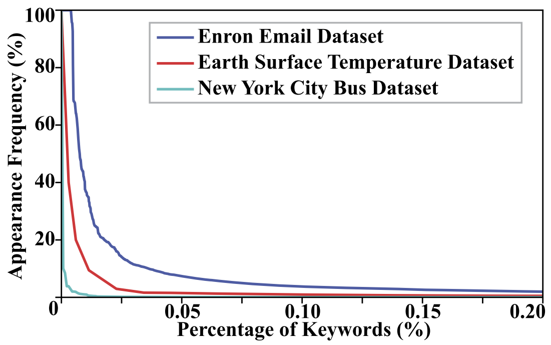

Fig. 3. Distribution of keyword appearance. In either database, only a tiny percentage (less than 0.05%) of keywords are associated with many (less than 10%) identifiers.

keyword associated with id. Note that the figures, i.e., temperature, latitude, and longitude, are narrowed down, losing all the digits after the decimal point.

- *New York City Bus Dataset.* It is an intelligent transportation dataset consisting of bus locations, routes, bus stops, etc. We treat each row in the dataset as a new identifier id, and each attribute in the row, e.g., origin name, destination name, vehicle reference number, as a keyword associated with id. Similarly, the figures, i.e., latitude and longitude of the origins and destinations, are narrowed down, losing the digits after the decimal point.

The sizes of the three datasets are given in Table 1. In particular, the New York City Bus Dataset has a more significant number of identifiers and identifier-keyword pairs, and the Enron Email Dataset has numerous distinct keywords. Fig. 3 depicts the distribution of the appearance times of different keywords. Some keywords are associated with all identifiers in the Enron Email Dataset because almost all emails contain some words, e.g., “message”, “content”, and “type”. We can observe that for all three datasets, no more than 0.05% keywords appear in at least 10% identifiers, which means there are very few high-frequency keywords. The New York City Bus Dataset has an even smaller number of high-frequency keywords than the others.

4.2. Setup and update

The experimental results of the database setup, addition, and deletion protocols on the three datasets are shown in Table 1. The setup operation takes considerable time and financial cost because of the encryption of a large amount of data and its storage on the blockchain. The number of transactions, time costs, and encrypted database size increase linearly as the number of identifier-keyword pairs increases in the original database. On the side of ITS data owners, most of the time is used in database encryption, i.e., generating L, L_{tag}, secret keys, and bloom filter. The three databases contain large volumes of data, resulting in a large number of transactions generated. The database setup of the three databases consumes transactions whose numbers are up to 42.3 thousand, 325.1 thousand, and 3.6 million. The blockchain nodes handle the received transactions by local storage, which takes a relatively shorter time. The sizes of databases after encrypted are as large as 532 MB, $4,084$ MB, and $45,820$ MB, respectively.

The addition and deletion operations consume much less time and fewer transactions than the setup operation from the perspectives of both the ITS data owner and the blockchain nodes. For example, in the Enron Email Dataset, each data owner and blockchain node only takes around 1 s on average to complete the operation of addition or deletion. For the large dataset, e.g., New York City Bus Dataset, such operation is still efficient and takes no more than 4 s on average.

We add extra data structures to boost the efficiency of multi-keyword search queries. The added data structures are the bloom filter bf, tag secret key K^T, and tag list L_{tag}. To this end, the proposed protocol demands approximately half more time, number of transactions, and amount of storage space compared to the traditional protocol. Nevertheless, the proposed approach only influences the database setup operation, making itself acceptable in that the setup protocol will be invoked only once in the whole life cycle of data usage.

4.3. Single-keyword search

Regarding single-keyword search, the time overhead from the perspective of the ITS data owner is low. The reason is that it only proceeds with a small number of symmetric encryption steps. As far as the blockchain nodes are concerned, they are required to pass through the dictionary D_{ori} twice and with a large number of writing operations, incurring relatively high time overhead. We perform experiments on single-keyword searches varying the keywords with different appearance times. The results are shown in Fig. 4. We can see that 5.1 s, 37.5 s, and 86.3 s are needed for the three datasets with no matched identifiers. Note that the times are spent traversing the dictionary D_{ori}. When more identifiers are in the query result, the search time increases linearly. When there are up to 500 matched identifiers, the single-keyword search times are increased to 14.6 s, 76.92 s, and 140.61 s for the three datasets. The search time per identifier drops when the results have more identifiers. It is because a relatively larger number of identifiers can reduce the average time overhead traversing the dictionary. In particular, when there are 500 matched identifiers, the search times per identifier are lower than 0.4 s for all three datasets.

Table 1
Database Setup and Dynamic Update.

Dataset (with Size)	Operation	No Multi-keyword Search Protocol	Multi-keyword Search Protocol
Enron Email Dataset 1, 352 MB in Size 517.0 K Identifiers 622.0 K Distinct Keywords 9.1 M Identifier-keyword Pairs	Setup	Data Owner 209 s Smart Contract 28, 219 Txs Blockchain Nodes 2.6 s Encrypted Database 378 MB	Data Owner 272 s Smart Contract 42, 356 Txs Blockchain Nodes 3.0 s Encrypted Database 532 MB
	Addition	Data Owner 1.4 s Smart Contract 1 Tx Blockchain Nodes 1.2 s	Data Owner 2.0 s Smart Contract 1 Tx Blockchain Nodes 1.5 s
	Deletion	Data Owner 1.0 s Smart Contract 1 Tx Blockchain Nodes 1.2 s	Data Owner 1.0 s Smart Contract 1 Tx Blockchain Nodes 1.4 s
Earth Surface Temperature Dataset 572 MB in Size 10.0 M Identifiers 8.7 K Distinct Keywords 69.9 M Identifier-keyword Pairs	Setup	Data Owner 1, 650 s Smart Contract 216, 763 Txs Blockchain Nodes 20.0 s Encrypted Database 2, 903 MB	Data Owner 2, 105 s Smart Contract 325, 145 Txs Blockchain Nodes 22.2 s Encrypted Database 4, 084 MB
	Addition	Data Owner 1.7 s Smart Contract 1 Tx Blockchain Nodes 2.1 s	Data Owner 2.2 s Smart Contract 1 Tx Blockchain Nodes 3.5 s
	Deletion	Data Owner 1.9 s Smart Contract 1 Tx Blockchain Nodes 2.5 s	Data Owner 2.4 s Smart Contract 1 Tx Blockchain Nodes 3.1 s
New York City Bus Dataset 5, 284 MB in Size 26.5 M Identifiers 134.0 K Distinct Keywords 783.4 M Identifier-keyword Pairs	Setup	Data Owner 19, 032 s Smart Contract 2, 429, 374 Txs Blockchain Nodes 243.2 s Encrypted Database 32, 544 MB	Data Owner 24, 216 s Smart Contract 3, 646, 429 Txs Blockchain Nodes 269.5 s Encrypted Database 45, 820 MB
	Addition	Data Owner 3.2 s Smart Contract 1 Tx Blockchain Nodes 2.8 s	Data Owner 3.5 s Smart Contract 1 Tx Blockchain Nodes 5.8 s
	Deletion	Data Owner 2.9 s Smart Contract 1 Tx Blockchain Nodes 3.2 s	Data Owner 3.1 s Smart Contract 1 Tx Blockchain Nodes 4.9 s

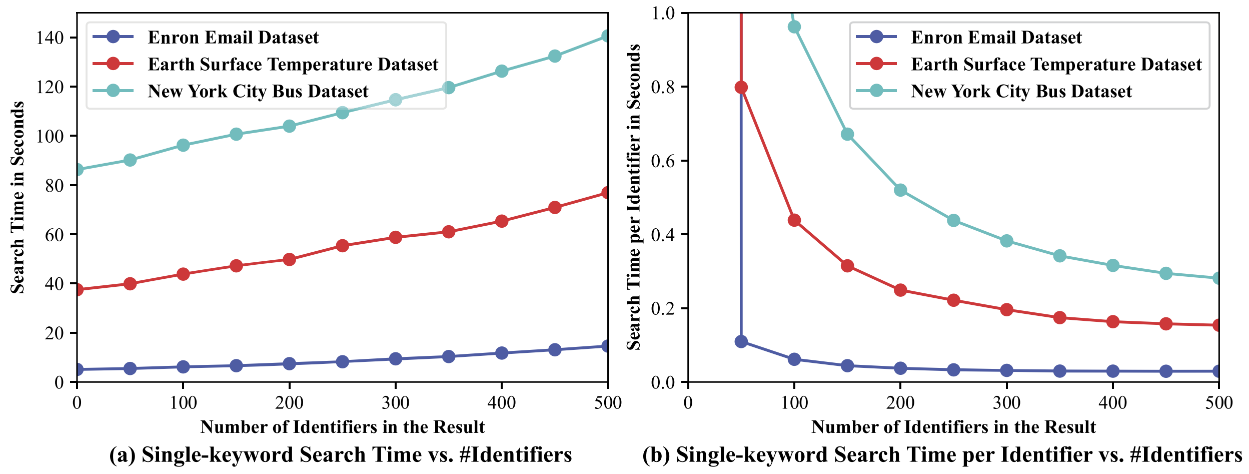

Fig. 4. Experimental results of the single-keyword search. (a) The search time with increasing identifiers in the results. (b) The search time per identifier with increasing identifiers in the results.

4.4. Multi-keyword search

We conduct experiments for multi-keyword search over the traditional method and our proposed method when the number of keywords ranges from 2 to 7. Such numbers are reasonable because we seldom use too many keywords to search databases. The traditional method is to apply the single-keyword search protocol multiple times and take the intersection of the results as the final output. In our method, we set β to be 10 since no more than 0.05% keywords appear in at least 10% identifiers for all the three databases, as shown in Fig. 3. We run each set of experiments for 100 times. The comparison results in terms of the time and

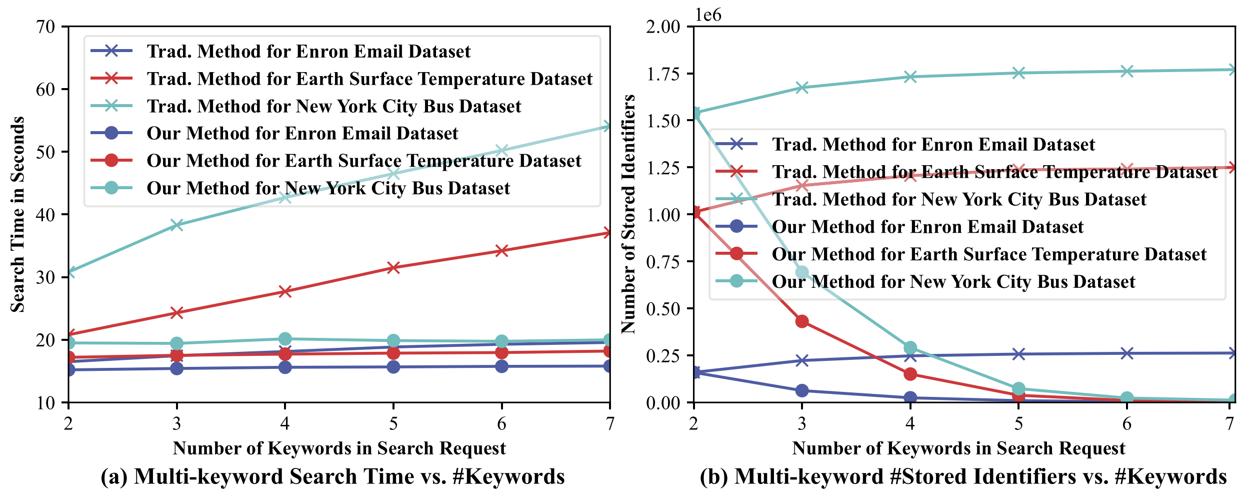

Fig. 5. Experimental results of the multi-keyword search. (a) The search time with increasing keywords in the request. (b) The number of stored identifiers (financial overhead) with increasing keywords in the request.

financial overhead are shown in Fig. 5. Note that extreme cases, e.g., all the keywords are of high frequencies, are not included in the experiments because the keywords are randomly generated.

In terms of the time overhead, the intersection method is significantly affected by the number of keywords because the intersections of more sets should be calculated when the number of keywords increases. Regarding our method, the time overhead to perform a multi-keyword search query does not significantly vary when there are more keywords. The reason is that there will be few candidate identifiers after filtering the database using the first low-frequency keyword. Furthermore, just a single step of tag comparison will be added to the computational overhead when there is one more keyword. When seven keywords are the input, the traditional method demands up to 19.6 s, 37.1 s, and 54.1 s. In contrast, our proposed method only needs 15.8 s, 18.2 s, and 20.0 s for the three datasets, respectively. We repeat a thousand times to choose a random number of two to seven keywords and perform the traditional and proposed multi-keyword search protocols. The results show that our proposed multi-keyword search protocol is superior to the traditional approach, with an average of 14.67% shorter query time.

Regarding the financial overhead of multi-keyword search, the data storage operations dominate compared to others. The reason is that the data is stored via confirming blockchain transactions, which is costly. As a result, we only consider the number of identifiers written to the smart contract state concerning the financial cost. Regarding the traditional method, the number of stored identifiers accumulates up to 60% when there are five keywords instead of two. The amount remains nearly the same when there are 5 or more keywords. The reason is that there will be few identifiers with the same 5 or more keywords. The financial cost of our proposed approach is remarkably saved when there are more keywords because the smart contract writes the result only once. Besides, the number of eligible identifiers decreases with more filtering keywords. For example, when there are 7 keywords in the search query, the traditional method stores up to $262,727$, $1,249,819$, and $1,769,918$ identifiers, while our proposed method only stores $1,485$, $5,342$, and $12,342$ transactions for the three datasets, respectively. The reduction of financial overhead is remarkable. We evaluate the financial overhead similar to time overhead. The results show that, on average, our proposed multi-keyword search protocol outperforms the traditional intersection approach by 59.96% in terms of the financial cost.

5. Conclusion and future directions

In this work, we propose a secure and efficient ITS data-sharing system with the support of database setup, dynamic update, and multi-keyword search. To the best of our knowledge, this work is the first to present an efficient multi-keyword search protocol over the blockchain. The main technical contributions lie in leveraging a bloom filter to select a low-frequency keyword in a search query and filtering the ITS database using the keyword to improve the search efficiency significantly. The proposed technique is common for crucial blockchain applications other than ITS. The future directions are twofold as follows.

On the one hand, it is highly demanded to design high-performance protocols supporting expressive queries, e.g., range, boolean, and structured queries, to make a blockchain system act like a search engine or a database in terms of query support and performance. These queries will enable blockchain-based ITS with support for map crowdsourcing, accurate localization, route planning, etc. On the other hand, different techniques, e.g., verifiable computation and searchable encryption, are currently employed to provide integrity and privacy. However, the techniques can hardly be integrated. As a result, it is essential to design a general framework or cryptographic primitive guaranteeing integrity and privacy simultaneously.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The work was supported by the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University, Hong Kong Research Grant Council Theme-based Research Scheme No. T41-603/20-R, Collaborative Research Fund No. C2004-21GF, and Research Impact Fund No. R5034-18 and No. R5009-21, Shenzhen Municipal Science and Technology Innovation Commission No. SGDX2020110309520302, National Natural Science Foundation of China No. 61802383, Research Project of Pazhou Lab for Excellent Young Scholars No. PZL2021KF0024, and Guangzhou Basic and Applied Basic Research Foundation No. 202201010330 and No. 202201020162.

References

[1] P.S. Almeida, C. Baquero, N. Preguiça, D. Hutchison, Scalable bloom filters, Inf. Process. Lett. 101 (2007) 255–261.
[2] M. Autili, L. Chen, C. Englund, C. Pompilio, M. Tivoli, Cooperative intelligent transport systems: choreography-based urban traffic coordination, IEEE Trans. Intell. Transp. Syst. 22 (2021) 2088–2099.
[3] C. Cai, J. Weng, X. Yuan, C. Wang, Enabling reliable keyword search in encrypted decentralized storage with fairness, IEEE Trans. Dependable Secure Comput. 18 (2018) 131–144.
[4] L. Chen, W.K. Lee, C.C. Chang, K.K.R. Choo, N. Zhang, Blockchain based searchable encryption for electronic health record sharing, Future Gener. Comput. Syst. 95 (2019) 420–429.
[5] J. Cui, F. Ouyang, Z. Ying, L. Wei, H. Zhong, Secure and efficient data sharing among vehicles based on consortium blockchain, IEEE Trans. Intell. Transp. Syst. 23 (2022) 8857–8867.
[6] N. Deepa, Q.V. Pham, D.C. Nguyen, S. Bhattacharya, B. Prabadevi, T.R. Gadekallu, P.K.R. Maddikunta, F. Fang, P.N. Pathirana, A survey on blockchain for big data: approaches, opportunities, and future directions, Future Gener. Comput. Syst. 131 (2022) 209–226.
[7] M. Dibaei, X. Zheng, Y. Xia, X. Xu, A. Jolfaei, A.K. Bashir, U. Tariq, D. Yu, A.V. Vasilakos, Investigating the prospect of leveraging blockchain and machine learning to secure vehicular networks: a survey, IEEE Trans. Intell. Transp. Syst. 23 (2022) 683–700.
[8] A. Gervais, G.O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 3–16.
[9] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions, J. ACM 33 (1986) 792–807.
[10] C. Gosman, T. Cornea, C. Dobre, F. Pop, A. Castiglione, Controlling and filtering users data in intelligent transportation system, Future Gener. Comput. Syst. 78 (2018) 807–816.
[11] Z. Guan, N. Wang, X. Fan, X. Liu, L. Wu, S. Wan, Achieving secure search over encrypted data for e-commerce: a blockchain approach, ACM Trans. Internet Technol. 21 (2020) 1–17.
[12] J. Guo, X. Ding, T. Wang, W. Jia, Combinatorial resources auction in decentralized edge-thing systems using blockchain and differential privacy, Inf. Sci. 607 (2022) 211–229.
[13] V. Hassija, V. Gupta, S. Garg, V. Chamola, Traffic jam probability estimation based on blockchain and deep neural networks, IEEE Trans. Intell. Transp. Syst. 22 (2020) 3919–3928.
[14] R.W. van der Heijden, S. Dietzel, T. Leinmüller, F. Kargl, Survey on misbehavior detection in cooperative intelligent transportation systems, IEEE Commun. Surv. Tutor. 21 (2018) 779–811.
[15] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, K. Ren, Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization, in: IEEE Conference on Computer Communications, IEEE, 2018, pp. 792–800.
[16] U. Javaid, M.N. Aman, B. Sikdar, DrivMan: driving trust management and data sharing in VANETs with blockchain and smart contracts, in: Vehicular Technology Conference (VTC-Spring), IEEE, 2019, pp. 1–5.
[17] S. Jiang, J. Cao, J.A. McCann, Y. Yang, Y. Liu, X. Wang, Y. Deng, Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain, in: International Conference on Blockchain, IEEE, 2019, pp. 405–410.
[18] S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma, J. He, BlocHIE: a blockchain-based platform for healthcare information exchange, in: International Conference on Smart Computing (SMARTCOMP), IEEE, 2018, pp. 49–56.
[19] K. Kambatla, G. Kollias, V. Kumar, A. Grama, Trends in big data analytics, J. Parallel Distrib. Comput. 74 (2014) 2561–2573.
[20] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, Y. Zhang, Blockchain for secure and efficient data sharing in vehicular edge computing and networks, IEEE Int. Things J. 6 (2018) 4660–4670.
[21] B. Klimt, Y. Yang, The enron corpus: a new dataset for email classification research, in: European Conference on Machine Learning, Springer, 2004, pp. 217–226.
[22] B. Ko, K. Liu, S.H. Son, K.J. Park, RSU-assisted adaptive scheduling for vehicle-to-vehicle data sharing in bidirectional road scenarios, IEEE Trans. Intell. Transp. Syst. 22 (2020) 977–989.
[23] Q. Kong, R. Lu, M. Ma, H. Bao, A privacy-preserving sensory data sharing scheme in Internet of vehicles, Future Gener. Comput. Syst. 92 (2019) 644–655.
[24] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: the blockchain model of cryptography and privacy-preserving smart contracts, in: IEEE Symposium on Security and Privacy, 2016, pp. 839–858.
[25] H. Krawczyk, M. Bellare, R. Canetti, HMAC: keyed-hashing for message authentication, RFC 2104 (1997) 1–11.
[26] S. Kudva, S. Badsha, S. Sengupta, I. Khalil, A. Zomaya, Towards secure and practical consensus for blockchain based VANET, Inf. Sci. 545 (2021) 170–187.
[27] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C.P.A. Ogah, Z. Sun, Blockchain-based dynamic key management for heterogeneous intelligent transportation systems, IEEE Int. Things J. 4 (2017) 1832–1843.
[28] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, Z. Zhang, CreditCoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles, IEEE Trans. Intell. Transp. Syst. 19 (2018) 2204–2220.
[29] X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, Future Gener. Comput. Syst. 107 (2020) 841–853.

[30] Y. Li, G. Liu, Z.L. Zhang, J. Luo, F. Zhang, CityLines: designing hybrid hub-and-spoke transit system with urban big data, IEEE Trans. Big Data 5 (2018) 576–587.
[31] M.B. Mollah, J. Zhao, D. Niyato, Y.L. Guan, C. Yuen, S. Sun, K.Y. Lam, L.H. Koh, Blockchain for the Internet of vehicles towards intelligent transportation systems: a survey, IEEE Int. Things J. 8 (2020) 4157–4185.
[32] R.A. Poldrack, K.J. Gorgolewski, Making big data open: data sharing in neuroimaging, Nat. Neurosci. 17 (2014) 1510–1517.
[33] Z. Qu, Z. Zhang, M. Zheng, A quantum blockchain-enabled framework for secure private electronic medical records in Internet of medical things, Inf. Sci. 612 (2022) 942–958.
[34] K.N. Qureshi, G. Jeon, M.M. Hassan, M.R. Hassan, K. Kaur, Blockchain-based privacy-preserving authentication model intelligent transportation systems, IEEE Trans. Intell. Transp. Syst. (2022).
[35] J.E. Siegel, D.C. Erb, S.E. Sarma, A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas, IEEE Trans. Intell. Transp. Syst. 19 (2017) 2391–2406.
[36] J. Sun, H. Xiong, S. Zhang, X. Liu, J. Yuan, R.H. Deng, A secure flexible and tampering-resistant data sharing system for vehicular social networks, IEEE Trans. Veh. Technol. 69 (2020) 12938–12950.
[37] J. Sun, G. Xu, T. Zhang, X. Cheng, X. Han, M. Tang, Secure data sharing with flexible cross-domain authorization in autonomous vehicle systems, IEEE Trans. Intell. Transp. Syst. (2022).
[38] X. Tang, C. Guo, K.K.R. Choo, Y. Liu, L. Li, A secure and trustworthy medical record sharing scheme based on searchable encryption and blockchain, Comput. Netw. 200 (2021) 108540.
[39] S. Underwood, Blockchain beyond bitcoin, Commun. ACM 59 (2016) 15–17.
[40] M. Veres, M. Moussa, Deep learning for intelligent transportation systems: a survey of emerging trends, IEEE Trans. Intell. Transp. Syst. 21 (2019) 3152–3168.
[41] Q. Wei, B. Li, W. Chang, Z. Jia, Z. Shen, Z. Shao, A survey of blockchain data management systems, ACM Trans. Embed. Comput. Syst. 21 (2022) 25:1–25:28.
[42] C. Xu, C. Zhang, J. Xu, VChain: enabling verifiable Boolean range queries over blockchain databases, in: International Conference on Management of Data, 2019, pp. 141–158.
[43] Y. Yuan, F.Y. Wang, Towards blockchain-based intelligent transportation systems, in: International Conference on Intelligent Transportation Systems, IEEE, 2016, pp. 2663–2668.
[44] C. Zhang, C. Xu, J. Xu, Y. Tang, B. Choi, GEMˆ2-tree: a gas-efficient structure for authenticated range queries in blockchain, in: International Conference on Data Engineering, IEEE, 2019, pp. 842–853.
[45] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, Z. Li, CityFlow: a multi-agent reinforcement learning environment for large scale city traffic scenario, in: The World Wide Web Conference, 2019, pp. 3620–3624.
[46] J. Zhang, F.Y. Wang, K. Wang, W.H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: a survey, IEEE Trans. Intell. Transp. Syst. 12 (2011) 1624–1639.
[47] K. Zhang, J. Long, X. Wang, H.N. Dai, K. Liang, M. Imran, Lightweight searchable encryption protocol for industrial Internet of things, IEEE Trans. Ind. Inform. 17 (2020) 4248–4259.
[48] Y. Zhang, R. Deng, X. Liu, D. Zheng, Outsourcing service fair payment based on blockchain and its applications in cloud computing, IEEE Trans. Serv. Comput. 14 (2021) 1152–1166.
[49] J. Zhu, J. Cao, D. Saxena, S. Jiang, H. Ferradi, Blockchain-Empowered Federated Learning: Challenges, Solutions, and Future Directions, ACM Computing Surveys, 2022.