IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 6, 2021, accepted February 18, 2021, date of publication February 24, 2021, date of current version March 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3061676

An Efficient Ciphertext Index Retrieval Scheme Based on Edge Computing Framework

JINTAO LING, HUIXIAN GU, HAIJIANG WANG, AND LEI ZHANG
School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China
Corresponding authors: Huixian Gu (kyyghx@aliyun.com) and Haijiang Wang (wanghaijiangyes@163.com)

This work was supported in part by the Natural Science Foundation of Zhejiang Province under Grant LQ20F020010.

ABSTRACT With the rapid development of mobile applications, more and more traffic is generated at the network's edge and forwarded between many users. The explosive growth of network traffic has imposed massive pressure on traditional network architectures. At the same time, users have increasing data security requirements because of frequent data breaches. Mobile edge storage is an emerging computing framework that ensures users enjoy a high quality of experience when they access cloud services and is gradually becoming the key technology to solve the above problems. In this paper, by exploiting searchable encryption and cooperative edge computing, we proposed an efficient ciphertext index retrieval scheme to tackle three issues simultaneously in a secure and efficient data search service scenario: (1) reducing data transportation latency to improve mobile user's quality of experience; (2) mitigating data traffic pressure on the backbone network; (3) guaranteeing the security of the data when users search data in the edge network. Simulation results show that our scheme can save about 80% of backbone network traffic than the traditional cloud computing scheme. It can also reduce network latency by approximately 30% for users.

INDEX TERMS Searchable encryption, cache hit radio, non-cooperative game theory, cache replacement strategy, edge computing.

I. INTRODUCTION

With the rapid development of mobile applications, mobile communications have generated more and more traffic between network edges and users. The explosive growth of network traffic has put tremendous pressure on traditional network architecture. Mobile edge storage is an emerging computing framework when users enjoy a high-quality experience when accessing cloud services. It has gradually become a key technology to solve the above problem. Forecasts predict that the world's mobile data traffic will surpass 90% of mobile data traffic and reach 77.5 monthly exabytes by 2022 [1]. Such explosive traffic has exerted a heavy burden on the current network architecture [2]. With the frequent interaction of these massive data, information leakage and data privacy have gradually become the focus of attention in mobile edge storage system.

As a new encryption technology, searchable encryption is favored by governments and enterprises, which enables users to search over encrypted data without exposing the contents of messages or the searched keyword to cloud storage operators. This has a huge appeal for governments and companies with a need for secrecy. Through many researchers' efforts in the field of encryption, many efficient searchable encryption schemes have been formed [3]–[22].

An important research direction of searchable encryption is symmetric searchable encryption (SSE), which has a low computational cost, reduced algorithm, fast speed, and is closer to the practical application scene. A classic SSE system is as follows [23]:

1) Users first extract keywords from local files and construct the index, then encrypt the index and files with the private key, and then upload them to the cloud server.
2) Users with query permissions use the private key to generate a trapdoor for the keyword that needs to be queried and then sent to the server. The trapdoor cannot reveal any information about the keyword. The simplest trapdoor is the keyword encrypted with the key.
3) The server executes the retrieval algorithm after receiving the trapdoor. The retrieval algorithm finds the encrypted file name corresponding to the trapdoor's keyword from the inverted index structure. Finally, the encrypted file corresponding to the user's file name

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

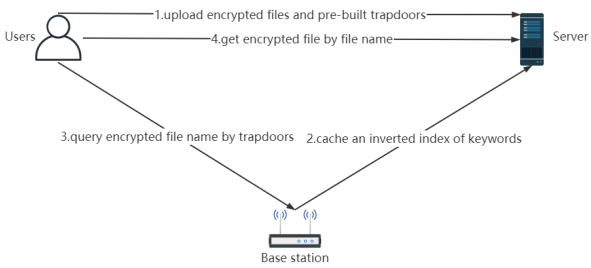


FIGURE 1. Symmetric searchable encryption schema process.

is returned. The server can only know whether the encrypted file has a keyword contained in the trapdoor.

4) Users use the key to decrypt the encrypted file returned by the server to obtain query results.

In the above process, the user can obtain the ciphertext's query authority by sharing the key and other methods. The user's key keep secret from the third party. The server neither knows the user's key nor the file content. So the user's encrypted file content is safe. Figure 1 shows the specific SSE process. However, most of the traditional SSE is based on cloud storage mode. Obviously, the cloud server undertakes a large number of search operations, and the file transmission has a large transmission overhead. In the mobile edge network, the base station is close to the user's site, which has a certain storage capacity and computing power. In this paper, we use the resources of the edge base station to cache encrypted file index to provide search efficiency.

Symmetric searchable encryption usually combines with a variety of encryption techniques. For example, the combination of symmetric searchable encryption and attribute-based encryption technology [18], [24] can make the combined scheme have the function of querying by fine-grained attributes. The combination of symmetric searchable encryption and proxy re-encryption [25] can enable the combined scheme to realize a multi-user search function. It should be pointed out that multi-user and multi-keyword often increase the complexity of the system. How to prevent the cost of encryption algorithm increase with the increase of users, which also has some work [26].

Current work on searchable encryption has focused on the single-server model. All data store in a single server rather than multiple servers. Users tend to choose multiple cloud service providers at the same time to ensure data security, while cloud service providers tend to provide services to more users to make money. Therefore, the multi-server multi-user model is one of the future development trends of searchable encryption [17]. The multi-server multi-user model's difficulties are as follows: keyword retrieval involves multiple untrusted objects; [27] exists key leakage problem under multi-user sharing; the multi-user single server model has the query result ordering problem the safety of PEKS.

With the rapid increase in the number of users with searchable encryption, the traditional server-centric service model will not cope with such a large number of requests. Searchable encryption in the encryption, update, retrieval phase will have more energy consumption. Computing migration and content caching in mobile edge networks must diffuse the pressure on the central servers. At present, the mainstream searchable encryption scheme bases on symmetric searchable encryption (SSE). By constructing the keyword of custom ciphertext into an inverted index [19], the user can find the trapdoor and the corresponding ciphertext in the inverted index according to the ciphertext of the user's keywords when querying. An index can be split into multiple units or merged into a single global index [28]. Therefore, with SSE, indexes can be naturally divided and cached in the edge network.

As we know, the user's request conforms to Zipf Law [29]. Users of the same network will often access the same index item. The same user will repeatedly request the same index item. Therefore, there are numerous application scenarios [24] in the cache edge network. Suppose the base station in the edge network near the user happens to have the index item needed by the user. In that case, the user does not need to get it from the remote central server, which significantly saves the traffic [30] in the edge network. For the central server, reducing the number of direct requests from users, the server's pressure and cost can be significantly reduced. The service provider saves the traffic cost paid to the network operator by reducing the backbone network traffic. Chen *et al.* [31] discussed in detail a partial migration decision problem in a multi-mobile device scenario. The author abstracts it into a solvable linear programming problem. The best solution obtained by exhaustive can reduce the amount of computation by 40% compared with no strategy. Searchable encrypted index entries are like static data cached by Content Delivery Network (CDN) in [32], and there can be many caching algorithms that ultimately reduce carrier traffic.

One approach is to use online technology that selects a specific cache item [24], [33] based on recent requests. It has the advantage of being able to change itself in more real-time to respond to user requests. The impact of caching on request delays can be felt more intuitively by users. Online technology is necessary for systems with higher real-time requirements. One problem with this technology is that base stations take up a lot of computing power and cannot integrate with other base stations. An alternative approach is to use the offline technique to preload possible index entries [34] on a base station close to the user. The base station can choose to cache index entries when the network is idle. The optimal cache algorithm is an NPC problem. If the optimal solution is required, it will consume too much computing resources. In order to improve the effectiveness of caching, there is much work on caching algorithms in edge networks. The number of indexes that the base station can cache is much smaller than the number of global indexes that the central server has. The number of indexes that the base station can cache and its strategy to cache data will significantly affect the edge network's performance [35]. Some of the work has focused on creating a collaborative cache [33], [36] which collaborates between

edge devices to form a connection network [34], [37] with minimal latency and energy consumption.

In this paper, we propose a cache placement algorithm, which combined a mobile edge network model and an edge network with game theory. This algorithm solves the searchable encrypted's index caching problem in the edge network. In our algorithm, we will build a mobile edge network model based on game theory algorithms [35]. Each node of the network is a base station, which can store index entries. Each node is responsible for maximizing its interests. Our primary researches are as follows:

1) We define an edge caching network model. The network contains many base station nodes. Base stations can provide services for mobile users. Every base station in the edge network can keep a limited number of indexes. Base stations can get indexes from other base stations that they do not have. If the requested index does not exist in any base station, the base station will obtain it from the central server.
2) We define a game theory model to find spontaneous cooperation phenomena in non-cooperative games. In our game, every base station is a player. Each player can decide what index to store and modify its strategy based on the data currently held by other players. If a player asks other players to provide data, they need to pay. In the game, we find the Nash equilibrium by maximizing the utility of each player. After further analysis, we find a pure Nash equilibrium. At the Nash equilibrium point, we can obtain a cache placement scheme for the locally optimal solution. By analyzing each player's cache strategy, we can find that there is a spontaneous collaboration between the caches. Finally, we design a distributed algorithm that can achieve balance.

Our solution has better security than CDN. Our scheme adopts the searchable encryption mechanism, which can search for content on the premise that the server is not aware of the encrypted content. Our scheme has better performance than the traditional encryption scheme because it distributes the data to the edge network for processing and reduces the pressure on the server.

The rest of this work is structured as follows. Section II describes the content migration strategies for edge networks and searchable encryption. Section III describes the problem, defines the system model and the game model and studies. Section IV provide the Nash equilibrium exists in the game. In section V, we found the pure Nash equilibrium and proposed a caching algorithm. The simulation results and conclusions are presented in Section VI and Section VII, respecively.

II. RELATED WORKS

A. SEARCHABLE ENCRYPTION

Searchable encryption so far there had been a lot of work research direction included: symmetrical searchable encryption and asymmetric searchable encryption; unconventional searchable encryption scheme; supported for multi-user multi-server; fine-grained control of retrieval content; the key to weight and add or delete; fuzzy query; the more keyword query; query results sorting [17]. Many lightweight solutions [4], [15], [21], [38] had been derived from symmetric searchable encryption (SSE), which had low key generation overhead. Lightweight solutions typically used pseudo-random number encryption [21]. Alternatively, delegating the encryption operation to the edge device for quick computation [39]. Generally speaking, lightweight could have lower energy consumption. An essential requirement in an encrypted search was to allow multi-user retrieval.

Liu *et al.* [9] and Cui *et al.* [40] proposed an encrypted search scheme that allowed multi-user to search in cloud storage. Users could generate private keys for encryption, and other users could query the encrypted content via the public key. As the number of users increases, it tended to increase the load on the server. It was very unfriendly to a service provider of searchable encryption. Li *et al.* [22] proposed a potentially secure multi-user multi-keyword searchable encryption scheme. This approach did not increase the data owner's encryption workload as the number of data users increases. To improve the accuracy of multi-keyword fuzzy search, Zhong *et al.* [41] develop an index tree and top-k search algorithm.

To reduce the load on the cloud server, Mollah *et al.* [42] proposed a secure data search and sharing scheme. The scheme implemented the weak trust hypothesis on edge devices. It also supported delegating computation-intensive encryption and decryption to edge devices, which reduced the overhead of keyword search latches. Zhong *et al.* [43] proposed a two-stage index-based central-keyword ranked search scheme. The scheme can reduce the computation cost in the query process. For multiple query keywords, the scheme's search results are more accurate. Wang *et al.* [16] used fog computing to distribute encrypted essential data to multiple fog nodes of an edge network. To support the distribution of encrypted data, Curtmola *et al.* [23] studied the trusted data outsourcing mechanism in encrypted search. This mechanism could send encrypted data security scores to untrusted objects. Cui *et al.* [44] use online/offline ABE technology and outsourcing technology to reduce the online calculation cost and the local calculation cost of mobile users. Some authors tried to apply blockchain to encrypted search [10], which improved the efficiency of search.

B. CONTENT MIGRATION STRATEGIES FOR EDGE NETWORKS

The traditional content migration strategy was similar to CDN. In an edge network, static resources could cache locally. It could be retrieved directly from the local when a single user requests, significantly reducing the response time. The related work mainly focuses on reducing the traffic of operators. The distributed algorithm [32] was adopted for CDN with the method of cumulative value compensation

and objective value compensation. Gharaibeh *et al.* [33] also worked on the impact of caching capacity on the caching effect. Caching capacity could improve the hit rate to a certain extent. If there were a cooperative feature in the migration strategy, the hit rate would be much higher than a non-cooperative strategy. The author of [33] studied the problem of collaborative caching in multi-cluster collaborative systems and proved its NP-completeness. This paper presented an online cache multi-unit collaboration system, which could obtain the best competition ratio without content popularity. Correspondingly, Garetto *et al.* [30] focused on the impact of content popularity on caching. The change in the cache performance of traditional LRU, random Q-LRU, and 2-LRU caching strategies for dynamically changing content popularity was tested. Since the mobile edge computing server (MECS) consumed a lot of computing power, The author of [45] proposed an energy-saving edge device uncertainty strategy. This strategy, which took into account size, mobility , and cost, could significantly reduce energy demand. The authors of [18] gave a comprehensive analysis of the security threats, challenges, and mechanisms that existed in edge networks and cloud computing, with an in-depth analysis of the potential synergies. The author of [34] presented a scheme for moving edge computing that minimized the energy allocated by cache resources. A connection network with the minimum delay and energy consumption construct through the allocation of cache resources. In general, researchers focused on the hit rate and the ultimate benefit of caching. Collaboration between nodes could have a significant impact on the final cache performance. Thus, many cooperation-based caching strategies had sprung up [24], [33], [36], [39], [46]. There is a game-theory-based content migration strategy in the offline strategies [35], [37], [47]. This strategy could reduce the energy consumption ratio by 40% without a content migration strategy. In our scheme, we drew on this game theory. In the simulation, we adopted the non-cooperative game theory scheme. We were surprised to find the potential synergies and collaboration scenarios mentioned in [18]. Moreover, our method could increase the edge network nodes into an increase of cache capacity, further improving the hit ratio because of this synergistic effect.

Overall, much work had focused on reducing the overhead of encrypted data [4], [13], [15], [21], [23], [39]. Some of the work focused on the edge of the contents of the network optimization of migration policies [17], [24], [31]–[35], [37], [45]. Some work also focused on data security [5], [6], [12], [15], [17], [23].

The above method considers the benefits of a single node. In our proposed scheme considers the network load increased by requests within the network. Combined with the cost in the network, we consider each node's comprehensive income as the game's goal and analyze our model offline.

III. MODEL AND PROBLEM FORMULATION

Much work has been done on searchable encryption and content migration decisions of mobile edge networks. The original SSE encryption scheme is to store the index on a cloud server. It does not take into account the efficiency of the network. We introduced edge caching on top of the original cloud storage-based encryption scheme, which is rare in past work. From the perspective of optimizing edge caching, we introduced a non-cooperative game algorithm to optimize the cloud storage index to increase the probability of users searching from the edge to the index, thereby reducing the user's search delay. This dramatically improves the efficiency and experience of user queries.

For example, suppose the time it takes for the user to look up the index from the cloud server is t_1, the time for the user to look up the index from the edge cache is t_2, the round-trip delay from the user to the cloud service is RTT_1, and the round-trip delay from the user request to the base station is RTT_2. Because users only need to query the file list when searching for files. They do not necessarily need to download encrypted files, so the time to download the files is not considered. Before using the edge cache, the user requested time $tr_1 = t_1 + RTT_1$. In the best case, the user can find the target index in the edge node and the total time requested by the user $tr_2 = t_2 + RTT_2, tr_2 \ll tr_1$. In the worst case, the user cannot find the target index at the edge node, $tr_2 = t_1 + t_2 + RTT_1 + RTT_2, tr_2 > tr_1$. Because $t_2 + RTT_2 \ll t_1 + RTT_1, tr_2 = tr_1$. Even with only a 50% hit rate, the average time for users to find will also be greatly reduced.

This section considers a searchable encryption scheme where the edge network cache the index of a keyword trapdoor. In this case, the user only needs to query the index from the edge network's base station. Compared with cloud computing's search method, edge computing can complete computing tasks closer to users. So our scheme can reduce the pressure of core network traffic and improve users' service experience.

A. SYSTEM MODEL

The development of encrypted search technology has been widely used in cloud storage (such as network disk, personal file storage, trade secret storage). In such a typical application scenario, reducing the delay of user acquisition and reducing core traffic has become the focus of academic attention. A vital optimization direction is to store the indexed content of the search in the margin. Different from CDN, our scheme stores the index of the encrypted content. Only users with the key can decrypt the content. A simple example of the scenario is shown in Figure 2.

In our scheme, the central server stores a large amount of ciphertext index information. Users can get whatever content they want from the central server. The base stations are on the edge of the network. Their computing and storage capacity is generally limited. The base station can cache the ciphertext trapdoor index and return the index information when a user requests a particular file. There are multiple base stations in the edge network. When a user requests something, the user first sends the index to the nearest edge base station. Then

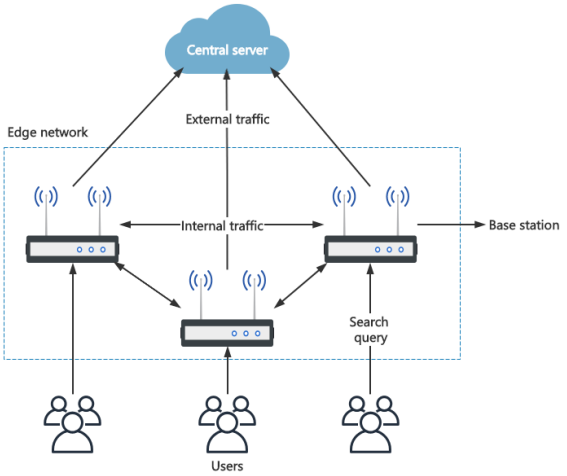


FIGURE 2. Example of the index cache network model.

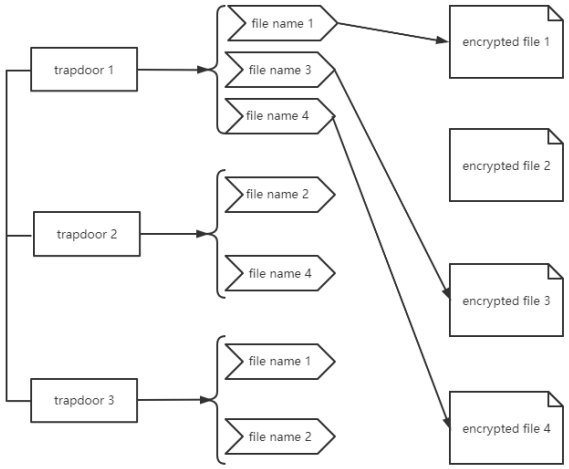


FIGURE 3. Invert index structure.

the result of the ciphertext query is obtained by querying the index obtained. The index of the ciphertext will be distributed in the base station near the user (The base station collects the trapdoor of the keyword in the user's request when querying the encrypted file and the index of the file in the server's response). The base station queries the user's index, and then the base station looks for the same index in its cache. If the base station directly connected to the user has the index of encrypted file required by the user, the base station will deliver the index of encrypted file directly to the user. The base station can construct its ciphertext index of keywords in the nearest edge network through the information recorded by the base station. Suppose the index does not exist in the nearest base station but exists in a neighboring base station. In that case, the base station will send a request to the neighboring base station and return the query result to the user separately.

In our game theory scheme, base stations collaborate to reduce the cache redundancy and improve base stations' storage resources' utilization rate. A single base station will also choose to replace the index of infrequently used keywords based on the request's content, aiming to increase the proportion of valid data in memory. We can reduce the pressure on the information center query through our scheme, improve the user's query speed, and reduce the base station cache cost. Our proposed scheme defines the base stations' cost as the delay between the request and the query result. Base stations' revenue is related to the number of file requests. When the user's request arrives at the base station, the base station will first search for encrypted files through the inverted index. Suppose the local base station has the keyword ciphertext searched by the user. In that case, the base station will directly return the list of the encrypted files corresponding to the keyword inverted index to the user. If the index does not exist, the base station will ask the base station's neighbor to query this keyword. At this time, if the neighbor base station exists, the base station will get the encrypted file list from the neighbor base station and return the encrypted file to the user. The base station will query the ciphertext corresponding to the keyword to the central server and send ciphertext to the user if it does not exist. The difference between these scenarios is that the user's query time increases over time. We assume that the corresponding encrypted file in the inverted index will not be changed to simplify our model, and we do not consider cache expiration cases.

Therefore, allocating an inverted index in the base station is the key to the problem we studied. It affects the user's query experience (query wait time) and is closely related to the base station's cache cost and utilization rate.

B. SYSTEM FLOW

This section provides a detailed system flow description of searchable encryption in the edge network cache system. The details are shown in Figure 2. We considered a classic SSE scheme [23]. In this SSE scheme, the user uses the private key to encrypt the file. Then the user encrypts the keyword by the private key to obtain trapdoors. Finally, the trapdoor and encrypted files are uploaded to the server and organized into an inverted index. Specifically, the structure of the inverted index is shown in Figure 3:

The construction of SSE's inverted index [23] is shown in Algorithm 1. Because the encrypted files are all encrypted and decrypted by the user, each encrypted file's key is unique. Each node's previous node stores the key of the node's file id in the array linked of keywords in the SSE scheme. By encrypting the file id, the server cannot know the specific file corresponding to the keyword searched by the user. This prevents the server from inferring the content of the file through the keyword.

In the traditional cloud-based search model, as the number of ciphertext increases, the index will snowball. The query time will also be longer. If a large number of users access the ciphertext simultaneously, the server will be blocked or even crash. A practical solution is to store part of the index in an edge base station, just like a CDN. The user's request conforms to Zipf law, and users in the same base station are likely to query the same several keywords. So for encrypted

Algorithm 1: SSE Index Construction Algorithm

Gen(1^k): sample $K_1, K_2, K_3 \xleftarrow{\$} \{0,1\}^k$, generate $K_4 \leftarrow$ **SKE2.Gen(1^k)** and output $K = (K_1, K_2, K_3, K_4)$

$Enc_K(D)$:

Initialization:

scan D and generate the set of distinct keywords $\delta(D)$

for all $w \in \delta(D)$,generate $D(w)$

initialize a global counter $\mathbf{ctr} = 1$

Building the array A:

//build a list L_i with nodes $N_{i,j}$ and store it in array A

for $1 \le i \le |\delta(D)|$ **do**

  sample a key $K_{i,0} \xleftarrow{\$} \{0,1\}$

  for $1 \le j \le |D(w_i)| - 1$ **do**

    let $\mathbf{id}(D_{i,j})$ be the j^{th} identifier in $D(w_i)$

    generate a key $K_{i,j} \leftarrow \mathbf{SKE1Gen}(1^k)$

    create a node $N_{i,j} = \mathbf{id}(D_{i,j})||K_{i,j}||\Psi_{K_i}(\mathbf{ctr}+1)$

    encrypt node $N_{i,j}$ under key $K_{i,j-1}$ and store it in A:

    $A[\Psi_{K_i}(\mathbf{ctr})] \leftarrow \mathbf{SKE1.Enc}_{K_{i,j-1}}(N_{i,j})$

    set $\mathbf{ctr} = \mathbf{ctr} + 1$

  end

  for the last node of L_i:

  set the address of the next node to **NULL**:

  $N_{i|D(w_i)|} = \mathbf{id}(D_{i,|D(w_i)|})||0^k||\mathbf{NULL}$

  encrypt the node $N_{i|D(w_i)}$ under key $K_{i,D(w_i)|-1}$ and store it in A:

  $A[\Psi_{K_i}(\mathbf{ctr})] \leftarrow \mathbf{SKE.Enc}_{K_{i,|D(w_i)|-1}}(N_{i,|D(w_i)|})$

  set $\mathbf{ctr} = \mathbf{ctr} + 1$

end

let $s' = \sum_{w_i \in \delta(D)} |D(w_i)|$

if $s' < s$ **then**

  set the remaining $s - s'$ entries of A to random values of the same size the existing s' entries of A

end

searches, it is reasonable to add an index to the base station. The distance between base stations and the transmission cost is much less than direct access to the central server. Therefore, cooperation between base stations with limited capacity is crucial. A larger edge buffer pool can establish through cooperation between base stations. Based on the above analysis, we propose the following solutions:

1) In the first step, the user uses the terminal device to query a keyword; the terminal device will connect to the nearest base station. The base station is expected to help the user get the ciphertext list corresponding to the keyword searched by the user from the server. Because the base station query close to the user has a shorter delay, and the probability of the same content query by different users covered by the same base station is high. So the priority is to try to query from the nearest base station. We do not need to know the user's specific geographic location. By comparing the signal-to-noise ratio of different base stations and choosing the base station with the highest signal-to-noise ratio, we can get the base station closest to the user on geographical.

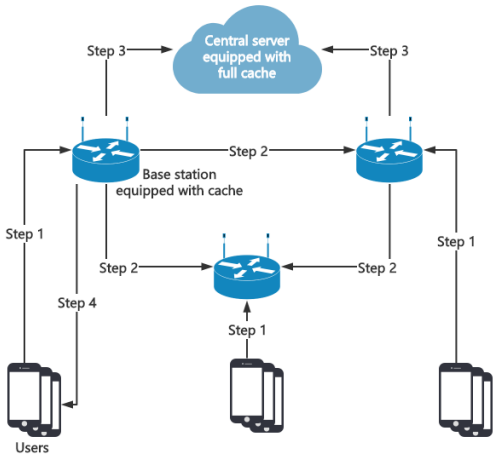


FIGURE 4. System flow chart.

2) In the second step, if the base station cannot find the content required by the keywords in the local index cache request, it will first forward the request to other adjacent nodes. Each forwarding is called a hop. The distance between the base station is greater than the distance between the user and the nearest base station. However, still much greater than the distance between the base station and the central server. (the time spent on the request positively correlates with the distance) Merely looking up between base stations takes less time than directly querying the central server. The base station can be guaranteed to be profitable. The adjacent base station is not necessarily directly connected, may also be across from other base stations. As long as it is the final appears to be beneficial, that is an adjacent base station.
3) In the third step, if nearby nodes cannot find the target content, the base station will forward the request to the server equipped with all the data. Because the center server needs to respond to a base station's request, the number of ciphertext indexes is greater than the number of base station's indexes. The search process will be prolonged. To simplify this, we combine the lookup time with the time the request travels over the network.
4) In the fourth step, the base station has obtained the contents of the ciphertext searched this time. The base station will make a judgment on the cache locally to determine whether to store the cache. For each base station, the base station will give value to each data. The data distribution of the entire edge network will determine The value of this data. If the base station caches some data, it will reduce the base station's total cost, and then the data will be valuable. Each base station caches the data that is most valuable to it.

C. PROBLEM FORMULATION

According to our description above, we consider the topology of an edge network $G = (D, e)$, which contains a set of finite cache base stations $D = \{1, 2 \ldots, D\}$. $e_{i,j}$ represents whether base station i is connected to base station j. $h(i, j)$ represents the shortest number of hops between base station i and base station j, so $h(i, j) = 0$, $h(i, D + 1) = K$ represents the time needed for base station to directly obtain index information from the central server. We assume that the distance between the central server and the base station is much greater than the distance between the base stations, so $h(i, D+1) = K \gg h(i, j), \forall j \in D$. We have $K = \{1, 2, \ldots, j, \ldots, K\}$ users in the edge network, which will make searchable encrypted query request. Let w_i^r be the number of times the user queries r keywords to a base station, and t^r represents the sum of all times the base station queries r keywords. Let a_i^r represent whether base station i caches the index r. We use ω to represent the revenue that the base station satisfies the users. ϵ denotes each hop's cost to an adjacent base station, and λ is the cost of a base station to query the index directly from a central server. Obviously, $\omega < \epsilon \ll \lambda$. To simplify the problem, we assume that all base stations have a cache size of c.

D. GAME MODLE

In our basic game model, the player is the base station on the edge network. There are D players in the game. Each player can hold c indexes in the set K. Different players can hold the same index repeatedly. The goal of the game is to find the best index placement strategy.

In the game model, what each base station needs to do during the game is adjust the cache configuration ($P_i = \{a_i | \sum_i a_i^r \leq p\} \subset \{0, 1\}^{|K|}$) holds. Most importantly, each base station's cache is only for its benefit maximization, not for the benefit maximization of all base stations. The reason $r_i \leq p$ exists in the formula is that each base station's cache is not necessarily filled. a_i^r is used to describe whether the index r is cached in base station i. Use P_{-i} to represent the cache policy configuration that base station i expects. $R_i(P_{-i}) \xrightarrow{def} \{n | a^r = 1, \forall n \in D\}$. According to our base station model, the utility of a base station in the edge network can be obtained in the following:

if $a_i^r = 0$

$$C_i^r(\{a_i^r\}, P_{-i}) = \begin{cases} t_i^r(\lambda - \omega), & a_i^r \notin R_i(P_{-i}) \\ t_i^r(\epsilon l(i, j) - \omega), & a_i^r \in R_i(P_{-i}) \end{cases} \quad (1)$$

if $a_i^r = 1$

$$C_i^r(\{a_i^r\}, P_{-i}) = \begin{cases} t_i^r \omega + \sum_{k \in D \setminus \{i\}} t_k^r \epsilon h(i, k), \\ a_i^r \notin R_i(P_{-i}) \\ t_i^r \omega + \sum_q t_q^r \epsilon h(i, q)/F \\ + \sum_s t_s^r \epsilon h(i, s), a_i^r \in R_i(P_{-i}) \end{cases} \quad (2)$$

We use $C_i^r(P_i, P_{-i})$ to denote the utility of the base station. $q = \{h(i, q) - h(R_i(P_{-i}, q) = 0 | q \in D \setminus (\{i\} \cup R_i(P_{-i}))\}$, $q = \{h(i, p) - h(R_i(P_{-i}, p) < 0 | p \in D \setminus (\{i\} \cup R_i(P_{-i}))\}$, $F = |q|$ represents the number of the nearest player that cache index r.

The expression (1) indicates that when base station i does not have an index r. Other base stations, except base station i, may have two situations: other base stations cache index r. The other is that other base stations do not cache index r. In the first case, since no base station has cached index r, the user requesting index r can only get resources from the central server. The user pays the base station some virtual money, and the base station pays the central server. In the second case, other base stations cache the index r, assuming that base station j stores the index r and is closest to base station i. The base station i can obtain the index r from base station j. The user pays base station i, which pays base station j. (the cost depends on the minimum number of hops from base station i to base station j) The user requests an index from base station i because the distance from the user to base station j is greater than the distance from base station i to base station j plus the distance from the user to base station i.

The expression (2) indicates that when base station i caches index r, there are also two cases. The first case is that no base station except base station i has index r, and base station i does not have any cost. Base station i benefits from fees paid by nearby users and other base stations that pay base station i when requested to index r. In the second case, at least one base station besides base station i also has index r. In this case, base station i does not have any cost. The revenue comes from users' payment close to base station i and the payment of nodes closer to base station i than other nodes with index r. Suppose another base station j and base station i are the same number of hops away from the requested base station and have an index r. In that case, base station i and base station j will split the benefits equally.

We analyze the utility of each base station node. The utility of base station i is divided into four parts, which are represented by u_1, u_2, u_3 and u_4, respectively:

Video r is cached in both base station i and other base stations. If the user requests index r, then u_1 in the utility of the base station can be described as:

$$u_1 = \sum_{a_i^r \in R_i(P_{-i})} ((t_i^r \omega) + \sum_q t_q^r \epsilon h(i, q)/F + \sum_s t_s^r \epsilon h(i, s)) \quad (3)$$

The above expression indicates that base station i and at least one other base station cache index r. If the edge network's base station has an index r, the user closest to the base station i will directly obtain the index r from base station i. If the base station j of the edge network without index r. Base station i is the nearest base station with index r. Base Station j will request index r from base station i and pay the fee. If multiple base stations have an index r, and the distance from the requested base station without index r is equal, multiple base stations will provide the index r, and the revenue will be shared. Finally, if base station i is the only one in the edge network with index r cached, there is no need to

share the benefits with other base stations. In the same way, we can analyze u_2, u_3, and u_4.

When index r is cached in any base station except base station i, sub-utility u_2 can only be obtained from the base station directly connected to base station i. Therefore, u_2 is described as follows:

$$u_2 = \sum_{a_i^r \in R_i(P_{-i})} t_i^r(\epsilon h(i,j) - \omega) \tag{4}$$

In this case, because base station i does not cache index r, it cannot provide the desired index r directly to the user from the local. Therefore, the revenue obtained by base station i should deduct the cost paid by base station i to other base stations, and the cost increases with the increase of $h(i, j)$.

When index r is cached only in base station i and no other base station exists, the sub-utility u_3 of base station i is expressed as:

$$u_3 = \sum_{a_i^r \notin R_i(P_{-i})} a_i^r(t_i^r \omega) + \sum_{k \in D \setminus \{i\}} t_k^r \epsilon h(i,k)) \tag{5}$$

Obviously, since only base station i caches index r, all requests for index r from users on the edge network are passed to base station i. Therefore, sub-utility u_3 is the sum of the revenue requested by the local user of base station i and the fees paid by other base stations to base station i.

In the last case, no base station has index r. When the user requests index r, the base station cannot be found in the edge network. We use sub-utility u_4 to describe:

$$u_4 = \sum_{a_i^r \notin R_i(P_{-i})} t_i^r(\lambda - \omega) \tag{6}$$

There is no base station cache index r in the edge network, and no base station can provide services for users. Hence, base stations need to obtain resources from the central server, which is the highest cost.

Finally, we can get the total utility of base station i through the above four sub-utilities, which can be described as follows:

$$U_i(P_i, P_{-i}) = \sum_r C_i^r(a_i^r, P_{-i}) = u_1 + u_2 + u_3 + u_4 \tag{7}$$

The equation shows the utility of a base station i so that we can define a game strategy. The game goal of the base station i is to maximize the utility of the base station i. The base station adjusts the cache placement strategy through the current mitigation situation of the edge network. Table 1 summarizes the key symbols used in this paper.

IV. NASH EQUILIBRIUM IN THE COOPERATIVE CACHE

No matter how the initial index distributes in base stations. It is not apparent whether the base stations on the edge network can achieve a stable cache allocation strategy through the summarized revenue maximization expression. We will then discuss whether base stations on the edge network can produce stable index allocations when given initial cache allocations.

TABLE 1. The key symbols used in this paper.

Symbol	Meaning
D	Base station with limited cache
K	All indexes that can be cached
c	The capacity of indexes that each base station can cache
t_i^r	The number of times base station i queried index r
$G = (D, e)$	The topology of a base station set D and an edge set e
$e_{i,j}$	Whether the base station i and base station j are adjacent
$h(i, j)$	Minimum number of hops between base station i and base station j
a_i^r	Whether the index r is cached by base station i
ω	The revenue the base station receives in response to local requests
ϵ	Cost per hop for base station to query indexes from other base stations
λ	Cost when base station queries the index directly from the central server
P_i	Caching policy for base station i
$R_i(\mathrm{P}\_i)$	P collection of all base stations with the index r cached except base station i
F	The number of base stations adjacent to base station i that do not cache the index r
$U_i(\cdot)$	Utility of base station i

When we study game theory, a key point is whether there is a Nash equilibrium in the defined game. In our game, Nash equilibrium shows that no base station changes its strategy when the index cached in other base stations is not changed and under the current strategy mode. According to the Nash equilibrium definition in the game, we can finally prove whether a Nash equilibrium exists.

Definition 1: Player i's best response to the profile p_i is mixed strategy $p_{-i}^* \in P_{-i}$ such that $m_i(p_i, p_{-i}) \leq m_i(p_i^*, p_{-i})$ for all strategies $s_i \in S_i$.

In our game model, if more than one player receives a request at the same time. Those players can try to adjust the index cache configuration strategy until maximized their respective revenue. Because we have more than two players and indexes, we can have multiple local-revenue maximizers, which means there are multiple best responses.

Definition 2: If a strategy profile $P = (p_1, p_2 \ldots p_n)$ is a Nash equilibrium, for all players i, p_i is a best response to p_{-i}.

The Nash equilibrium strategy is not necessarily the optimal solution among many strategies, but it is a stable one among many strategies. In a Nash equilibrium, everyone makes their own best strategy based on the other players, so as long as all the players know the other players' strategy, they won't change their strategy.

Theorem 1 (Nash 1951): Every game with a finite number of players and action profiles has at least one nash equilibrium.

For our model, the number of indexes that base stations can cache is limited, as are the number of base stations (players) on the edge network. According to theorem 1, we can know that there must be a non-cooperative Nash equilibrium scheme in the model.

V. PURE STRATEGY NASH EQUILIBRIUM

According to the derivation in the previous section and the conclusion of Theorem 1, we can prove that there must be a

Nash equilibrium strategy for a base station i. However, this does not mean that we can get a pure strategy Nash equilibrium from it, nor guide the actual process. Pure strategy Nash equilibrium means that in a pure policy combination, the node itself will not change if other base stations' given strategy does not change. Next, we will prove how to obtain a pure strategy Nash equilibrium.

In the edge network, each base station can communicate with other base stations to know each other's strategy, and finally form a unique strategy. Suppose that at some point before Nash equilibrium is reached, base station i has a strategy $e_1, e_2 \ldots, e_i, \ldots e_D$, to optimize the utility, assuming that there is a policy for $u_i(e_1, e_2, \ldots, a_i, \ldots e_D) > u_i(e_1, e_2, \ldots, e_i, \ldots e_D)$, This new strategy can be replaced with base station i as long as the utility is increased. At this time, from the perspective of the entire edge network as a whole, base station i achieves a pure strategy Nash equilibrium.

Let's take base station i as an example and look at the whole process. At first base station i is full of empty $\{\varnothing_1, \varnothing_2 \ldots, \varnothing_N\}$, we select any slot in it and put an index e_1, which makes $u_i(e_1, \varnothing_2 \ldots, \varnothing_D) > u_i(\varnothing_1, \varnothing_2, \ldots, \varnothing_N)$, so we replace it with base station i. By repeating this method until no new replacement strategy can increase the utility, we make base station i achieve a pure strategy Nash equilibrium.

Algorithm 2: Cache Placement Algorithm

Input: $D, p, \omega, \epsilon, \lambda, t_i^j, h(i, j)$
Output: $s_D\{a_1, a_2, \ldots, a_i, \ldots, a_D\}$

```
s_D{∅_1, ∅_2, ..., ∅_i, ..., ∅_D}
u_i(P_i, P_-i)=0
i = 1
flag=True
while flag do
    for i<D do
        e_i = u_i(a_i^r = 1, P_-i) − u_i(a_i^r = 0, P_-i)
        earnings add (i, e_i)
        sort earning by value with desc and truncated to
          the p th
        if earnings == s_D then
            flag = Flase
        end
        update s_D by earnings
    end
end
return s_D
```

VI. COMPUTATINAL EXPERIMENTS

In this section, we will perform a computational simulation of the game theory model. The simulation aims to determine the influence of the number of nodes and key parameters on the model's final performance. Finally, by comparing with the greedy cache algorithm and global optimal solution, we find out the game theory model's suitable application scenario. In the greedy algorithm, each node only considers how to maximize a single node's revenue and does not consider the benefits that other nodes bring to itself. It will not obtain a Nash Equilibrium through multiple games like game theory algorithms. How to achieve the global optimal solution algorithm is an NP problem. The algorithm obtains the cache layout with the most significant benefit to the entire network through a great number of calculations. This algorithm brings a sizeable computational load to the server. Our key parameters include cost per hop between base stations, cache capacity, data popularity, number of nodes, and connectivity between nodes. The model's performance is measured by the nodes' cost in the hit ratio(including the local hit, one hop hit, two hops hit, more than two hits), the nodes' revenue, and the trunk network traffic.

In the simulation, we base on Python, build a small edge network model, and implement the game theory algorithm. The game theory algorithm is shown in Algorithm 2:

We set the default parameters $\omega = 10$, $\epsilon = 10$, $\lambda = 1000$, $D = 20$. These parameters change with the simulation requirements in the specific simulation. Our request in simulation is generated by Zipf distribution. The parameter A used in generation represents the uniformity of distributed data. The larger parameter A is, the more concentrated the data distribution is, the closer it is to 1, and the more uniform the data distribution is.

In the specific operation, we simulate the user sending a request to the base station. The user will provide λ benefit to the base station. If the base station can respond to the user's request, the base station does not need to pay any cost. If the base station itself cannot respond to the request, but other nodes can respond to the request, then a cost will be paid to the base station in the middle at each hop, and the cost of ϵ will intercept each hop. If none of the edge stations can respond to the user's request, the base station will fetch the central server's content. However, this process will cost the base station all of its benefits. In the simulation, we will test the same data set for each base station and calculate its average value to avoid single-node position influence. The number of user requests for different indexes will be divided by the number of all user requests to eliminate the sample size's effect.

The parameters we set in the simulation are only for studying our game theory model's performance under different conditions. It does not represent the specific parameters in the actual application scenario, so we do not need to pay special attention to the units.

A. THE INFLUENCE OF BASE STATION NUMBER ON SYSTEM PERFORMANCE

Each base station's cache index capacity is much smaller than the total number of indexes stored in the central server. So a single node cannot achieve a 100% hit rate. Under the caching scheme based on game theory, each base station maximizes its interests but finally forms partial cooperation characteristics. As the number of base stations increases, the remaining cache space will be split with other nodes, except for the extremely high-frequency index. As the number of nodes in

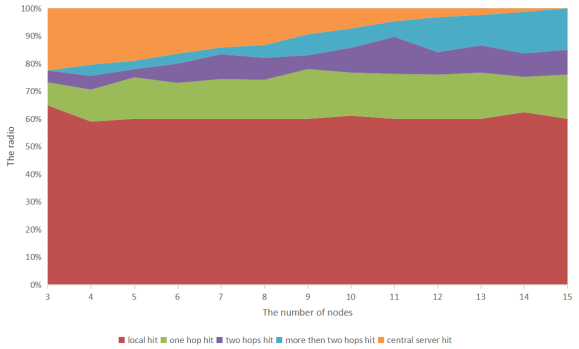


FIGURE 5. The hit ratio varies with the number of nodes.

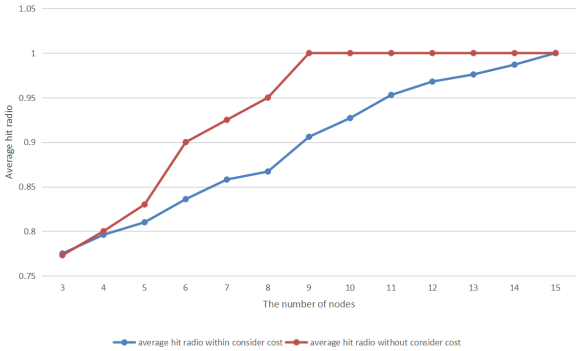


FIGURE 6. Consider the impact of cost on the hit ratio as nodes increase.

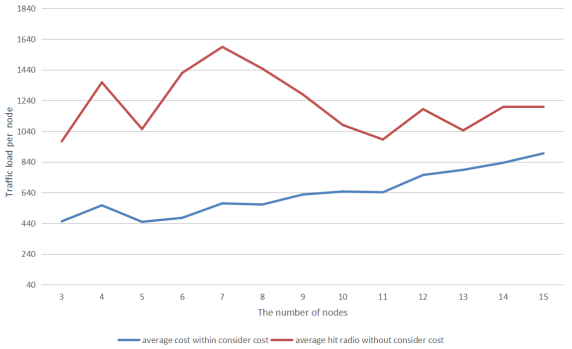


FIGURE 7. Consider the impact of costs on the traffic load as nodes increase.

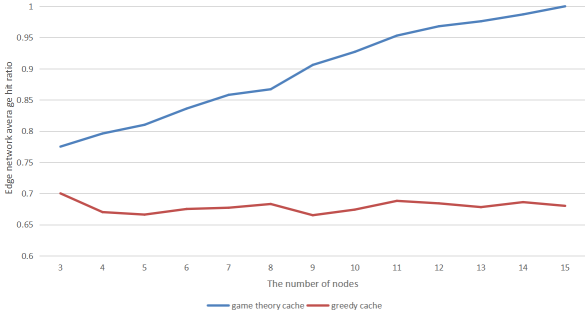


FIGURE 8. The hit ratio varies with the number of nodes.

the network increases, each node's cache size stays the same, the overall cache size increases, and eventually, the edge network caches all the indexes, which makes the edge hit rate up to 100%.

In Figure 5, you can see the distribution of hit rates. The hit rate for local requests stabilizes at 60%, indicating that each node stores the same high-frequency data. The one-hop hit rate is also stable, with very close nodes working in pairs to keep it steady, just like the local hit rate. It can be seen that with the expansion of the number of nodes in the edge network, the hit rate of the central server is gradually squeezed. As the edge network's scale grows to expand, the probability of rare indexes cached in other base stations will be larger. Because the network size is small enough, base stations can get the indexes requested by users from further nodes. The direct result of the expansion is that the base station can leave enough space for the cache to cooperate with the surrounding nodes to improve further the cache hit ratio.

The figure 6 compares the game theory's average edge hit ratio cache model within and without considering cost when the number of nodes changes. When the number of hops is so high that it costs more than getting the index directly from the central server, the base station needs to decide whether to consider the cost's impact. Suppose the cost take into account. The base station fetches from the central server directly when there are too many hops, even if the base station's index exists in the edge network. This node costs more to obtain the index from the base station than to obtain it directly from the server, which is a negative return for the base station. The goal of the base station is to obtain more revenue rather than a higher hit ratio.

Figure 7 compares each node's flow load with the node overview change with and without cost consideration. In the experiment, each node underwent 1000 test cases. The traffic complexity of each node refers to the communication required by the node to request outward once. Each hop will consider traffic if the request to the server directly is also considered traffic. To avoid the influence of changes in the number of nodes on the total traffic, we use the average traffic of each node as the reference object. According to Figure 7, we can see that when considering the cost, the traffic load is much smaller than that without considering the cost. Because without considering the cost, the direct cost exceeds the direct request to the server. The index will also be requested from the base station that exists nearby. It leads to a sharp increase in traffic on the edge network. The traffic load increases with the increase of the number of nodes because the base station has further choices with increased network size. Hence, the increase in hops leads to an increase in traffic load. Combine Figure 7. It can also find that the local hit and one hop hit remain basically unchanged, while the hit ratio of farther base stations increases. It is related to our parameter because we think the traffic load required to request the server directly is 1.

Figure 8 compares the hit ratio between the game theory caching model algorithm and the greedy caching algorithm. A greedy algorithm means that when a node caches an index,

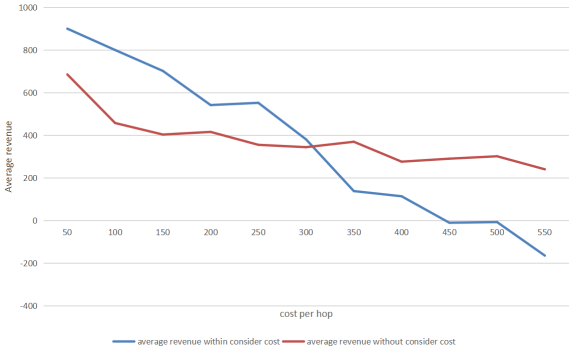


FIGURE 9. The average revenue varies with cost per hop.

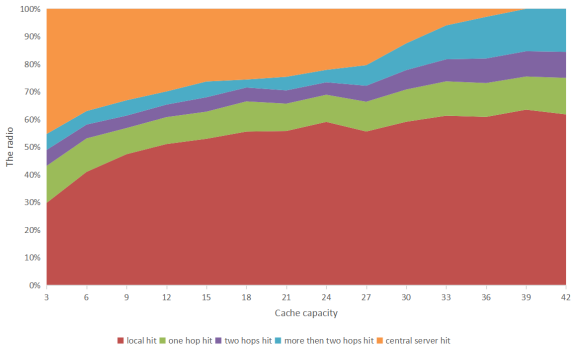


FIGURE 10. The hit ratio varies with the number of nodes.

it does not consider other nodes' interests but only considers its own interests maximization. In the greedy model, the nodes do not cooperate because we assume that each node receives the same user requests. Then the node caches the index with the highest frequency, and if each node receives the same request, the cached content of each node is the same. Since there is no cooperation, even if the number of nodes increases, each node is an isolated individual and does not cooperate to maximize the cache. As the number of nodes increases, the game theory model takes out part of the cache to cooperate with the nearby base stations. The increase in nodes represents an increase in the number of portions of the cache that work together, ultimately increasing the cache's hit rate.

B. IMPACT OF BASE STATION CACHE CAPACITY AND COST PER HOP ON MODEL PERFORMANCE

As can be seen from Figure 9, with the increasing cost of each hop, the revenue obtained by the node will decrease regardless of whether the cost is considered or not. The difference is that without accounting for costs, the node's revenue declines until it becomes a negative income. It is unacceptable to mobile operators. In the case of considering the expense, although the income is relatively low at the beginning, with the increase of each hop of expense, the decline becomes limited and finally reaches the fundamental stability. When considering the cost, if the cost is too much, it will limit the access distance and form a small collaborative network base station topology around it. The higher the cost, the smaller the topology. Although the total capacity is small, the further increase in overhead is limited. It also indicates that our game theory model will perform well in small network structures.

Figure 10 shows how the cache hit ratio varies with each base station's cache capacity. The local cache hit ratio increases significantly at the beginning as the cache size increases and gradually stabilizes as the local cache hit ratio reaches close to 50%. After the initial local hit rate rises to a specific bottleneck, the hit rate of nodes with more than two hops replaces the local hit rate. Finally, making the edge network hit rate reach 100%. There was a slight increase in the percentage of first-hop and second-hop shots, but it was not significant. The local hit ratio is the most important for the

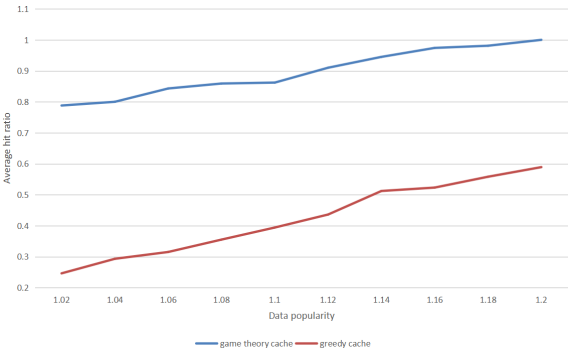


FIGURE 11. Hit ratio varies with the popularity of the data.

base station. The local response request does not need to pay the cost to other base stations so that the higher profit can be obtained. However, when the high-frequency index of local storage reaches a bottleneck, each additional index does not benefit the base station. Cooperate with other base stations to provide low-frequency indexes for other base stations. There are enough other base stations that will bring more revenue. Therefore, after the cache capacity exceeds 30, more than two hops' hit rate increases rapidly. Compared to Figure 5, you can see that the base station's cache capacity primarily determines the local hit ratio. The edge network's total cache capacity has the most significant impact on the cache hit ratio of two hops and above.

Figure 11, we test the impact of data popularity on the model's average hit ratio. The data popularity is the concentration degree of index data, the closer to 1, the more uniform. In greedy caching algorithms, all the data is distributed evenly initially. Even if the greedy algorithm caches the index with the highest frequency, it will not have a high hit rate because the data is too scattered. Base stations in the game theory model can get indexes from nearby base stations, so the average hit rate is higher than greedy cache algorithms.

C. A COMPARISON BETWEEN GLOBAL OPTIMIZATION AND GAME THEORY

We test the average return of both the game theory algorithm and the global optimal solution algorithm in figure 12. As the number of nodes increases, the revenue of both is increasing

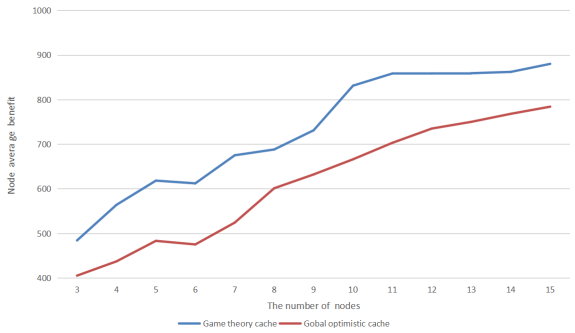


FIGURE 12. Comparison of algorithm returns with the number of nodes.

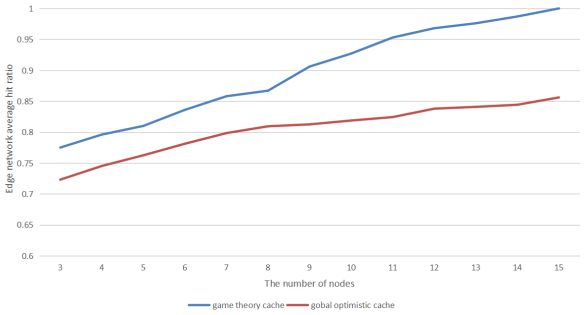


FIGURE 13. Comparison of the hit ratio of the algorithm with the number of nodes.

rapidly. In the game theory model, this is because the network's size increases the probability of storing the needed low-frequency index in further nodes. For the global optimal solution, the cooperation between nodes is considered initially. The average revenue will increase with the increase of nodes. The global optimal solution takes global benefit as the ultimate goal, so it pays less attention to the nodes. Finally, the game theory model's average node yield is greater than that of the global optimal solution. In Figure 13, we test the effect of the number of nodes on the game theory cache algorithm's average hit ratio and the global optimal cache algorithm. We can see that the game theory cache algorithm's average hit ratio increases rapidly as the node size increases. The global optimal solution does not make better use of the added nodes than the game theory algorithm. Although the hit rate also increases with the number of nodes, the global optimal solution rises more slowly than the game theory algorithm. The global optimal solution algorithm focuses on global revenue rather than the hit ratio. It is more sensitive to the increase in count of hops and the increase in cost. Game theory caching algorithms can be considered a viable operation as long as the cost does not exceed the request's benefit, and therefore the hit ratio is higher.

D. COMPARISON OF EXISTING EDGE CACHING SOLUTIONS

The existing edge caching scheme mainly improves the overall system capacity, improves the cache hit rate, and reduces transmission overhead. Table 2 compares the existing edge caching schemes with the game theory-based scheme we proposed. The results show that our scheme has a clear advantage in improving the cache hit rate and reduce network traffic aspects.

TABLE 2. Comparison of edge cache solutions.

Target	Schema	Result
Improve cache hit rate Reduce transmission overhead	Based on game theory	Increase about 25% of cache hit rate Reduce about 80% of network traffic Reduce about 30% of network latency
Improve overall system capacity	Based on user preferences [48]	Increase about 30% of the overall system capacity
Improve cache hit rate	Based on reinforcement learning [49]	N/A
Reduce network traffic	Based on Markov decision process [50]	Reduce about 70% of network traffic
Reduce network latency	Based on integer linear programming [51]	Reduce about 30% of network latency
Reduce network latency	Grouping-based strategy [52]	N/A

VII. CONCLUSION

In this paper, we propose a game-theoretic caching strategy for indexing in mobile edge networks. Through our cache placement algorithm, each edge node could select its caching strategy in advance, effectively alleviating the traffic pressure on the backbone network. Compared with the global optimal solution algorithm, the game theory algorithm's hit rate has increased by about 10%, and the average node revenue has increased by about 20%. Compared with traditional cloud computing, our solution saves about 80% of backbone network traffic and 30% of network latency. In further work, we will study how this paper's caching strategy ensures security under heterogeneous infrastructure [53]. In addition, we will study the sorting optimization problem in distributed caches in the future.

REFERENCES

[1] G. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, Feb. 2019.
[2] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014, doi: 10.1109/MCOM.2014.6871674.
[3] A. Awad, A. Matthews, Y. Qiao, and B. Lee, "Chaotic searchable encryption for mobile cloud storage," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 440–452, Apr. 2018, doi: 10.1109/TCC.2015.2511747.
[4] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3712–3723, Aug. 2018, doi: 10.1109/TII.2017.2784395.
[5] Y. Yao, Z. Zhai, J. Liu, and Z. Li, "Lattice-based key-aggregate (searchable) encryption in cloud storage," *IEEE Access*, vol. 7, pp. 164544–164555, 2019, doi: 10.1109/ACCESS.2019.2952163.
[6] X. Liu, G. Yang, W. Susilo, J. Tonien, R. Chen, and L. Xixiang, "Message-locked searchable encryption: A new versatile tool for secure cloud storage," *IEEE Trans. Services Comput.*, early access, Jul. 2, 2020, doi: 10.1109/TSC.2020.3006532.

[7] K. Li, W. Zhang, C. Yang, and N. Yu, "Security analysis on one-to-many order preserving encryption-based cloud data search," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1918–1926, Sep. 2015, doi: 10.1109/TIFS.2015.2435697.

[8] L. Zhang, J. Su, and Y. Mu, "Outsourcing attributed-based ranked searchable encryption with revocation for cloud storage," *IEEE Access*, vol. 8, pp. 104344–104356, 2020, doi: 10.1109/ACCESS.2020.3000049.

[9] X. Liu, G. Yang, Y. Mu, and R. H. Deng, "Multi-user verifiable searchable symmetric encryption for cloud storage," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1322–1332, Nov. 2020, doi: 10.1109/TDSC.2018.2876831.

[10] J. Niu, X. Li, J. Gao, and Y. Han, "Blockchain-based anti-key-leakage key aggregation searchable encryption for IoT," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1502–1518, Feb. 2020, doi: 10.1109/JIOT.2019.2956322.

[11] H. Wang, J. Ning, X. Huang, G. Wei, G. S. Poh, and X. Liu, "Secure fine-grained encrypted keyword search for e-Healthcare cloud," *IEEE Trans. Dependable Secure Comput.*, early access, May 19, 2019, doi: 10.1109/TDSC.2019.2916569.

[12] Y. Yang, "Towards multi-user private keyword search for cloud computing," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, Washington, DC, USA, Jul. 2011, pp. 758–759, doi: 10.1109/CLOUD.2011.76.

[13] A.-P. Xiong, Q.-X. Gan, X.-X. He, and Q. Zhao, "A searchable encryption of CP-ABE scheme in cloud storage," in *Proc. 10th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. (ICCWAMTIP)*, Chengdu, China, Dec. 2013, pp. 345–349, doi: 10.1109/ICCWAMTIP.2013.6716664.

[14] J.-Y. Huang and I.-E. Liao, "A searchable encryption scheme for outsourcing cloud storage," in *Proc. IEEE Int. Conf. Commun., Netw. Satell. (ComNetSat)*, Bali, Indonesia, Jul. 2012, pp. 142–146, doi: 10.1109/ComNetSat.2012.6380794.

[15] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 772–785, Sep. 2019, doi: 10.1109/TSC.2018.2823309.

[16] S. Wang, Y. Li, H. Wang, X. Zhang, and J. Chen, "Attribute-based weighted keyword search scheme supporting multi-search mechanism in fog computing," in *Proc. 7th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 6th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, New York, NY, USA, Aug. 2020, pp. 23–28, doi: 10.1109/CSCloud-EdgeCom49738.2020.00014.

[17] P. J. Sun, "Privacy protection and data security in cloud computing: A survey, challenges, and solutions," *IEEE Access*, vol. 7, pp. 147420–147452, 2019, doi: 10.1109/ACCESS.2019.2946185.

[18] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: A survey," *IEEE Trans. Services Comput.*, vol. 11, no. 6, pp. 978–996, Nov. 2018, doi: 10.1109/TSC.2017.2762296.

[19] J. Xu, C. Ying, S. Tan, Z. Sun, P. Wang, and Z. Sun, "An attribute-based searchable encryption scheme supporting trapdoor updating," in *Proc. IEEE 16th Int. Conf Dependable, Autonomic Secure Comput., 16th Int. Conf Pervasive Intell. Comput., 4th Int. Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Athens, Greece, Aug. 2018, pp. 7–14, doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00017.

[20] L. Tajan, D. Westhoff, C. A. Reuter, and F. Armknecht, "Private information retrieval and searchable encryption for privacy-preserving multi-client cloud auditing," in *Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Barcelona, Spain, Dec. 2016, pp. 162–169, doi: 10.1109/ICITST.2016.7856690.

[21] X. Yang, G. Chen, M. Wang, and X. Pei, "Lightweight searchable encryption scheme based on certificateless cryptosystem," in *Proc. 4th Int. Conf. Mech., Control Comput. Eng. (ICMCCE)*, Hohhot, China, Oct. 2019, pp. 669–6693, doi: 10.1109/ICMCCE48743.2019.00155.

[22] S. Li, F. Wang, T. Shi, and J. Kuang, "Probably secure multi-user multi-keyword searchable encryption scheme in cloud storage," in *Proc. IEEE 3rd Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Chengdu, China, Mar. 2019, pp. 1368–1372, doi: 10.1109/ITNEC.2019.8729084.

[23] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Nov. 2011.

[24] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 80–87, Jun. 2018, doi: 10.1109/MWC.2018.1700303.

[25] Z. Ya-Ling, L. Kai, W. Shang-Ping, and S. Qin-Dong, "A multi-users searchable encryption scheme with proxy re-encryption," in *Proc. 10th Int. Conf. Comput. Intell. Secur.*, Kunming, China, Nov. 2014, pp. 563–567, doi: 10.1109/CIS.2014.167.

[26] J. Zhang, "Semantic-based searchable encryption in cloud: Issues and challenges," in *Proc. 1st Int. Conf. Comput. Intell. Theory, Syst. Appl. (CCITSA)*, Yilan, Taiwan, Dec. 2015, pp. 163–165, doi: 10.1109/CCITSA.2015.29.

[27] X.-J. Shi and S.-P. Hu, "Fuzzy multi-keyword query on encrypted data in the cloud," in *Proc. 4th Int. Conf Appl. Comput. Inf. Technol./3rd Int. Conf Comput. Sci./Intell. Appl. Inform./1st Int. Conf Big Data, Cloud Comput., Data Sci. Eng. (ACIT-CSII-BCD)*, Las Vegas, NV, USA, Dec. 2016, pp. 419–425, doi: 10.1109/ACIT-CSII-BCD.2016.088.

[28] C. Liu, L. Zhu, and J. Chen, "Efficient searchable symmetric encryption for storing multiple source data on cloud," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, Aug. 2015, pp. 451–458, doi: 10.1109/Trustcom.2015.406.

[29] D. Christopoulos, S. Chatzinotas, and B. Ottersten, "Cellular-broadcast service convergence through caching for CoMP cloud RANs," in *Proc. IEEE Symp. Commun. Veh. Technol. Benelux (SCVT)*, Luxembourg City, Luxembourg, Nov. 2015, pp. 1–6, doi: 10.1109/SCVT.2015.7374245.

[30] M. Garetto, E. Leonardi, and S. Traverso, "Efficient analysis of caching strategies under dynamic content popularity," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Kowloon, Apr. 2015, pp. 2263–2271, doi: 10.1109/INFOCOM.2015.7218613.

[31] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.

[32] V. Pacifici and G. Dan, "Distributed caching algorithms for interconnected operator CDNs," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 380–391, Feb. 2017, doi: 10.1109/JSAC.2017.2659118.

[33] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug. 2016, doi: 10.1109/TMC.2015.2474364.

[34] Y. Tang, "Minimizing energy for caching resource allocation in information-centric networking with mobile edge computing," in *Proc. IEEE Int. Conf Dependable, Autonomic Secure Comput., Int. Conf Pervasive Intell. Comput., Int. Conf Cloud Big Data Comput., Int. Conf Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Fukuoka, Japan, Aug. 2019, pp. 301–304, doi: 10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00062.

[35] F. Chao, Y. F. Richard, H. Tao, L. Jiang, and L. Yunjie, "A game theoretic approach for energy-efficient in-network caching in content-centric networks," *China Commun.*, vol. 11, no. 11, pp. 135–145, Nov. 2014, doi: 10.1109/CC.2014.7004531.

[36] H. Wu, J. Li, J. Zhi, Y. Ren, and L. Li, "Edge-oriented collaborative caching in information-centric networking," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Barcelona, Spain, Jun. 2019, pp. 1–6, doi: 10.1109/ISCC47284.2019.8969688.

[37] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016, doi: 10.1109/ACCESS.2016.2597169.

[38] W. Wang, P. Xu, D. Liu, L. T. Yang, and Z. Yan, "Lightweighted secure searching over public-key ciphertexts for edge-cloud-assisted industrial IoT devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4221–4230, Jun. 2020, doi: 10.1109/TII.2019.2950295.

[39] Y. Tao, P. Xu, and H. Jin, "Secure data sharing and search for cloud-edge-collaborative storage," *IEEE Access*, vol. 8, pp. 15963–15972, 2020, doi: 10.1109/ACCESS.2019.2962600.

[40] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "AKSER: Attribute-based keyword search with efficient revocation in cloud computing," *Inf. Sci.*, vol. 423, pp. 343–352, Jan. 2018.

[41] H. Zhong, Z. Li, J. Cui, Y. Sun, and L. Liu, "Efficient dynamic multi-keyword fuzzy search over encrypted cloud data," *J. Netw. Comput. Appl.*, vol. 149, Jan. 2020, Art. no. 102469.

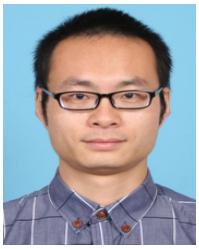
[42] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud-assisted Internet of Things," *IEEE Cloud Comput.*, vol. 4, no. 1, pp. 34–42, Jan. 2017, doi: 10.1109/MCC.2017.9.

[43] H. Zhong, Z. Li, Y. Xu, Z. Chen, and J. Cui, "Two-stage index-based central keyword-ranked searches over encrypted cloud data," *Sci. China Inf. Sci.*, vol. 63, no. 3, pp. 1–3, Mar. 2020.

[44] J. Cui, H. Zhou, Y. Xu, and H. Zhong, "OOABKS: Online/offline attribute-based encryption for keyword search in mobile cloud," *Inf. Sci.*, vol. 489, pp. 63–77, Jul. 2019.
[45] M. Sarra, B. Samia, S. Khaled, and D. Mehammed, "New caching system under uncertainty for mobile edge computing," in *Proc. 4th Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Rome, Italy, Jun. 2019, pp. 129–134, doi: 10.1109/FMEC.2019.8795356.
[46] X. Hu and J. Gong, "Distributed in-network cooperative caching," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst.*, Hangzhou, China, Oct. 2012, pp. 735–740, doi: 10.1109/CCIS.2012.6664272.
[47] H. Gu and H. Wang, "A distributed caching scheme using non-cooperative game for mobile edge networks," *IEEE Access*, vol. 8, pp. 142747–142757, 2020, doi: 10.1109/ACCESS.2020.3009683.
[48] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Oct. 2014, doi: 10.1109/TNET.2013.2294111.
[49] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Barcelona, Spain, Aug. 2014, pp. 917–921, doi: 10.1109/ISWCS.2014.6933484.
[50] J. Gu, W. Wang, A. Huang, H. Shan, and Z. Zhang, "Distributed cache replacement for caching-enable base stations in cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 2648–2653, doi: 10.1109/ICC.2014.6883723.
[51] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017, doi: 10.1109/TMC.2016.2597851.
[52] D. Ren, X. Gui, W. Lu, J. An, H. Dai, and X. Liang, "GHCC: Grouping-based and hierarchical collaborative caching for mobile edge computing," in *Proc. 16th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, Shanghai, China, May 2018, pp. 1–6, doi: 10.23919/WIOPT.2018.8362881.
[53] W. Zhao, Z. Chen, K. Li, N. Liu, B. Xia, and L. Luo, "Caching-aided physical layer security in wireless cache-enabled heterogeneous networks," *IEEE Access*, vol. 6, pp. 68920–68931, 2018, doi: 10.1109/ACCESS.2018.2880339.
[54] L. Zhu, M. Li, and Z. Zhang, "Secure fog-assisted crowdsensing with collusion resistance: From data reporting to data requesting," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5473–5484, Jun. 2019, doi: 10.1109/JIOT.2019.2902459.

JINTAO LING is currently pursuing the bachelor's degree with the Zhejiang University of Science and Technology. His research interests include edge cache and searchable encryption.

HUIXIAN GU is currently pursuing the degree with the Zhejiang University of Science and Technology. His research interests include edge cache, wireless communication, and searchable encryption.



HAIJIANG WANG received the M.S. degree from Zhengzhou University, in 2013, and the Ph.D. degree from Shanghai Jiao Tong University, in 2018. He is currently a Teacher with the School of Information and Electronic Engineering, Zhejiang University of Science and Technology. His research interests include cryptography and information security, in particular, public-key encryption, attribute-based encryption, and searchable encryption.

LEI ZHANG received the M.S. degree from Tsinghua University, in 2006. He is currently a Teacher with the School of Information and Electronic Engineering, Zhejiang University of Science and Technology. His research interest includes the communication and its security in the Internet of Things.