Toward Privacy-Preserving Cybertwin-Based Spatiotemporal Keyword Query for ITS in 6G Era

Yunguo Guan, Rongxing Lu, *Fellow, IEEE*, Yandong Zheng, Songnian Zhang, Jun Shao, *Associate Member, IEEE*, and Guiyi Wei, *Member, IEEE*

***Abstract*—The sixth-generation (6G) communication technology has been attracting great interests from both industry and academia, as it is regarded as a promising approach to achieve more stable and low-latency communication. These promising features of 6G make it an enabler for cybertwin, a technique to create digital representations for physical objects to implement various functionalities. In this article, we consider a cybertwin-based spatiotemporal keyword query service over a dynamic message data set in intelligent transportation system (ITS) scenarios. Particularly, in the considered service, publishers upload messages to the cloud, and each cybertwin predictively launches queries to retrieve messages on behalf of the corresponding vehicle, such that each vehicle can timely receive messages that are of its interest whenever it arrives at a location. Nevertheless, as the cloud is not fully trustable, there exist privacy concerns related to the messages and queries. Up to now, although many schemes have been proposed to handle privacy-preserving spatial, temporal, or keyword queries, none of them can simultaneously support queries containing both spatial, temporal, and keyword criteria on dynamic data sets. Aiming at the issue, we design a layered index based on segment trees to dynamically organize messages containing both spatial, temporal, and keyword information. Moreover, based on a symmetric homomorphic encryption scheme, we encrypt the messages and queries and present a two-server privacy-preserving spatiotemporal keyword query scheme. We analyze the security of the proposed scheme and also conduct extensive experiments to evaluate its performance. The results show that our proposed scheme is indeed privacy preserving and computationally efficient.**

***Index Terms*—Cybertwin, intelligent transportation system (ITS), privacy-preserving, sixth-generation (6G), spatiotemporal keyword query.**

I. Introduction

DRIVEN by the pressing need for connectivity with diverse requirements, wireless communication techniques have been evolving for centuries and, in return, they boost the deployment of connected devices together with information technology. The fifth-generation (5G) technology standard, as the latest generation of wireless technology, has been estimated to reach a market size of U.S. $41.48 billion by 2020 and expand at a compound annual growth rate of 43.9% from 2021 to 2027, as reported in [1]. Although it significantly benefits our daily lives by enabling connections between heterogeneous devices, there are still many challenges ahead, e.g., lower latency, higher data rates, and wider coverage. Therefore, as its successor, sixth generation (6G) has been attracting interests from both industry and academia, and it is expected to achieve low-latency communication and highly dynamic network topology. These features make 6G an enabler for various applications that rely heavily on stable and low-latency communications, such as cybertwin considered in this work [2]. Cybertwin is regarded as a promising technique [3] to create digital representations for physical objects to implement various functionalities, e.g., communication anchors. Furthermore, by leveraging the computational resource, a cybertwin can predict the future state of the corresponding physical object and take some predefined actions based on the prediction [4]. Fig. 1 shows the architecture of a general 6G-based cybertwin application in the intelligent transportation system (ITS). Based on this architecture, we consider a cybertwin-based spatiotemporal keyword query service. In the service, a cloud is employed for an enhanced quality of service and flexible computational resources, similar to many other query services [5]–[7]. Then, vehicles in the scenario are the physical objects and each vehicle has a cybertwin, and a set of publishers deliver messages to the authorized vehicles through the cloud. Specifically, the publishers upload their messages to the cloud, and each cybertwin predicts the future locations of the corresponding vehicle and pulls the messages that may be of interest to the vehicle. In this way, not only each vehicle can promptly receive the messages of its interest whenever it arrives at a location, but the cybertwins can also mine some insights from these messages to assist driving [8]. For instance, by collecting messages related to traffic, the cybertwin can select the best route among several candidate routes and push the resulting route to the corresponding vehicle.

While enjoying the cybertwin-based spatiotemporal keyword query service, the publishers and the vehicles have their own privacy concerns, as the cloud is not fully trustable. On the one hand, the messages are private to the publishers and should only be available to authorized users. On the

Manuscript received February 12, 2021; revised May 23, 2021; accepted June 29, 2021. Date of publication July 12, 2021; date of current version November 5, 2021. This work was supported in part by NSERC Discovery under Grant 04009; in part by ZJNSF under Grant LZ18F020003; and in part by NSFC under Grant U1709217. *(Corresponding author: Rongxing Lu.)*

Yunguo Guan, Rongxing Lu, Yandong Zheng, and Songnian Zhang are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B5A3, Canada (e-mail: yguan4@unb.ca; rlu1@unb.ca; yzheng8@unb.ca; szhang17@unb.ca).

Jun Shao is with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: chn.junshao@gmail.com).

Guiyi Wei is with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: weigy@zjgsu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3096674

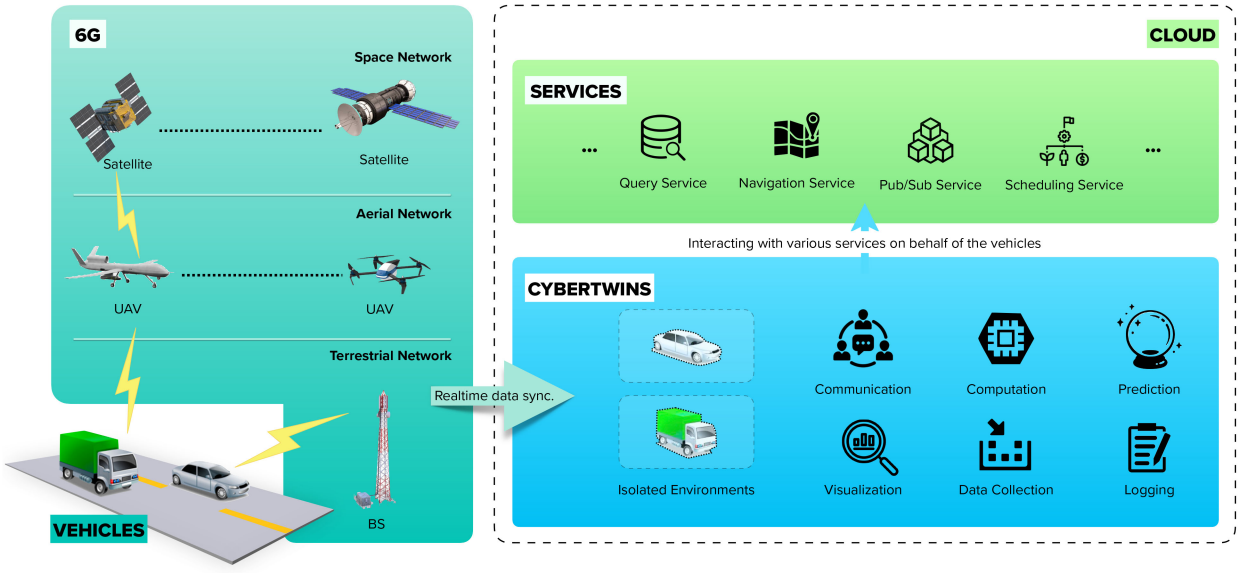


Fig. 1. Architecture of 6G-based cybertwin applications in the ITS scenario.

other hand, the queries submitted by the cybertwins should be protected. Otherwise, the locations and other information included in the queries may be revealed to the cloud and other adversaries, which may be exploited for further attacks and put the vehicles and drivers in danger. Therefore, both the messages and the queries need to be encrypted before being uploaded to the cloud. However, it is commonly acknowledged that encryption techniques inevitably hinder the cloud to conduct queries. As detailed in Section VII, although many schemes [9]–[26] are proposed to support queries with spatial, temporal, or keyword criteria, they cannot be efficiently applied to our scenario. Some of them [9]–[19] can only support queries with spatial or spatio-textual queries over static data sets. Some others [20]–[26] can only support queries with multiple keywords over dynamic data sets. Adapting these works to handle spatiotemporal keyword query is inefficient, as it needs to additionally choose a large number of keywords to represent areas and time periods. As a result, the existing works cannot be efficiently adapted to support spatiotemporal keyword queries over a dynamic data set. Thus, querying with both spatial, temporal, and keyword conditions over a dynamic data set is still challenging.

To address the above challenges, we present our privacy-preserving cybertwin-based spatiotemporal keyword query scheme for ITS scenarios. The main contributions of this article are fourfold.

1) First, we design a layered data structure based on segment trees for dynamically indexing the messages posted by the publishers, and the index can efficiently support queries containing both spatial, temporal, and keyword criteria.
2) Second, we propose two algorithms to, respectively, encrypt the index and queries. These two algorithms are characterized by outputting ciphertexts of a symmetric homomorphic encryption (SHE) scheme without knowing the secret key.
3) Third, we present our proposed cybertwin-based privacy-preserving spatiotemporal keyword query scheme to index messages and conduct queries over the messages while preserving the privacy of both the messages and the queries.
4) Finally, we analyze the security of our proposed scheme and conduct extensive experiments to demonstrate the efficiency of our proposed scheme. The results show that our proposed scheme is indeed privacy preserving and computationally efficient.

The remainder of this article is organized as follows. In Section II, we introduce our system model, security model, and design goal. After that, we review some preliminaries in Section III. In Section IV, we present our proposed scheme, followed by its security analysis and performance evaluation, respectively, in Sections V and VI. Then, we present some related work in Section VII. Finally, we draw our conclusion in Section VIII.

II. Models and Design Goal

In this section, we formalize our system model and security model and identify our design goal.

A. System Model

In our system model, we consider a privacy-preserving cybertwin-based spatiotemporal keyword query scenario in ITS, which contains five types of entities, namely, a service provider, a set of publishers, a cloud consisting of two servers $\{CS_1, CS_2\}$, a set of vehicles, and a set of cybertwins corresponding to the vehicles, as shown in Fig. 2.

1) Service Provider: The service provider outsources the privacy-preserving cybertwin-based spatiotemporal keyword query service to the cloud, and it is responsible for setting up the whole system. That is, it will generate and securely distribute secret keys to each entity in the system. Moreover, it partitions the city map into several regions

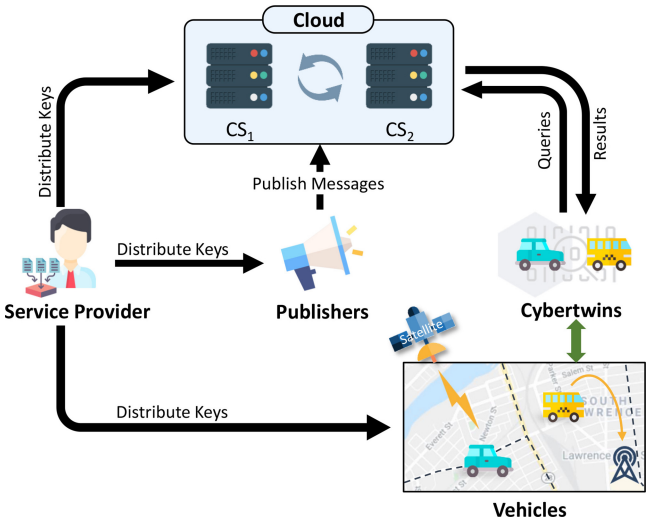


Fig. 2. System model under consideration.

TABLE I
FORMAT OF A PUBLISHED MESSAGE m_i

	Content	Location	Keywords	Release Date	Release Time	Expire Time
m_i	c_i	L_i	W_i	dt	t_i	$\hat{t}_i$
e.g.	"..."	3	{"Rain"}	20210201	0830	1620

$\mathcal{R} = \{R_1, R_2, \ldots, R_{|\mathcal{R}|}\}$ based on historical data, such that each region $R_i \in \mathcal{R}$ roughly has a similar rate of new messages.

2) Publishers: Each publisher continuously publishes messages to authorized vehicles through the cloud. Specifically, a message $m_i = (c_i, \mathsf{L}_i, W_i, dt, t_i, \hat{t}_i)$ has six attributes, namely: 1) its content c_i; 2) its location L_i; 3) its related keywords W_i; 4) its release date dt; 5) its release time t_i; and 6) its expire time $\hat{t}_i$, as shown in Table I.

3) Cloud: The cloud contains two servers $\{\mathrm{CS}_1, \mathrm{CS}_2\}$ and runs the outsourced spatiotemporal keyword query service. In specific, it stores the messages published by the publishers, and on receiving a spatiotemporal keyword query request, it responds with a set of messages satisfying the request. Specifically, given a spatiotemporal keyword query request $(\mathsf{L}_q, t_q, \hat{t}_q, W_q, dt)$, the cloud responds a set of messages satisfying that: 1) they are in the same region as L_q; 2) they are published after t_q; 3) they are valid at $\hat{t}_q$; and 4) each of them contains all keywords in W_q.

4) Vehicles and Cybertwins: To continuously obtain messages related to its location and keywords without constantly querying the service, each vehicle in the system deploys a cybertwin to predictively query the service, and the cybertwin locates in the cloud but has a safe and isolated environment. In specific, each vehicle V maintains a secure and stable connection to its cybertwin and continuously updates its location and keywords $\{\mathsf{L}_v, W_v\}$ to the cybertwin. Then, the latter predicts a set of the vehicle's future locations $\widetilde{L}_v = \{(\mathsf{L}_{v,k}, t_k) \mid k = 1, 2, \ldots\}$, where each tuple $(\mathsf{L}_{v,k}, t_k)$ represents that a possible location of V at time t_k. After that, for each $(\mathsf{L}_{v,k}, t_k) \in \widetilde{L}_v$, the cybertwin periodically queries the service to retrieve the messages at location $\mathsf{L}_{v,k}$ and will be valid at t_k. Moreover, to improve the query efficiency, the cybertwin only requests messages that are newly published since the last query, i.e., $t_i > t_{\text{old}}$, where t_{old} is the timestamp when the last query is submitted. That is, for each location tuple $(\mathsf{L}_{v,k}, t_k)$, the cybertwin launches a spatiotemporal keyword query $\mathcal{Q}_k = (\mathsf{L}_q = \mathsf{L}_{v,k}, t_q = t_{\text{old}}, \hat{t}_q = t_k, W_q = W_v, dt)$.

Communication Model: By employing 6G communication and mature authentication techniques, we assume reliable communication channels can always be established: 1) between each vehicle and its cybertwin; 2) between each cybertwin and the cloud; and 3) between each publisher and the cloud.

B. Security Model

In our security model, the service provider is *trusted*, since he/she owns the service and has no motivation to deviate from it. That is, it will honestly generate and distribute secret keys for all entities in the system. Similar to many existing works (e.g., [6], [27], and [28]), the publishers, vehicles, and cybertwins are authorized by the service provider and will be punished if they deviate from the protocol, so they are considered to be *honest*. Specifically, the publishers will honestly publish messages with correct locations and keywords to the cloud; the vehicles will faithfully update their locations and selected keywords to the corresponding cybertwins; and the cybertwins will honestly query the service and push the query results to the vehicles when they are near the messages' locations. However, the two servers in the cloud are *honest-but-curious*. In specific, although they will faithfully respond to the queries from cybertwins with correct results, they might be curious about the plaintexts of the messages published by the publishers and the vehicles' keywords and predicted locations. Nevertheless, they will not collude with each other. This assumption is reasonable as it captures the limitations of real-world adversaries' power. Note that the external attackers may also launch other active attacks, e.g., Denial of Service (DoS) attacks, to the network. Since this work focuses on privacy preservation, those attacks are beyond the scope of this article and will be discussed in our future work.

C. Design Goal

The design goal of this work is to present a privacy-preserving cybertwin-based spatiotemporal keyword query service to meet the requirements defined in the system model and security model. In specific, the proposed scheme should have the following two properties.

1) *Proposed Scheme Should Be Privacy Preserving:* The cloud servers should not be able to obtain the contents, locations, and related keywords of the published messages, and the queries' locations and keywords. As for the release date and time, they are close to the date and time when the cloud servers receive the encrypted message but are less likely to reveal the content of the message. Hence, we will not protect these two fields of messages.

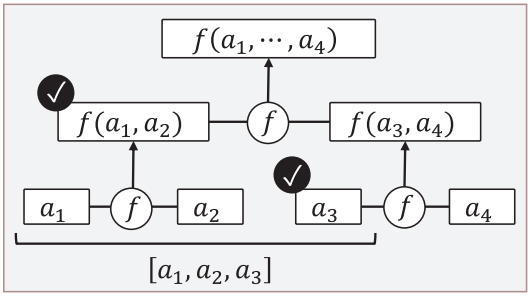


Fig. 3. Example of a segment tree with four records $[a_1, a_2, a_3, a_4]$. With the segment tree, $f(a_1, a_2, a_3)$ can be computed as $f(a_1, a_2, a_3) = f(f(a_1, a_2), f(a_3))$.

2) *Proposed Scheme Should Be Efficient:* To achieve privacy preservation in the cybertwin-based spatiotemporal keyword query service, some cryptographic techniques need to be employed. However, since our work focuses on the ITS scenario, its efficiency should be taken into consideration. That is, to make the scheme practical, its computational cost should be minimized.

III. Preliminaries

In this section, we recall several techniques that will be used in our proposed scheme, namely, segment trees, a SHE scheme [29], two SHE-based privacy-preserving protocols, and the Bloom filter technique.

A. Segment Tree

A segment tree is a binary tree-based data structure for conducting efficient range aggregation queries on a sequential data set, in which the range aggregation function $f(\cdot)$ satisfies

$$f(U) = f(f(S_1), f(S_2), \ldots, f(S_s)),$$
$$\text{where the set } U = S_1 \cup S_2 \cup \cdots \cup S_s,$$
$$\text{and any two subsets } S_i \cap S_j = \emptyset, \text{ for } 1 \le i < j \le s.$$

As illustrated in Fig. 3, a segment tree consists of two types of nodes, namely, inner nodes and leaf nodes. Each leaf node represents a data record in the data set, and each inner node stores the result of applying the selected range aggregation function $f(\cdot)$ on its child nodes. Then, based on the properties of segment trees, a query can be conducted by applying $f(\cdot)$ to the results in a set of nodes that can precisely cover the query range. For example, to obtain $f(a_1, a_2, a_3)$ in the segment tree in Fig. 3, we can compute $f(f(a_1, a_2), f(a_3))$. Generally, when the range length of a query is L, the computational complexity of conducting a query on the given range is $2(\lceil \log L \rceil - 1) = O(\log L)$. Hereinafter, we refer to a forest of segment trees descendingly sorted by their heights as a segment forest for the simplicity of description.

B. Symmetric Homomorphic Encryption

SHE is a SHE scheme [29], which is proved to be indistinguishable chosen plaintext attack (IND-CPA) secure [30]. It comprises the following three algorithms, namely, *key generation*, *encryption*, and *decryption*.

1) Key Generation: Given three security parameters (k_0, k_1, k_2) satisfying $k_1 \ll k_2 < k_0$, the algorithm generates the secret key $\mathsf{SK} = (p, q, \mathcal{L})$, where p and q are two large prime numbers with $|p| = |q| = k_0$ and $\mathcal{L}$ is a random number of bit length $|\mathcal{L}| = k_2$. After that, it computes $\mathcal{N} = pq$ and sets the public parameter $\mathsf{PP} = (k_0, k_1, k_2, \mathcal{N})$. Then, the basic message space $\mathcal{M}$ is $[-2^{k_1-1}, 2^{k_1-1})$.

2) Encryption: With the secret key $\mathsf{SK} = (p, q, \mathcal{L})$, a message $m \in \mathcal{M}$ can be encrypted into a ciphertext $c = E(m) = (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N}$, where $r \in \{0, 1\}^{k_2}$ and $r' \in \{0, 1\}^{k_0}$ are two random numbers.

3) Decryption: With the secret key $\mathsf{SK} = (p, q, \mathcal{L})$, the plaintext m of a ciphertext c can be recovered by computing $m = D(c) = (c \bmod p) \bmod \mathcal{L}$. Since $k_1 \ll k_2 < k_0$, we have $m + r\mathcal{L} < p$ and $m < \mathcal{L}$. Then, we can verify the correctness of the decryption as follows:

$$D(c) = (c \bmod p) \bmod \mathcal{L} = (r\mathcal{L} + m) \bmod \mathcal{L} = m.$$

Given two ciphertexts $c_1 = E(m_1)$ and $c_1 = E(m_2)$, or a ciphertext c_1 and a plaintext m_2, the SHE scheme supports the following nice two homomorphic addition (*Homo-Add*) properties and two homomorphic multiplication (*Homo-Mul*) properties as follows: 1) *Homo-Add-I:* $c_1 + c_2 \bmod \mathcal{N} \rightarrow E(m_1 + m_2)$; 2) *Homo-Mul-I:* $c_1 \cdot c_2 \bmod \mathcal{N} \rightarrow E(m_1 \cdot m_2)$; 3) *Homo-Add-II:* $c_1 + m_2 \bmod \mathcal{N} \rightarrow E(m_1 + m_2)$; and 4) *Homo-Mul-II:* $c_1 \cdot m_2 \bmod \mathcal{N} \rightarrow E(m_1 \cdot m_2)$ when $m_2 > 0$.

Note that after σ times of *Homo-Mul-I* operations, the bit length of $r\mathcal{L}$ in the resulting ciphertext will be $2(\sigma + 1) \cdot k_2$. To correctly decrypt the ciphertext, we need to ensure that $\alpha\mathcal{L} < p$, i.e., $2(\sigma + 1) \cdot k_2 < k_0$. Hence, SHE's maximum depth of multiplication $\sigma = \lfloor (k_0/2k_2) - 1 \rfloor$.

C. Two SHE-Based Privacy-Preserving Protocols

Based on the above SHE scheme, we introduce two privacy-preserving protocols, namely, a bootstrapping protocol and a sign calculating protocol. Both of the protocols are run between CS_1 and CS_2. Before running these protocols, the two cloud servers hold some secrets distributed by the service provider. Specifically, CS_1 holds $\{\mathsf{PP}, E(0)_1, E(0)_2, E(-1)\}$, while CS_2 holds SK of the SHE scheme.

1) Protocol 1 (Privacy-Preserving Bootstrapping Protocol): Given the security parameters $\{k_0, k_1, k_2\}$, the SHE scheme can only support a limited number of multiplications on one ciphertext, as analyzed in Section III. Therefore, the bootstrapping protocol is employed to support infinite numbers of multiplications. Initially, CS_1 holds a ciphertext $c = E(m)$, and CS_2 holds the corresponding private key SK. After running the protocol, CS_1 will obtain a new ciphertext $c' = E(m)$ with the same plaintext m while CS_2 knows nothing about m. Specifically, the protocol runs in the following steps.

Step 1: Given $c = E(m)$, CS_1 homomorphically computes $\tilde{c} = E(m + r)$, where $r \in \{0, 1\}^{k_2-2}$ is a random number. After that, it sends $\tilde{c}$ to CS_2.

Step 2: On receiving $\tilde{c}$, CS_2 decrypts it to obtain $\tilde{m} = D(\tilde{c}) = m + r$. Then, it sends a ciphertext of $\tilde{m}$ encrypted by the private key SK, i.e., $\tilde{c}' = E(\tilde{m})$, to CS_1.

Step 3: Based on the received $\tilde{c}'$, CS_1 recovers the bootstrapped ciphertext c' as $c' = \tilde{c}' - r = E(m)$.

Correctness: The goal of the privacy-preserving bootstrapping protocol is to securely decrypt and re-encrypt a ciphertext, so its correctness holds *iff*: 1) CS_2 can correctly recover the garbled plaintext $m' = m + r$, i.e., $D(\tilde{c}) = \tilde{m}$ and 2) after running step 3, the ciphertext obtained by CS_1 can be correctly decrypted, i.e., $D(\tilde{c}') = m$. Since $m \in \mathcal{M}$ and $r \in \{0,1\}^{k_2-2}$, we have $m + r < \mathcal{L}$. Thus, CS_2 can correctly recover $\tilde{m} = m + r$ by running the *decryption* algorithm of the SHE scheme. Similarly, as $m + r < \mathcal{L}$, we can deduce that $(m + \alpha\mathcal{L} - r) \bmod n < p$ holds. Therefore, $c' = \tilde{c}' - r$ can be correctly decrypted.

2) Protocol 2 (Privacy-Preserving Sign Calculating Protocol): The privacy-preserving sign calculating protocol is to securely compute the encrypted sign of a given ciphertext's plaintext. That is, given a ciphertext $c = E(m)$, after running the protocol with CS_2, CS_1 can obtain $E(\mathtt{sign}(m))$, where $\mathtt{sign}(m) = 1$ if $m \geq 0$, or $\mathtt{sign}(m) = 0$ otherwise. Specifically, the protocol runs in the following steps.

Step 1: Given $c = E(m)$, CS_1 first computes

$$\tilde{c} = \begin{cases} r_1 \cdot E(m) + r_2 \bmod \mathcal{N}, & b = 1 \\ E(-1) \cdot (r1 \cdot E(m) + r_2) \bmod \mathcal{N}, & b = 0 \end{cases}$$

where r_1 and r_2 are two random numbers satisfying $0 < r_2 < r_1$, $r_1 \in \{0,1\}^{k_2-k_1-1}$, and $b \in \{0,1\}$ is a random bit. Then, it sends $\tilde{c}$ to CS_2.

Step 2: Upon receiving $\tilde{c}$, CS_2 decrypts it to obtain $\tilde{m} = D(\tilde{c}) = (-1)^{1-b}(r_1 m + r_2) \bmod \mathcal{L}$. If $\tilde{m} < (\mathcal{L}/2)$, it sets $\tilde{c}' = E(1)$; otherwise, it sets $\tilde{c}' = E(0)$. Finally, CS_2 sends $\tilde{c}'$ to CS_1.

Step 3: Based on the received $\tilde{c}'$, CS_1 computes $c' = E(\mathtt{sign}(m)) = 1 - b + \tilde{c}' \cdot (2b + E(-1))$.

Correctness: The correctness of the privacy-preserving sign calculating protocol holds *iff* c' satisfies that $D(c') = 1$ when $m \geq 0$, and $D(c') = 0$ otherwise. Then, we first show the relationship between m and $\tilde{c}'$ in the following two cases.

1) *When the Random Bit $b = 1$:* By keeping $m = |m|$ when $m \geq 0$ and computing $m = \mathcal{L} - |m|$; otherwise, we have

$$\begin{aligned} \tilde{m} &= D(\tilde{c}) \\ &= \begin{cases} r_1 \cdot |m| + r_2 \bmod \mathcal{L}, & \text{if } m \geq 0 \\ r_1 \cdot (\mathcal{L} - |m|) + r_2 \bmod \mathcal{L}, & \text{otherwise,} \end{cases} \\ &= \begin{cases} r_1 \cdot |m| + r_2, & \text{if } m \geq 0 \\ \mathcal{L} - r_1 \cdot |m| + r_2, & \text{otherwise.} \end{cases} \end{aligned}$$

Since the bit length of $r_1 \cdot m$ is $k_2 - 1$ will not exceed that of $(\mathcal{L}/2)$, we can easily deduce that $D(\tilde{c}) < (\mathcal{L}/2)$ when $m \geq 0$, or $D(\tilde{c}) > (\mathcal{L}/2)$ otherwise. Then, CS_1 can obtain $\tilde{c}' = E(1)$ if $m > 0$, and $\tilde{c}' = E(0)$ otherwise.

2) *When the Random Bit $b = 0$:* Similarly, we can deduce that when $b = 0$, CS_1 can obtain $\tilde{c}' = E(0)$ if $m > 0$, and $\tilde{c}' = E(1)$ otherwise.

Based on the results of these two cases, we can draw the truth table of $D(c') = D(E(\mathtt{sign}(m)))$ in Table II. From the table, the correctness of the privacy-preserving sign calculating protocol follows.

TABLE II
TRUTH TABLE OF $D(c') = D(E(\mathtt{sign}(m)))$

$\mathtt{sign}(m)$	b	$D(\tilde{c}')$	$D(c') = D(E(\mathtt{sign}(m)))$
0	0	1	0
0	1	0	0
1	0	0	1
1	1	1	1

D. Bloom Filter

A Bloom filter is a data structure used for answering approximate membership queries. That is, given a Bloom filter built from a set $\mathcal{S}$ [denoted by $\mathrm{BF}(\mathcal{S})$], one can determine whether an element $s_i \in \mathcal{S}$ or not with a bounded false-positive rate. In specific, a Bloom filter consists of a vector of Δ bits, i.e., $\{b_i\}_{i=1}^{\Delta}$, and f hash functions $\{h_i\}_{i=1}^{\mathsf{f}}$, where each hash function $h_i : \{0,1\}^* \mapsto \{1, 2, \ldots, \Delta\}$ maps an arbitrary input to a bit in the vector. Then, given a set $\mathcal{S}$, the corresponding Bloom filter $\mathrm{BF}(\mathcal{S})$ is built by: 1) initializing each bit in the vector to 0 and 2) setting positions $\{h_i(s_j) \mid i = 1, 2, \ldots, \mathsf{f}, s_j \in \mathcal{S}\}$ to 1. After that, $\mathrm{BF}(\mathcal{S})$ determines whether $s \in \mathcal{S}$ or not by testing whether all the corresponding locations $\{h_i(s) \mid i = 1, 2, \ldots, \mathsf{f}\}$ are 1s or not. If any of these bits is 0, then s is definitely not a member of $\mathcal{S}$; otherwise, $s \in \mathcal{S}$ with a false-positive rate less than $(1 - e^{-\mathsf{f} \cdot |\mathcal{S}|/\Delta})^{\mathsf{f}}$, where e is Euler's number.

IV. OUR PROPOSED SCHEME

In this section, we present our privacy-preserving cybertwin-based spatiotemporal keyword query (CSKQ) scheme, which consists of four parts, namely, system initialization, message publishing, message indexing, and query conducting.

A. System Initialization

In the system initialization, given security parameters $\{k_0, k_1, k_2\}$, the service provider generates the secret keys for all entities, and each vehicle initializes its own cybertwin. Specifically, the system initialization phase comprises the following five steps.

Step 1: With the security parameters $\{k_0, k_1, k_2\}$, the service provider runs the *key generation* algorithm of the SHE scheme to obtain $\{\mathsf{PP}, \mathsf{SK}\}$. After that, the service provider further computes an SHE ciphertext $E(-1)$, together with two ciphertexts of zero, i.e., $E(0)_1$ and $E(0)_2$.

Step 2: The service provider partitions the city map into several nonoverlapped regions $\mathcal{R}$, such that each region $R_i \in \mathcal{R}$ roughly has the same rate of new messages. Furthermore, the service provider selects a hash function $H(\cdot) : \{0,1\}^* \mapsto \{1, 2, \ldots, |\mathcal{R}|\}$ and a randomly generated key K_1 for $H(\cdot)$.

Step 3: The service provider generates a secret key K_2 for a symmetric-key encryption (SE) scheme, e.g., the advanced encryption standard (AES) scheme.

Step 4: The service provider publishes the public parameter PP, the hash function $H(\cdot)$ and three ciphertexts $\{E(-1),\ E(0)_1,\ E(0)_2\}$. Furthermore, the service

provider securely sends $\{K_1, K_2\}$ to the publishers and vehicles, and sends SK to CS_2.

Step 5: Upon receiving the secrets from the service provider, each vehicle deploys its own cybertwin. Specifically, it securely sends $\{K_1, K_2\}$ to its cybertwin and establishes a secure and reliable connection to the cybertwin.

B. Message Publishing

After the initialization, the publishers can publish messages to the authorized vehicles through the cloud platform. To protect the messages from the *honest-but-curious* cloud, each message needs to be encrypted. Specifically, on a specific date dt (e.g., 20200201), a publisher publishes a message $m_i = (c_i, \mathsf{L}_i, W_i, dt, t_i, \hat{t}_i)$ in the following three steps.

Step 1: The publisher encrypts m_i's content c_i, location L_i, and expire time $\hat{t}_i$ (e.g., 1620) with the secret key K_2, i.e., $\mathrm{SE}(K_2, c_i \| \mathsf{L}_i \| dt \| \hat{t}_i)$. Then, the publisher maps L_i to the corresponding region $R_{\mathsf{L}_i} \in \mathcal{R}$ and computes $H(K_1 \| dt \| R_{\mathsf{L}_i})$.

Step 2: The publisher builds a Bloom filter $\mathrm{BF}(\widetilde{W}_i)$, where $\widetilde{W}_i = \{\widetilde{w}_i = K_1 \| dt \| w_i \mid w_i \in W_i\}$ is constructed from m_i's keyword set W_i.

Step 3: The publisher further encrypts the expire time $\hat{t}_i$ into $E(\hat{t}_i)$ by computing

$$E(\hat{t}_i) = \hat{t}_i + r_1 \cdot E(0)_1 + r_2 \cdot E(0)_2 \bmod \mathcal{N} \quad (1)$$

where r_1 and $r_2 \in \{0, 1\}^{k_2}$ are random numbers. Then, it uploads $E(m_i) = \{\mathrm{SE}(K_2, c_i \| \mathsf{L}_i \| dt \| \hat{t}_i), H(K_1 \| dt \| R_{\mathsf{L}_i}), \mathrm{BF}(\widetilde{W}_i), dt, t_i, E(\hat{t}_i)\}$ to CS_1 via a secure channel.

C. Message Indexing

To organize the encrypted message $E(m_i)$ for efficiently supporting spatiotemporal keyword queries, CS_1 maintains a layered index, as shown in Fig. 4. In the index, based on the messages' released dates, they are first separated into groups, and each group of messages is further organized in $|\mathcal{R}|$ segment forests. Specifically, CS_1 creates a new group of segment forests on each day. Then, upon receiving an encrypted message $E(m_i)$, it indexes $E(m_i)$ in the current group by running the following steps.

Step 1: CS_1 locates the segment forest linked to m_i's hashed location $H(K_1 \| dt \| R_{\mathsf{L}_i})$, denoted by $\mathcal{SF}_{\mathsf{L}_i}$, and it creates an empty one if $\mathcal{SF}_{\mathsf{L}_i}$ does not exist.

Step 2: CS_1 appends $E(m_i)$ into the segment forest $\mathcal{SF}_{\mathsf{L}_i}$ as a segment tree whose height is 1. After that, to ensure that the resulting segment forest only contains trees of different heights, CS_1 adjusts the segment forest by recursively running the $\mathrm{ADJUST}(\mathcal{SF})$ in Algorithm 1 with the help of CS_2. The main idea of the algorithm is to test whether the last two trees, denoted as $\mathcal{SF}[-1]$ and $\mathcal{SF}[-2]$, in the forest are of the same height. If the two trees have different heights, then the heights of all trees in it are different and the algorithm stops. Otherwise, if the two trees have the same height, the algorithm builds a parent node for them, in which the Bloom filter, the release timestamp, and the expire time are respectively computed at lines 17–22. After computing the expire time of a new segment tree node, CS_1 needs to run Protocol 1 to obtain a refreshed ciphertext if the height of the new parent node can be divided by the multiplication depth σ. At the end of the algorithm, it recursively adjusts the resulting segment forest until all trees in the forest are of different heights.

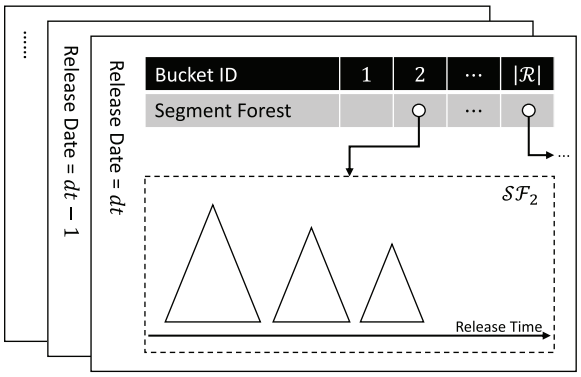


Fig. 4. Data structure for indexing messages, in which the messages published on the same date and in the same region will be organized in the same segment forest.

After running the above steps, messages are stored in the cloud and ready to be queried by the authorized vehicles.

D. Query Conducting

After the initialization, the authorized vehicles can access the messages published by the publishers. In specific, to receive messages that are related to itself, a vehicle V continuously synchronizes its location and keyword changes to its cybertwin. Based on the history trajectories and preferences of the vehicle and other contextual information, the cybertwin predicts the future locations of the vehicle. However, as we mainly focus on privacy preservation and the prediction will be conducted by cybertwin over plaintext information, we omit the details related to the prediction. Assume that the last time for the cybertwin to query the service is at time t_{old} of the date dt, for each predicted location $(\mathsf{L}_{v,k}, t_k)$, it queries the cloud in the following steps.

Step 1: The cybertwin maps the location $\mathsf{L}_{v,k}$ to the corresponding region $R_{v,k} \in \mathcal{R}$ and computes $H(K_1 \| dt \| R_{v,k})$.

Step 2: The cybertwin builds a Bloom filter $BF(\widetilde{W}_v)$, where $\widetilde{W}_v = \{\widetilde{w}_i = K_1 \| dt \| w_j \mid w_j \in W_v\}$ is constructed from the vehicle V's keyword set W_v.

Step 3: The cybertwin encrypts t_k as $E(t_k)$ by computing (1). Then, it submits $E(\mathcal{Q}) = (H(K_1 \| dt \| R_{v,k}), t_{\mathrm{old}}, E(t_k), \mathrm{BF}(\widetilde{W}_v), dt)$ to CS_1 through a secure channel.

Step 4: On receiving the query token $\mathcal{Q}$, CS_1 first locates the segment forest linked to $H(K_1 \| dt \| R_{v,k})$ in the

Algorithm 1 Segment Forest Adjustment

Input: The original segment forest $\mathcal{SF}_{\mathsf{L}_i} = [T_1, T_2, \cdots, T_s]$.
Output: The updated segment forest $\mathcal{SF}^*_{\mathsf{L}_i} = [T'_1, T'_2, \cdots, T'_{s'}]$

1: **return** $\mathcal{SF}^*_{\mathsf{L}_i}$ =ADJUST($\mathcal{SF}_{\mathsf{L}_i}$)

2: **function** ADJUST($\mathcal{SF}$)
3: // Compare the heights of the last two trees, denoted as $\mathcal{SF}[-1]$
4: // and $\mathcal{SF}[-2]$, in the forest
5: **if** height($\mathcal{SF}[-1]$) $\neq$ height($\mathcal{SF}[-2]$) **then**
6: // Do nothing when the last two trees have different heights
7: **return** $\mathcal{SF}$
8: **else**
9: // When the last two trees have the same height
10: right = $\mathcal{SF}$.removeLast()
11: left = $\mathcal{SF}$.removeLast()
12: // Build a new parent node for the two nodes
13: root = new Node()
14: root.left = left
15: root.right = right
16: // Computing $BF(W_i)$, t_i, $\hat{t}_i$ for the parent node
17: root.keywords = OR(left.keywords, right.keywords)
18: **if** left.release > right.release **then**
19: root.release = left.release
20: **else**
21: root.release = right.release
22: root.expireTime = MAX(left.expireTime, right.expireTime)
23: // Append the resulting tree into $\mathcal{SF}$
24: $\mathcal{SF}$.append(root)
25: // Recursively check the resulting segment forest
26: **return** ADJUST($\mathcal{SF}$)

27: **function** OR(left, right)
28: ret = new Boolean[left.length]
29: **for** i = 0; i < ret.length; i++ **do**
30: // Compute OR operation for each bit
31: ret[i] = left[i] + right[i] − left[i] ∗ right[i]
32: **return** ret

33: **function** MAX(left, right)
34: // Invoke the privacy-preserving sign calculating protocol
35: b = sign(left - right)
36: // max(left, right) can be computed with the encrypted bit b
37: max = b ∗ left + (E(0) + b) ∗ right
38: **return** max

Algorithm 2 Conducting Query

Input: The segment forest $\mathcal{SF}_{v,k}$, the query request $\mathcal{Q}$.
Output: The query result *Resp*

1: *Resp* = {} // Init *Resp* as an empty list
2: **for all** tree **in** $\mathcal{SF}_{v,k}$ **do**
3: *Resp* = *Resp* ∪ QUERYNODE(tree.root)
4: **return** *Resp*

5: **function** QUERYNODE(node)
6: // Whether the node has been visited in previous queries
7: **if** node.releaseDate < $\mathcal{Q}.t_{old}$ **then**
8: **return** ⊥ // Return an empty list
9:
10: // Whether the node's Bloom filter of keywords satisfies $BF(W_q)$
11: f_1 = VERIFYBF(node.keywords, $\mathcal{Q}$.keywords)
12: **if** $f_1 < 0$ **then**
13: **return** ⊥ // Return an empty list
14:
15: // Compare the node's expire time and $E(t_k)$
16: f_2 = sign(node.expireTime − $\mathcal{Q}.E(t_k)$)
17: $f = p\text{Sign}(f_2 + E(-1))$
18: **if** $f == 0$ **then**
19: **return** ⊥ // return an empty list
20: **else if** node is a *leaf node* **then**
21: // Return a singleton list containing the node's encrypted message
22: **return** {node.$E(m)$}
23: **else**
24: // Recursively query the node's children
25: l = QUERY(node.left)
26: r = QUERY(node.right)
27: **return** $l \cup r$

28: // Verifies whether $W_2 \subset W_1$ or not
29: **function** VERIFYBF($BF(W_1)$, $BF(W_2)$)
30: numOfOne = 0
31: numExpected = 0
32: **for** i = 0; i < len($BF(W_1)$); i++ **do**
33: numOfOne += $BF(W_1)[i] \cdot BF(W_2)[i]$
34: numExpected += $BF(W_2)[i]$
35: **return** numOfOne - numExpected

group of messages whose release date is *dt*, denoted as $\mathcal{SF}_{v,k}$.

Step 5: CS_1 runs Algorithm 2 with CS_2 to obtain the query result. The main idea is to query each segment tree in $\mathcal{SF}_{v,k}$. For each segment tree, the algorithm runs QUERYNODE() to obtain a list of leaf nodes that satisfy the query request $\mathcal{Q}$. Specifically, for each node, it respectively checks: 1) whether the latest release date of the messages in the node is larger than t_{old} (line 7); 2) whether the node's keyword set contains the query keywords (line 11); and 3) whether the latest expire date of these messages is larger than t_k (line 16). If either the first two conditions fail, the node and its children are pruned. Otherwise, CS_1 further verifies whether the third condition is satisfied by running a modified sign calculating protocol, denoted by pSign, to obtain $D(f_2) = p\text{Sign}(f_2 + E(-1))$, where $f_2 = \texttt{sign}(E(\hat{t}_i) - E(t_k))$. Similar to Protocol 2, $p\text{Sign}(x) = 1$ if $x \geq 0$, and $p\text{Sign}(x) = 0$ otherwise. The only difference between Protocol 2 and pSign is that, in *step 2* of pSign, CS_2 sends a plaintext 1 or 0 instead of $E(1)$ or $E(0)$ to CS_1. Then, CS_1 obtains $D(f_2) = 1$ if $b \neq D(\tilde{c})$; or $D(f_2) = 0$, otherwise. If the value of $D(f_2) = 1$, the algorithm recursively runs on the node's children or returns itself if it is a leaf node. Otherwise, the algorithm returns an empty list. In a nutshell, through this process, the algorithm first ignores the messages that are published before t_{old}. Then, it queries the rest part of the segment forest and prunes a specific node if it does not satisfy the criteria. Thus, generally, it will obtain a significantly improved performance.

Step 6: After running Algorithm 2, CS_1 obtains a set of messages Resp satisfying the query request $\mathcal{Q}$. Then, it sends Resp to the cybertwin.

Step 7: On receiving the query result, the cybertwin decrypts the messages. Then, it feeds the obtained messages to the driving assistance system and pushes them to the vehicle when it arrives at the corresponding location.

Correctness: The correctness of the privacy-preserving CSKQ scheme holds *iff* all messages satisfy: 1) the spatial criteria; 2) the keyword criteria; and 3) the temporal criteria of a query that will be included in the corresponding Resp.

Therefore, we show that a message will be excluded from Resp if it does not satisfy at least one of the three types of criteria as follows.

C-1: A message will be excluded from Resp if it does not satisfy the spatial criteria of the corresponding query. A message $m_i = (c_i, \mathsf{L}_i, W_i, dt, t_i, \hat{t}_i)$ is encrypted as

$$E(m_i) = \big(SE\big(K_2, c_i \| \mathsf{L}_i \| dt \| \hat{t}_i\big), H\big(K_1 \| dt \| R_{\mathsf{L}_i}\big) \\ \mathrm{BF}\big(\widetilde{W}_i\big), dt, t_i, E\big(\hat{t}_i\big)\big)$$

and uploaded to the cloud. Then, as detailed in Section IV, the cloud puts $E(m_i)$ into a bucket where all messages' $H(K_1\|dt\|R_{\mathsf{L}_i})$ are the same. While conducting an encrypted query $E(\mathcal{Q}) = (H(K_1\|dt\|R_{v,k}), t_{\text{old}}, E(t_k), \mathrm{BF}(\widetilde{W}_v), dt)$, CS_1 queries the segment forest where each message m_i in it satisfies that $H(K_1\|dt\|R_{\mathsf{L}_i}) = H(K_1\|dt\|R_{v,k})$. Since K_1 and dt are the same for valid messages and queries, the ranges of the message and the query, i.e., $R_{v,k}$ and R_{L_i} should also be the same with a considerably high probability, based on the property of $H(\cdot)$. Thus, a message will be excluded from Resp if it does not satisfy the spatial criteria of the query.

C-2: A message will be excluded from Resp if it does not satisfy the keyword criteria of the corresponding query. While encrypting a message m_i, the corresponding publisher builds a Bloom filter $\mathrm{BF}(\widetilde{W}_i)$ to represent m_i's keywords W_i. Similarly, a query's keywords set W_q is also represented as $BF(\widetilde{W}_q)$ by the cybertwin. Then, as demonstrated in VERIFYBF in Algorithm 2, the function computes

$$f_1 = \sum_{i=1}^{\ell} \big(\mathrm{BF}\big(\widetilde{W}_i\big)[i] \cdot \mathrm{BF}\big(\widetilde{W}_q\big)[i] - \mathrm{BF}\big(\widetilde{W}_q\big)[i]\big)$$

where $\ell = \mathrm{len}(\mathrm{BF}(\widetilde{W}_i))$. The value f_1 indicates whether all positions of 1s in $\mathrm{BF}(\widetilde{W}_q)$ are also 1s in $\mathrm{BF}(\widetilde{W}_i)$, i.e., whether $W_q \subset W_i$. In specific, if the value $f_1 < 0$, we know $W_q \not\subset W_i$ and the message m_i does not satisfy the query request. Otherwise, the value $f_1 = 0$, and we know m_i satisfies the query request with a negligible false-positive rate. Thus, a message will be excluded from Resp if it does not satisfy the keyword criteria of the corresponding query.

C-3: A message will be excluded from Resp if it does not satisfy the temporal criteria of the corresponding query. The release timestamp t_i is directly sent to CS_1 while the expire time $\hat{t}_i$ of a message m_i is encrypted as $E(\hat{t}_i)$. Then, the cloud checks whether m_i satisfies a query $\mathcal{Q}$ or not by testing whether $t_i < t_{\text{old}}$ and $\hat{t}_i < t_{v,k}$, respectively, on lines 7 and 16 in Algorithm 2. Thus, a message will be excluded from Resp if it does not satisfy the temporal criteria of the corresponding query.

From the above analysis, the correctness of the proposed privacy-preserving CSKQ scheme holds.

V. Security Analysis

In this section, we analyze the security of our proposed scheme. Specifically, we first show that SHE ciphertexts obtained through (1) have the same security as the original SHE scheme [30]. Then, following our security model, we, respectively, show that our proposed scheme can preserve the privacy of the published messages and the query requests.

A. Security of SHE Ciphertexts Obtained Through (1)

We prove the SHE ciphertexts obtained through (1) are IND-CPA secure upon the extended $(\mathcal{L}, p)$-based decision assumption. Before that, we first recall two subsets of $\mathbb{Z}_{\mathcal{N}}$ defined in [30] as follows:

$$\begin{cases} \mathbb{S} = \{\alpha\mathcal{L} + \beta p \bmod \mathcal{N} \mid \alpha, \beta \in \mathbb{Z}_{\mathcal{N}}, \alpha\mathcal{L} < p\} \\ \overline{\mathbb{S}} = \mathbb{Z}_{\mathcal{N}} \setminus \mathbb{S} : \{\alpha\mathcal{L} + \beta p \bmod \mathcal{N} \mid \alpha, \beta \in \mathbb{Z}_{\mathcal{N}}, \alpha\mathcal{L} \geq p\} \end{cases}$$

and introduce our extended $(\mathcal{L}, p)$-based decision problem and its assumption. Here, let us keep in mind that the bit lengths $|p| = k_0$ and $|\mathcal{L}| = k_2$.

Lemma 1: Given security parameters k_0 and k_2 satisfying $3k_2 < k_0$ and two elements $\{e_i\}_{i=1}^2$ in $\mathbb{S}$, where each

$$e_i = \alpha_i\mathcal{L} + \beta_i p \bmod \mathcal{N}, \text{ with } \alpha_i \in \{0,1\}^{k_2}, \beta_i \in \mathbb{Z}_{\mathcal{N}}$$

a new value $x \leftarrow XGen(e_1, e_2)$, i.e.,

$$x = r_1 e_1 + r_2 e_2 \bmod \mathcal{N} \tag{2}$$

generated by (e_1, e_2) with two random numbers $r_1, r_2 \in \{0,1\}^{k_2}$ is also a valid element of $\mathbb{S}$.

Proof: By expanding (2), we have

$$\begin{aligned} x &= (r_1\alpha_1 + r_2\alpha_2) \cdot \mathcal{L} + (r_1\beta_1 + r_2\beta_2) \cdot p \bmod \mathcal{N} \\ &= \alpha_x\mathcal{L} + \beta_x p \bmod \mathcal{N} \end{aligned}$$

where $\beta_x = r_1\beta_1 + r_2\beta_2 \bmod \mathcal{N} \in \mathbb{Z}_{\mathcal{N}}$, $\alpha_x = (r_1\alpha_1 + r_2\alpha_2)$. As the bit lengths of r_i, α_i, and $\mathcal{L}$ are both k_2, the bit length of $\alpha_x\mathcal{L}$ is $3k_2 < k_0$, where k_0 is the bit length of p. Hence, we have $\alpha_x\mathcal{L} < p$, and $x \in \mathbb{S}$. ∎

Definition 1 [Extended $(\mathcal{L}, p)$-Based Decision Problem]: Given $(k_0, k_2, \mathcal{N}, e_1, e_2)$, flipping a coin $z \in \{0,1\}$. If $z = 0$, generate $x \in \mathbb{S} \leftarrow XGen(e_1, e_2)$; if $z = 1$, randomly draw $x \in \overline{\mathbb{S}}$. The extended $(\mathcal{L}, p)$-based decision problem is to determine whether $x \in \mathbb{S}$ or not for a given tuple $(k_0, k_2, \mathcal{N}, e_1, e_2, x)$, i.e., to determine the value of z without knowing $(p, q, \mathcal{L}, r_1, r_2)$.

Here, we demonstrate the intractability of the extended $(\mathcal{L}, p)$-based decision problem. Without knowing $(p, q, \mathcal{L}, r_1, r_2)$, as analyzed in [30], the distinguisher $\mathcal{B}$ can obtain $\{p, q, \mathcal{L}\}$ only by exhausting spaces of p and $\mathcal{L}$, i.e., $\{0,1\}^{k_0}$ and $\{0,1\}^{k_2}$. Then, except $\{p, q, \mathcal{L}\}$, the distinguisher $\mathcal{B}$ can solve the problem only by searching the values of the two random numbers r_1 and r_2. Specifically, given a problem instance $(k_0, k_2, \mathcal{N}, e_1, e_2, x)$, $\mathcal{B}$ first exhausts $r_1 \in \{0,1\}^{k_2}$, and for each r_1, it verifies whether

$$r_2 = (x - r_1 \cdot e_1) \cdot e_2^{-1} \bmod \mathcal{N} \stackrel{?}{\in} \{0,1\}^{k_2}.$$

If the resulting $r_2 \in \{0,1\}^{k_2}$, the corresponding pair of (r_1, r_2) is a correct guess, and $\mathcal{B}$ knows $x \in \mathbb{S}$. Therefore, we need to choose proper parameters $\{k_0, k_2\}$ to ensure the extended $(\mathcal{L}, p)$-based decision problem is intractable.

Definition 2 [Extended $(\mathcal{L}, p)$-Based Decision Assumption]: The Extended $(\mathcal{L}, p)$-Based Decision Problem

is hard if, for any polynomial time algorithm, its advantage in solving the problem is a negligible function in k_0 and k_2.

Theorem 1: A SHE ciphertext obtained through (1) is IND-CPA secure under the extended $(\mathcal{L}, p)$-based decision assumption.

Proof: Assume that there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ that has a non-negligible advantage ε to break the IND-CPA security of the ciphertexts obtained through (1). Then, we can construct a distinguisher $\mathcal{B}$ that exploits $\mathcal{A}$'s ability to break the extended $(\mathcal{L}, p)$-based decision problem with another nonnegligible advantage. Specifically, $\mathcal{B}$ and $\mathcal{A}$ interact as follows.

1) With an extended $(\mathcal{L}, p)$-based decision problem instance $(k_0, k_2, \mathcal{N}, e_1, e_2, x)$, $\mathcal{B}$ chooses k_1 such that $k_1 \ll k_2$ and sets the message space $\mathcal{M} = [-2^{k_1-1}, 2^{k_1-1})$. Then, $\mathcal{B}$ sends $(k_0, k_1, k_2, \mathcal{N}, E(0)_1 = e_1, E(0)_2 = e_2)$ to $\mathcal{A}$.
2) Upon receiving $(k_0, k_1, k_2, \mathcal{N}, E(0)_1, E(0)_2)$, $\mathcal{A}$ chooses two messages $m_0, m_1 \in \mathcal{M}$ and sends them to $\mathcal{B}$. On receiving $\{m_0, m_1\}$, $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$, computes $c = m_b + x \bmod \mathcal{N}$, and sends c as a ciphertext to $\mathcal{A}$.
3) On receiving c, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ as its guess on b. Then, $\mathcal{B}$ outputs $z' = 0$ as its guess on z if $b' = b$, and $z' = 1$ otherwise.

Obliviously, when $z = 0$, i.e., $x \in \mathbb{S}$, the ciphertext $c = m + \alpha\mathcal{L} + \beta p \bmod \mathcal{N}$ is a valid ciphertext. Thereby, $\mathcal{A}$ can exert its advantage and correctly guess b with probability $\Pr[b' = b \mid z = 0] = (1/2) + \varepsilon$, so $\Pr[\mathcal{B}$ success $\mid z = 0] = (1/2) + \varepsilon$. On the other hand, when $z = 1$, the ciphertext c is not a valid ciphertext, and $\mathcal{A}$ can only randomly make its guess. Thus, we have $\Pr[\mathcal{B}$ success $\mid z = 1] = \Pr[b' = b \mid z = 1] = (1/2)$. Summarizing the above two cases, we have $\Pr[\mathcal{B}$ success$] = (1/2) \times ([1/2] + \varepsilon) + (1/2) \times (1/2) = (1/2) + (\varepsilon/2)$. Since ε is nonnegligible, the above result contradicts with the extended $(\mathcal{L}, p)$-based decision assumption. Thus, the ciphertext obtained through (1) is IND-CPA secure under the extended $(\mathcal{L}, p)$-based decision assumption. ■

B. Security of the Proposed Scheme

Published Messages Are Privacy Preserving: As detailed in Section IV, a message m_i is encrypted into $E(m_i) = (\mathrm{SE}(K_2, c_i \| \mathsf{L}_i \| dt \| \hat{t}_i), H(K_1 \| dt \| R_{\mathsf{L}_i}), \mathrm{BF}(\widetilde{W}_i), dt, t_i, E(\hat{t}_i))$ by the corresponding publisher. Then, the message is uploaded to CS_1, and the latter inserts $E(m_i)$ into the index with the help of CS_2. Since CS_1 and CS_2 are *honest-but-curious* and *non-collusive*, we, respectively, show that the encrypted message is secure against them as follows.

For CS_1: After system initialization, CS_1 has access to PP and $\{E(-1), E(0)_1, E(0)_2\}$. On the one hand, since CS_1 does not know the secret keys K_1 and K_2, it can neither recover R_{L_i} from $H(K_1 \| dt \| R_{\mathsf{L}_i})$, enumerate all possible combinations of keywords, nor decrypt $SE(K_2, c_i \| \mathsf{L}_i \| dt \| \hat{t}_i)$. Furthermore, as each element in $\mathrm{BF}(\widetilde{W}_i)$ contains dt, without knowing K_1, one cannot link the keyword Bloom filters $\mathrm{BF}(\widetilde{W}_i)$ on different dates dt even they are built from the same set of keywords. Similarly, the hashed regions $H(K_1 \| dt \| R_{\mathsf{L}_i})$ generated for different dates dt cannot be linked, which makes it harder for CS_1 to infer the location of the encrypted message. On the other hand, following (1), the SHE ciphertext $E(\hat{t}_i)$ is encrypted and processed with $\{E(-1), E(0)_1, E(0)_2\}$, which is also obtained by CS_1. However, based on the security of the original SHE scheme and Theorem 1, CS_1 cannot obtain the corresponding plaintext. Furthermore, while indexing the message $E(m_i)$, CS_1 runs Algorithm 1 with the help of CS_2. Nevertheless, during running the algorithm, CS_1 can only receive the encrypted maximum $E(\max(\mathrm{left}.\hat{t}, \mathrm{right}.\hat{t}))$, but it cannot know the corresponding plaintext. Thus, the plaintexts, locations, and keywords of the published messages are privacy preserving against CS_1.

For CS_2: Although CS_2 holds SK, it cannot directly access the encrypted messages. That is, CS_2 can only receive a ciphertext $\tilde{c} = E(r_1 \cdot m + r_2)$ during each invocation of the privacy-preserving sign calculating protocol, but it cannot recover the plaintext m without knowing r_1 and r_2. Thus, the encrypted messages are privacy-preserving against CS_2. Therefore, the plaintexts, locations, and keywords published messages are privacy-preserving under our considered model.

Query Requests Are Privacy Preserving: While launching a query $\mathcal{Q} = \{\mathsf{L}_q, t_q, \hat{t}_q, W_q, dt\}$, a cybertwin encrypts it to be $E(\mathcal{Q}) = \{H(K_1 \| dt \| \mathsf{L}_q), t_q, E(\hat{t}_q), BF(\widetilde{W}_q), dt\}$. Then, CS_1 receives $E(\mathcal{Q})$ and conducts the query with the help of CS_2. Hence, we show the query requests are privacy-preserving against the two *honest-but-curious* cloud servers $\{\mathrm{CS}_1, \mathrm{CS}_2\}$.

For CS_1: On the one hand, similar to the encrypted messages, without knowing K_1 and SK, CS_1 cannot recover the plaintexts of $H(K_1 \| dt \| \mathsf{L}_q)$, $BF(\widetilde{W}_q)$ or $E(\hat{t}_q)$ in the encrypted query request $E(\mathcal{Q})$. Furthermore, CS_1 cannot link them on different release dates dt to infer the location of the query. On the other hand, during conducting the query, CS_1 runs Algorithm 2 with the help of CS_2, and it receives several plaintexts as comparison results. Specifically, in each invocation of the QUERYNODE function, CS_1 receives one plaintext from CS_2. However, the plaintext only reveals whether the node satisfies the expire time requirement or not. Therefore, the *honest-but-curious* CS_1 cannot obtain the location and keywords of the query request.

For CS_2: Before query conducting, since the query request is uploaded to CS_1, CS_2 cannot receive the ciphertexts or plaintexts of the query request. While helping CS_1 to conduct the query, CS_2 receives several SHE ciphertexts through the pSign protocol. Although CS_2 can decrypt them with SK, each of the obtained plaintexts contains two random numbers r_1 and r_2. Without knowing these two random numbers, CS_2 cannot obtain the actual plaintext, which is

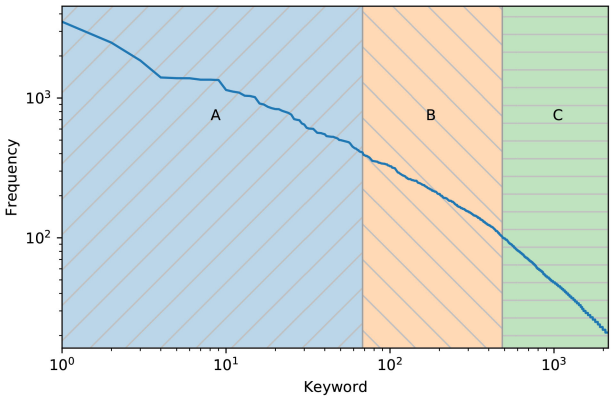


Fig. 5. Sorted frequency distribution of the keywords extracted from the data set. Based on their frequencies, we roughly group the keywords into three categories, namely, A, B, and C.

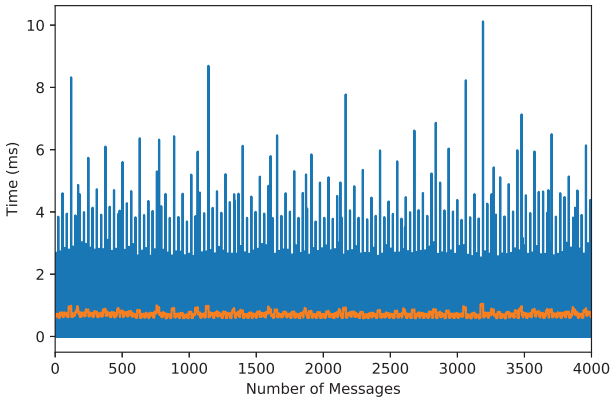


Fig. 6. Time consumption for the cloud to index one message versus different numbers of messages in the corresponding segment forest.

the intermediate information related to the query. Thereby, CS_2 cannot obtain the plaintext of the query request. Thus, the locations and keywords of query requests are privacy preserving under our considered model.

From the above analysis, we can see the proposed spatiotemporal keyword query scheme can indeed preserve the privacy of the published messages and the query requests.

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed privacy-preserving spatiotemporal keyword query scheme. To this end, we implemented our scheme in Java. In the implementation, we set the security parameters $k_0 = 2048$, $k_1 = 32$, and $k_2 = 160$ and we choose AES as the underlying SE scheme. In the implementation, we use the Bloom filter provided by Google Guava,[1] and the number of expected insertions and the false-positive probability are respectively set to 200 and 0.01. Then, we conduct several experiments on an Intel Core i5-1038NG7 CPU @2.00GHz, 16-GB RAM and macOS 11.2 operating system over a data set extracted from a dump file of Wikipedia [31]. Specifically, we first extract 14 464 documents from a wikidump file and exclude the documents whose length is smaller than 500. After that, we respectively extract the location and keywords of each document with the help of spaCy [32], where the keywords are the most frequently appeared nouns, proper nouns, or adjectives. As shown in Fig. 5, we partition the keywords into three categories based on the number of documents including them. Then, we demonstrate the performance of our proposed scheme in message publishing and query conducting as follows.

A. Message Publishing and Indexing

As detailed in Section IV-B, to publish a message m_i, CS_1 runs one round of SE and one round of SHE encryption through (1). Therefore, the computational complexity for a publisher to encrypt a message is $O(1)$. According to our experiment, it only takes averagely 153.37 μs for a publisher to encrypt a message. Then, the publisher uploads the encrypted message to CS_1, and the latter indexes the message by integrating it into the corresponding segment forest through Algorithm 1. We can easily see that CS_1 and CS_2 need to jointly create multiple parent nodes with a relatively low probability. Specifically, to insert an encrypted message into the index, the two cloud servers jointly create no more than two nodes on average. In Fig. 6, we plot the time consumption for the cloud to add messages into the index and an orange line indicating the moving average of the time consumption with the sliding window length of 20. As shown in the figure, although a small portion of insertions take less than 10 ms to complete, the moving average time consumption is less than 1 ms. Thus, message publishing and indexing in our proposed scheme is efficient for both the publishers and the cloud.

B. Query Conducting

To launch a spatiotemporal keyword query $\mathcal{Q}$, a cybertwin first encrypts it as $E(\mathcal{Q})$ by computing (1) to conduct one SHE encryption. According to our experiment, it only takes 57.30 μs for a cybertwin to encrypt a query request. Then, CS_1 and CS_2 together run Algorithm 2, and the computational cost is affected by the number of nodes visited by the algorithm. According to our experiment, the average time consumption for verifying a node is 111.41 μs, and the number of visited nodes will be affected by the number of nodes in the bucket that satisfying the query request. Then, we draw Figs. 7 and 8 to further demonstrate the relationship between the average time consumption and several parameters related to the bucket and queries.

In Fig. 7, we plot the relationship between the number of messages in the bucket n, the popularity of query keywords, the cybertwin's last query time t_{old}, and the average time consumption for query conducting. Specifically, to demonstrate the effect of the popularity of query keywords on query time consumption, we construct queries where each query contains one keyword from one of the three categories in Fig. 5. Furthermore, to demonstrate the effect of t_{old} on different sizes of buckets, we convert t_{old} to be (t_{old}/n), i.e., the percentage of messages excluded by their release time t. As shown in the

[1] https://guava.dev

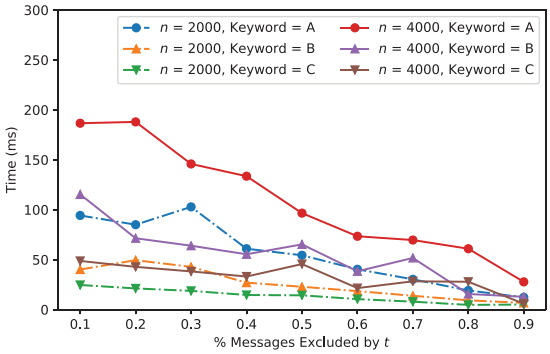


Fig. 7. Average time consumption for the cloud conduct a query that contains a keyword from one of the three categories over a bucket with $n = 2000$ and 4000 with varied (t_{old}/n), i.e., the percentage of messages excluded by their release times.

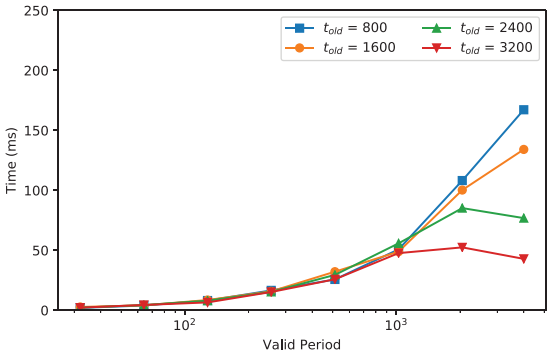


Fig. 8. Average time consumption for the cloud conduct a query with one keyword from category A over a bucket of 4000 messages with varied valid periods, i.e., $\hat{t}_i - t_i$ of each message m_i.

figure, the average time consumption for the cloud servers to conduct a query increases with n, i.e., the number of messages in the bucket. Furthermore, it reduces as the frequency of the selected keywords reducing and (t_{old}/n) increasing.

In Fig. 8, we plot the relationship between messages' valid period and the average time consumption for the cloud to conduct a query. Specifically, the bucket contains 4000 messages, where the valid period of each message m_i, i.e., $\hat{t}_i - t_i$, ranges from 32 to 4000 minutes. The query's t_{old} ranges from 800 to 3200, and it contains a random keyword from category of keywords with highest frequency of appearance in the data set, i.e., category A in Fig. 5. Furthermore, the query time t_q is set to 4000. As shown in the figure, the average time consumption for the cloud servers to conduct a query increases with the valid period. This is mainly because that, as the valid period increases, the number of valid messages may increase. However, since a larger amount of messages will be excluded as t_{old} increases, the time consumption will not keep increasing when t_{old} is greater than $\hat{t}_i - t_i$.

VII. Related Work

In this section, we review some related works, which are closely related to our proposed scheme in terms of privacy-preserving spatiotemporal range queries on outsourced data sets and privacy-preserving keyword queries on dynamic data sets.

Many schemes were proposed to support privacy-preserving spatial or spatio-textual queries over an outsourced data set. On the one hand, many schemes were proposed to support privacy-preserving multidimensional or spatial range queries and kNN queries. Focusing on privacy-preserving spatial kNN queries, Wong *et al.* [9] proposed the asymmetric scalar-product preserving encryption scheme (ASPE) and employed it to achieve distance calculation, but its security is not that strong. By presenting a hierarchical index named $\widehat{R}$-tree encrypted by ASPE, Wang and Ravishankar [10] proposed a scheme to achieve privacy-preserving multidimensional range queries. To obtain better security, Chi *et al.* [11] presented an enhanced ASPE scheme and built a privacy-preserving multidimensional range query scheme by securely detecting rectangle intersection. Cui *et al.* [12] further improved the security by building the scheme upon the Paillier public-key encryption scheme. Then, Elmehdwi *et al.* [13] built a Paillier-based secure kNN query scheme by designing several secure protocols. This scheme can well preserve the data privacy of the data set and the query but introduces heavy computational costs. To achieve better efficiency, Boldyera *et al.* [14] and Choi *et al.* [15], respectively, proposed order-preserving encryption (OPE) and mutable order-preserving encoding (mOPE) to efficiently support secure multidimensional kNN queries, but they do not protect the order relationship of records and may invoke privacy concerns. On the other hand, there are also many schemes focus on privacy-preserving spatio-textual queries. Su *et al.* [16] built a privacy-preserving top-k spatial keyword query scheme based on an IR-Tree index. Later, they improved their work in [17] to achieve better query performance. Cui *et al.* [18] proposed a privacy-preserving scheme to support boolean spatio-textual queries based on an R-tree index. Negi *et al.* [19] proposed a privacy-preserving top-k spatio-textual query scheme for smart city scenarios, and a quad-tree-based index is employed to index the data set. However, these schemes [9]–[19] can only support static data sets and cannot handle queries containing all spatial, temporal, and textual criteria. Therefore, these schemes are not applicable in our scenario.

There are also some schemes proposed for achieving textual queries over dynamic data sets. Kamara *et al.* [20] proposed a dynamic searchable symmetric encryption (DSSE) scheme, but it can only support single-keyword queries. They further improved query efficiency through parallelism in [21]. Naveed *et al.* [22] designed a primitive named blind storage to support light-weight single keyword queries over a dynamic data set. Kim *et al.* [23] constructed an efficient forward secure DSSE scheme for single-keyword queries by employing a dual-dictionary index. These schemes only support single-keyword queries and are not applicable in our scenario. Furthermore, some schemes focus on privacy-preserving multikeyword queries over dynamic data sets. Xia *et al.* [24] proposed a privacy-preserving multikeyword rank query scheme where all records are organized a keyword balanced binary tree. Du *et al.* [25] designed a privacy-preserving multikeyword query scheme with both symmetric and public-key constructions. Zheng *et al.* [26] proposed a privacy-preserving conjunctive keyword query scheme that

naturally supports queries over dynamic data. However, some of them were built upon the symmetric encryption scheme and do not support multiple publishers. Some of them can support multiple publishers but still cannot efficiently handle spatial and temporal criteria. Therefore, they cannot apply our scenario.

As discussed above, many schemes proposed to address queries containing spatial, textual, or spatio-textual criteria, and some of them support queries over dynamic data sets. However, none of them can simultaneously support queries containing both spatial, temporal, and keyword criteria. Therefore, they are not applicable in our scenario.

VIII. Conclusion

In this article, we have presented a privacy-preserving cybertwin-based spatiotemporal keyword query for the ITS scenario. Specifically, we first designed a layered index to dynamically organize messages containing all spatial, temporal, and keyword information, and the index can efficiently support queries over these messages. Then, we presented two algorithms for, respectively, encrypting the index and queries into ciphertexts of the SHE scheme without knowing the secret key. After that, we presented our privacy-preserving cybertwin-based spatiotemporal keyword query for the ITS scenario, which can conduct encrypted queries over the encrypted index. Meanwhile, security analysis demonstrated that our scheme can preserve the privacy of the messages and the queries. In addition, performance evaluation showed that our scheme is indeed computationally efficient. In our future work, we will further evaluate the efficiency of our scheme in real-world application scenarios.

References

[1] Grand View Research. (May 2020). *Global 5G Services Market Size Report, 2021-2027*. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/5g-services-market

[2] P. Jia, X. Wang, and X. Shen, "Digital-twin-enabled intelligent distributed clock synchronization in industrial IoT systems," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4548–4559, Mar. 2021.

[3] Q. Yu, J. Ren, Y. Fu, Y. Li, and W. Zhang, "Cybertwin: An origin of next generation network architecture," *IEEE Wireless Commun.*, vol. 26, no. 6, pp. 111–117, Dec. 2019.

[4] Q. Yu, J. Ren, H. Zhou, and W. Zhang, "A cybertwin based network architecture for 6G," in *Proc. 2nd 6G Wireless Summit (6G SUMMIT)*, Levi, Finland, Mar. 2020, pp. 1–5.

[5] R. Lu, "A new communication-efficient privacy-preserving range query scheme in fog-enhanced IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2497–2505, Apr. 2019.

[6] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving *k*-NN query for outsourced ehealthcare data," *J. Med. Syst.*, vol. 43, no. 5, pp. 1–13, 2019.

[7] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving efficient and privacy-preserving exact set similarity search over encrypted data," *IEEE Trans. Dependable Security Comput.*, early access, Jun. 23, 2020, doi: 10.1109/TDSC.2020.3004442.

[8] Z. Wang *et al.*, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *Proc. 91st IEEE Veh. Technol. Conf. (VTC-Spring)*, Antwerp, Belgium, May 2020, pp. 1–6.

[9] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Providence, RI, USA, 2009, pp. 139–152.

[10] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using Rp-trees," in *Proc. 29th IEEE Int. Conf. Data Eng. (ICDE)*, Brisbane, QLD, Australia, Apr. 2013, pp. 314–325.

[11] J. Chi, C. Hong, M. Zhang, and Z. Zhang, "Fast multi-dimensional range queries on encrypted cloud databases," in *Proc. 22nd Int. Conf. Database Syst. Adv. Appl.*, vol. 10177. Suzhou, China, Mar. 2017, pp. 559–575.

[12] N. Cui, X. Yang, B. Wang, J. Geng, and J. Li, "Secure range query over encrypted data in outsourced environments," *World Wide Web*, vol. 23, no. 1, pp. 491–517, 2020.

[13] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Chicago, IL, USA, 2014, pp. 664–675.

[14] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," IACR Cryptol. ePrint Archives, Lyon, France, Rep. 2012/624, 2012.

[15] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino, "Secure kNN query processing in untrusted cloud environments," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2818–2831, Nov. 2014.

[16] S. Su, Y. Teng, X. Cheng, Y. Wang, and G. Li, "Privacy-preserving top-k spatial keyword queries over outsourced database," in *Proc. 20th Int. Conf. Database Syst. Adv. Appl.*, vol. 9049. Hanoi, Vietnam, Apr. 2015, pp. 589–608.

[17] S. Su, Y. Teng, X. Cheng, K. Xiao, G. Li, and J. Chen, "Privacy-preserving top-k spatial keyword queries in untrusted cloud environments," *IEEE Trans. Services Comput.*, vol. 11, no. 5, pp. 796–809, Sep./Oct. 2018.

[18] N. Cui, J. Li, X. Yang, B. Wang, M. Reynolds, and Y. Xiang, "When geo-text meets security: Privacy-preserving Boolean spatial keyword queries," in *Proc. 35th IEEE Int. Conf. Data Eng. (ICDE)*, Macao, China, Apr. 2019, pp. 1046–1057.

[19] D. Negi, S. Ray, and R. Lu, "Pystin: Enabling secure LBS in smart cities with privacy-preserving top-*k* spatial-textual query," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7788–7799, Oct. 2019.

[20] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Raleigh, NC, USA, Oct. 2012, pp. 965–976.

[21] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. 17th Int. Conf. Financ. Cryptogr. Data Security*, vol. 7859. Okinawa, Japan, Apr. 2013, pp. 258–274.

[22] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *Proc. IEEE Symp. Security Privacy (SP)*, Berkeley, CA, USA, 2014, pp. 639–654.

[23] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, Dallas, TX, USA, 2017, pp. 1449–1463.

[24] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.

[25] M. Du, Q. Wang, M. He, and J. Weng, "Privacy-preserving indexing and query processing for secure dynamic cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2320–2332, Sep. 2018.

[26] Y. Zheng, R. Lu, J. Shao, F. Yin, and H. Zhu, "Achieving practical symmetric searchable encryption with search pattern privacy over cloud," *IEEE Trans. Services Comput.*, early access, May 4, 2020, doi: 10.1109/TSC.2020.2992303.

[27] R. Li, A. X. Liu, H. Xu, Y. Liu, and H. Yuan, "Adaptive secure nearest neighbor query processing over encrypted data," *IEEE Trans. Dependable Security Comput.*, early access, May 28, 2020, doi: 10.1109/TDSC.2020.2998039.

[28] M. Du, S. Wu, Q. Wang, D. Chen, P. Jiang, and A. Mohaisen, "GraphShield: Dynamic large graphs for secure queries with forward privacy," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 18, 2020, doi: 10.1109/TKDE.2020.3024883.

[29] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving $O(\log^3 n)$ communication-efficient privacy-preserving range query in fog-based IoT," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5220–5232, Jun. 2020.

[30] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Trans. Dependable Security Comput.*, early access, Feb. 23, 2021, doi: 10.1109/TDSC.2021.3061611.

[31] Wikimedia. *Wikimedia Downloads*. Accessed: Jan. 2021. [Online]. Available: https://dumps.wikimedia.org/

[32] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. (2020). *spaCy: Industrial-Strength Natural Language Processing in Python*. [Online]. Available: https://doi.org/10.5281/zenodo.1212303

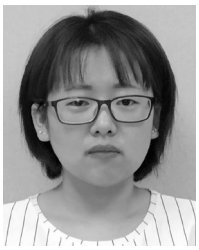
Yunguo Guan is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada.

His research interests include applied cryptography and game theory.

Rongxing Lu (Fellow, IEEE) received the first Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2006, and the second Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012.

He worked as an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, from April 2013 to August 2016. He is an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Fredericton, NB, Canada. He worked as a Postdoctoral Fellow with the University of Waterloo from May 2012 to April 2013. He has published extensively in his areas of expertise (with H-index 76 from Google Scholar as of May 2021). His research interests include applied cryptography, privacy enhancing technologies, and IoT-big data security and privacy.

Dr. Lu was awarded the most prestigious "Governor General's Gold Medal," from the University of Waterloo in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia–Pacific Outstanding Young Researcher Award, in 2013. He was the recipient of nine best (student) paper awards from some reputable journals and conferences. He currently serves as the Vice-Chair (Conferences) for IEEE ComSoc Communications and Information Security Technical Committee. He is the Winner of 2016–2017 Excellence in Teaching Award, FCS, UNB.



Yandong Zheng received the M.S. degree from the Department of Computer Science, Beihang University, Beijing, China, in 2017. She is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada.

Her research interest includes cloud computing security, big data privacy, and applied privacy.

Songnian Zhang received the M.S. degree from Xidian University, Xi'an, China, in 2016. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada.

His research interest includes cloud computing security, big data query, and query privacy.

Jun Shao (Associate Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008.

He was a Postdoctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.

Guiyi Wei (Member, IEEE) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in December 2006, where he was advised by Cheung Kong Chair Prof. Yao Zheng.

He is a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou. His research interests include wireless networks, mobile computing, cloud computing, social networks, and network security.