A Symmetric Lookup-based Secure P2P Routing Algorithm

Bingqing Luo[1], Yiai Jin[2], Shengmei Luo[2], Zhixin Sun[1]
[1]Key Laboratory of Broadband Wireless Communication and Sensor Network Technology
Nanjing University of Posts and Telecommunications, Nanjing, China
[e-mail: ice820@126.com, sunzx@njupt.edu.cn]
[2]Zhongxing Telecommunication Equipment Corporation, Nanjing, China
[e-mail : jin.yiai@zte.com.cn, luo.shengmei@zte.com.cn]
*Corresponding author: Zhixin Sun

Received April 26, 2015; revised August 6, 2015; accepted March 13, 2016; published May 31, 2016

Abstract

To prevent structured peer to peer (P2P) overlay networksfrom being attacked by malicious nodes, a symmetric lookup-based routing algorithm referred to as Symmetric-Chord is proposed in this paper. The proposed algorithm determines the precision of routing lookup by constructing multiple paths to the destination. The selective routing algorithm is used to acquire information on the neighbors of the root. Authenticity of the root is validated via consistency shown between the information ascertained from the neighbors and information from the yet-to-be-verified root, resulting in greater efficiency of resource lookup. Simulation results demonstrate that Symmetric-Chordhas the capability of detecting malicious nodes both accurately and efficiently, so as to identify which root holds the correct key, and provides an effective approach to the routing security for the P2P overlay network.

Keywords : Symmetric routing lookup, Selective routing, Distributedhash table, P2P

This work is supported by the National Natural Science Foundation of China (60973140, 61170276, 61373135), The research project of Jiangsu Province (BY2013011), The Jiangsu provincial science and technology enterprises innovation fund project (BC2013027), the High-level Personnel Project Funding of Jiangsu Province Six Talents Peak and Jiangsu Province Blue Engineering Project. The major project of Jiangsu Province University Natural Science (12KJA520003)

http://dx.doi.org/10.3837/tiis.2016.05.014 **ISSN : 1976-7277**

1. Introduction

How to locate resources in apeer to peer (P2P) network is a main measure of its performance. The resource lookup algorithm, based on the distributed hash table (DHT) is not only a recent breakthrough in P2P domaina, but also represents the future directionof the distributed routing algorithms for P2P networks. Typical algorithms encompass Pastry[1], Chord[2], CAN[3], and Tapestry[4]. In the context of recent developments in the new generation of information technologies (e.g. mobile internet, internet of things, and cloud computing), an increased application of the P2P technique and an expansion of P2P services inevitably imposes greater demands on the security of P2P systems [5][6].

Significant progress has already been achieved in the security of the structuredP2P overlay network. Methods to resolve routing lookup attacks have been proposed to increase system security. A security scheme for the P2P overlay network was proposed in [7], whereby the overlay layer can detect the bottom layer security settings and provide the P2P network with a valid security certificate. The node-based security certificate facilitates node selection and access control, thus offering improved network security. Haiying Shen proposed a resilient routing table in [8] to balance loads during lookup and the positioning of network resources. This scheme allows each node to maintain a routing table of varying size based on its energy. Despite its capacity to alleviate damage generated by malicious nodes from the variable routing table, this scheme's drawback is that it needs to maintain a large amount of system information.

In [9], Xu Xiangproposed a DHT routing protocol that ensures security through the detection, tracking and bypassing of malicious nodes. However, this method implements secure routing without taking into account routing efficiency and network load balance. Therefore, when the number of malicious nodes reaches a threshold, this scheme is prone to cause network congestion and paralysis. A social relation-based DHT secure routing scheme is proposed in [10], i.e. a scheme based on a trusted relationship between nodes relying on social relationhips. Instead of merely taking into account routing efficiency, the query node forwardsthe route based on social relation during the routing process. The only downside is that the mechanism for establishing the social relation relies on other widely used network services (e.g. Yahoo),and clearlythese network services are not readily available in all scenarios.

A secure and robust DHT routing algorithm called DHTBL is proposed in [11]. DHTBLpossesses the advantage of the introduction ofan anti-attack blacklist to the secure DHT routing scheme. The authors proceed to detail how to surpass the existing DNSBLS strategy with DHTBL, and also prove beyond a reasonable doubt that DHTBLis effective in theresistance of DOS attacks and ensures the correct delivery of messages. Nonetheless problems arise in respect to the adopted neighbor monitoring-based certificate management if the certificate is invalid, or if the node changes or leaves quickly. In the case of the neighbor's certificate being invalid, the new node fails tobe added. Furthermore, this structure is damaging to the P2P structure.

Jia Xu provided an improved version (i.e. CloudSEC) of the Chord algorithm in [12]. The CloudSECarchitectureis a dynamic peer overlay comprised of three types of structural components from top to bottom. In addition to its effective data querying resources, in the heterogeneous network security infrastructure this architecture can also perform

data-intensive tasks by using the availible set of distributed computing resources. Unfortunately this method substantially adds to the system's complexity. To avoid attacks caused by the random allocation of node IDs in Pastry, a novel architecture and algorithm re-named SEPastry is proposed in [13], which is capable of vigorously resisting various attacks against node IDs without computing cipher texts. The authors however failed to consider routing tablemaintenance and asecure routingstrategy.

In their endeavors to resolve these issues, theauthors in [14][15] provided each node with an additional pointer table that stores node information counterclockwise. The routing information can thenbe transmitted either counterclockwise or clockwise, but attention was focused on improving the efficiency of the routing query and neglectedpotential attacks by malicious nodes.

In view of the above analysis, the existing problems of the solution for the structured P2P routing security can be summarized as follows:

(1) Locate the resource with the correct information delivery quickly by improving the route table or increasing the authentication. The problem withthis kind of method is that it islikely to cause routing inefficiencies, as well as the destruction of the P2P network structure.The establishment of a certification center causesmalicious nodes topermeate through the whole system easily.

(2) To improve messaging correctness, it is not based on the optimal efficiency of routing, rather it isbased on a relationship of trust, including social relations, with the relationship between the physical connections. The problem with this approach is the established mechanism of the class relationship generally depends on other web services;

(3) Selective route is based on the optimal routing efficiencyand lookup of the right target node through redundant routing.The problem of these methods is that redundant queryis easier to lead to significant network traffic load.

To address these routing security problems of P2P applications based on the distributed hash table, this paper presentsa symmetric lookup-based P2P secure routing algorithm called Symmetric-Chord. Symmetric-Chord supports two way query in the independent clockwise and anticlockwise routes to locate the keyword, and when the results are different in the two ways, the algorithm provides the root validating mechanism to ensure the loyalty of the nodes.Symmetric-Chord provides an approach to secure delivery of routing information during the key-based lookup, avoiding malicious attacks during the process of routing lookup and improvesefficiency of resource lookup.

2. Symmetric Lookup-based P2P Secure Routing Algorithm

The symmetric lookup-based P2P secure routing algorithm presented by this paper allows applicants to query the independent clockwise and anticlockwise routes. The two returned query results are compared in order to determine whether malicious nodes exist and whether the routing process is correct. By this method, the correctness of the query results can be identified in a simple and effective fashion. If the query results are inconsistent, then the system will run the root validating mechanism. In order to achieve this, selective routes are used togather this data about the neighbor of the to-be-validated root. Subsequently, consistency between the information of the to-be-validated root and the information of its neighbor will be determined to additionally validate the authenticity of the node.

Definition 1: Refers to the set of routes that do not pass the specified nodes during the search for the route of the key. By performing this procedure, while validating the root, we can bypass the potentially malicious nodes identified during the previous routing process, thus avoiding their impact on validation security.

Definition 2: Malicious nodes refer to those that do not strictly adhere to the original routing rules, and thereby mislead other valid nodes with wrong information, with the consequential effect of forwarding the routes incorrectly, colluding with other malicious nodes, and abandone lookup information and any other nodes that lead to the failure in the search of the key.

The conditions for routing security of the P2P overlay network include secure allocation of IDs to nodes, secure maintenance of the routing table, and secure routing. Symmetric-Chord achieves secure routing through the detection of malicious nodes via symmetric lookup. It can then securely maintain the routing table using the secure routing algorithm. Secure allocation of IDs to nodes is out of the scope of this paper. In this paper, it is assumed that the IDs are securely allocated to the nodes, i.e. the attackers are unable to obtainmany or consecutive node IDs.

2.1 Definition of the algorithm's data structure

In order for symmetric lookup and selective routing to be implemented, and so that malicious nodes can be detected via this secure routing algorithm, the routing table (pointer table) in the Chord algorithm is modified to enable symmetric lookup. Different paths to the query root (i.e. the node responsible for the queried key) are also built to check the accuracy of the root.In the modified Chord system, each node stores the clockwise pointer table (i.e. the original pointer table) and the added anticlockwise pointer table. The items in both tables are centered on the current node and symmetric clockwise and anticlockwise. The clockwise and anticlockwise pointer table recorded by the node n is given in **Table 1**and **Table2**, respectively.

Table 1. Definition of the pointer table (clockwise) of node n in the Chord system where the node label consists of m digits

Labels	**Definitions**	**Names of sub-items**
cw-finger[k].start	$(n+2^{k-1}) \bmod 2^m, 1 \le k \le m$	The starting value of the labelled interval pointed by the k[th] item in the pointer table of node n
cw-finger[k].interval	[cw-finger[k].start, cw-finger[k+1].start]	The separationof the labelled interval pointed by the k[th] item in the pointer table of node n
cw-finger[k].node	first node ≥ n. cw-finger[k].start	The forwarding nodeof the interval pointed by the k[th] item in the pointer table of node n
cw-successor	Direct successor of node n,cw-finger[1].node	Successor
cw-predecessor	Clockwise direct predecessor of node n	Predecessor

Table 2. Definition of the pointer table (anticlockwise) of node n in the Chord system where the node label consists of m digits

Labels	***Definitions***	***Names of sub-items***
anticw-finger[k].start	$(n-2^{k-1}+2^m) \bmod 2^m, 1 \le k \le m$	The starting value of the labelled interval pointed by the k^{th} item in the pointer table of node n
anticw-finger[k].interval	[anticw-finger[k].start, anticw-finger[k+1].start]	The separationof the labelled interval pointed by the k^{th} item in the pointer table of node n
anticw-finger[k].node	first node $\le$ n. anticw-finger[k].start	The forwarding nodeof the interval pointed by the k^{th} item in the pointer table of node n
anticw-successor	anticw-finger[1].node	Successor
anticw-predecessor	Anticlockwise direct predecessor of node n	Predecessor

According to the modified version of the Chord pointer table, in the case where m=3, the node labels are 0, 1, 3 and the improved Chord system where the keys are 1, 2, and 6 are illustrated in **Fig. 1**.

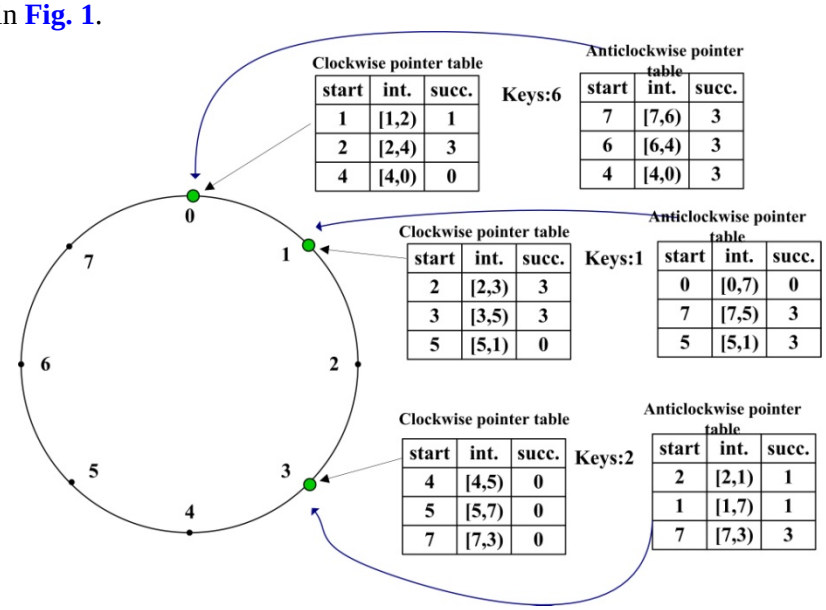


Fig. 1. Distribution of the keys 1,2 and 6 in the improved Chord system consisting of nodes 0, 1 and 3 when m=3

Details of the anticlockwise search for key 1 from the node 0 is as follows. The anticlockwise successor of node 0 is the node 3, and key 1 $\notin (0,3]$ anticlockwise. So in the anticlockwise pointer table, we search for the predecessor closest to key 1in a bottom-up fasionand ascertain node 3. Node 0 will submit the query request to node 3. The anticlockwise successor of node 3 is node 1 and key $1 \in (3,1]$. Therefore node 3 is the anticlockwise predecessor of key 1. Considering that key 1 is equal to the successor of node 3, the query

requestis finally delivered to node 1. That is to say, the anticlockwise query requestis complete.

In this paper, a Chord structure-based local recovery method is used to detect malicious nodes. In other words, the actual DHT structure of the local area is recovered with multiple neighbors of the suspicious node. Consequently, we can determine whether the suspicious node creates a wrong route and if this node is malicious or fails. A table of the information on the neighbors is added to the Symmetric-Chord. The neighbor information table of each node stores q predecessors and q successors (the value of q is m+1). The neighbor information table of node n is defined in **Table 3**.

Table 3. Definition of the neighbor information table of node n in the Symmetric-Chord system

Number	Successors	Predecessors
1	successor(n)	predecessor(n)
2	successor(successor(n))	predecessor(predecessor(n))
……	……	……
q	$\underbrace{\text{successor(successor(.....(n).....))}}_{q}$	$\underbrace{\text{predecessor (predecessor (……(n)……))}}_{q}$

In Symmetric-Chord, all node labels form a ring. Hence, their relative positions can be used to determine whether a node is the intermediate node of two other nodes, as is shown in **Fig. 2**.

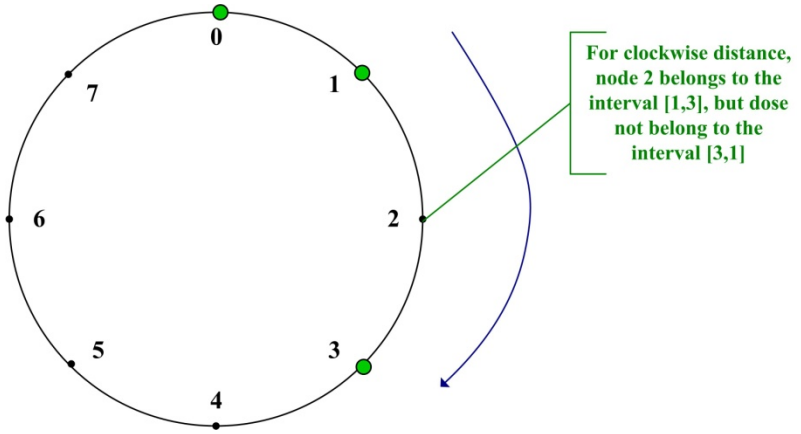


Fig. 2. illustration of checking whether a node is within an interval

For the interval [A, B], we compute the clockwise distance as $(B+2^m-A)\%2^m$, and the anticlockwise distance as $(A+2^m-B)\%2^m$. In **Fig. 3**, for the given interval [1, 3], the clockwise distance between nodes 3 and 1 is $(3+2^m-1)\%2^m=2$, and the distance between nodes 2 and 1 is $(2+2^m-1)\%2^m=1$.Because the distance between nodes 2 and 1 is less than the distance between nodes 3 and 1, node 2 belongs to the interval [1, 3]. Also considering the interval [3, 1], the distance between the nodes 1 and 3 is $(1+2^m-3)\%2^m=6$, and the

distance between nodes 2 and 3 is $(2+2^m-3)\%2^m=7$. The distance between nodes 2 and 3 is less than that between nodes 1 and 3. Therefore node 2 does not belong to the interval [3, 1].

2.2 Details of the symmetric routing algorithm

Symmetric-Chord detects malicious nodes bya process of two-way lookup. During the query, an iteration strategy is used to improve routing security. The iterative routing algorithm is as follows. The node which initiates the query acquires the next routing node n to be queried from its pointer table, and directs the query request forward to n in order to find the next route. After receiving the query request, node n searches its pointer table for the next routing node n', and then returns the information relating to n' to the query initiating node. After receiving the reply from n, the query initiator forwards the query request to n' in order to find the next node n''. Query nodes continually repeatthis process until the root key is found. By using the iterative routing algorithm, the query initiator can determine the intermediate nodes passed by the route. Thus, with the information aboutthese intermediate nodes, the query initiator is then capable of obtaining details of how it approaches the root key.

Assume that the query initiator obtains the next node n' via the intermediate node n along the route. Based on the definition of the neighbor information table of node n in the Symmetric-Chord system, on receiving the request made by the query initiator to query the key, n' returns different results to the query initiator in the following three cases.

(1) If $\text{key} \in [\text{n},\text{predecessor}(\text{n}^{'})]$, then it means that it is wrong for n to return n' as the next routing node to the query initiator, because it should return the node before the predecessor(n'). The query goes wrong and node n is malicious.

(2) Considering the clockwise query,if $key \in (cw-predecessor(n),\ n]$. As for the anticlockwise query, if $key \in [n^{'}, anticw-successor(n^{'}))$, then it means that $n^{'}$ is the root responsible for the key. The query in this direction is finished.

(3) If neither of the two cases above holds true, it means that $key \notin (\text{n}^{'},\text{n})$. But the query is not over. n' acquires the next node n'' from the pointer table in this direction and returns it to the query initiator. For clockwise routing, if we fail to find a node that is not only closer to the key than n' and is located between n' and the key, then it means that n' is the predecessor of the key, and $anticw-successor(n^{'})$ is the root of the key. Hence, n' needs to return $anticw-successor(n^{'})$ to the query initiator.

The query initiatorsimultaneously initiatesthe query process in both the clockwise and anticlockwise directions. Whilemaking a query, we can gradually obtain the intermediate nodes between the query initiator and the root key. The query can gradually approach the root in both directions. During the query process, if an intermediate node replies with a wrong query, this means that suspicious nodes may occur during the query, and the query then fails. If the root keys obtained in the clockwise and anticlockwise directions are the same node when the query is complete, this means that the query achieves the desired result and the query succeeds.

2.3 Root validating mechanism

Symmetric-Chord provides a two way query in the independent clockwise and anticlockwise routes to locate the keyword, and if the query was successful or not can be judged according to the query results consistency. When the results in the two directions are different, it means that there are malicious nodes in the query route. Then the Symmetric-Chordemploys the root validating mechanism to recognize the potential malicious nodes and ensure the correct root node which stores the keyword. The root validating mechanism is addressed by setting the global detection center, which is set in the whole P2P network. Based on the system capacity and safety state, the global detection center is composed of one or several servers. When an error is found during query, the query results and process will be submitted to the global detection center, and the center validates the root node through selective routing algorithm .

If the query initiator discovers that the query has gone wrong and acknowledges that there may be malicious nodes, the global detection center validates the roots via the selective routing algorithm. After receiving the message submitted by the query initiator that the query has failed, the detection center requests the information regarding the neighbor and pointer tables from the routing nodes passed during the query process. Each P2P node proceeds to return its neighbor and pointer tables. When the information is received from the routing nodes, the detection center immediately checks whether the behavior of each node was consistent with its neighbor and pointer tables during the query. Any inconsistency indicates that there are malicious nodes present along this path. Alternatively, the detection center will use the selective routing algorithm to obtain the neighbor and pointer tables of the q predecessors and successors of the roots acquired by querying in the clockwise and anticlockwise directions. These tables will then be employed to form the regional structural information. By checking the consistency between regional information, we can detect malicious nodes and determine whether the root is the correct root key.

To prevent the process of obtaining the immediate predecessors and successors of the to-be-validated nodesfrom being affected again by the malicious nodes while validating the root, the immediate predecessors and successors of the query root are obtained through the selective routing algorithm. When this algorithm is used to obtain the neighbors of a root, the routing path will not encompass the intermediate nodes passed by the symmetric routes. Under the given conditions, a path such as this likely exists:

Theorem: Let m denote the number of digits of the key, and q denote the number of predecessors and successors stored by each node in the symmetric lookup-based Chord system. If $q \geq m+1$, then by starting with the query initiator, there is at least one path which is not only capable of acquiring information aboutthe neighbors of the symmetric query root, but also contains no intermediate routingnodes in the process of the symmetric query.

Proof: Let n_0 denote the query initiator, n_{key} denote the responsible root key, and $n_1 n_2 ... n_m$ denote the intermediate nodes passed by the route. So all routing nodes that need to be queried are $\{n_0 n_1 ... n_m, n_{key}\}$, where $\{n_0 n_1 ... n_m\}$ which form the intermediate routing nodes.

n_1 is the i th item of the pointer table of n_0, $dis(n_1, n_0) \geq 2^{i-1}$;

So the distance between n_1 and n_{key} is $dis(n_1, n_{key}) \leq 2^{i-1}$;

Andthe distance between n_1 and n_{key} is at most halfthe distance between n_0 and n_{key}.

Similarly, assuming $n_k \in \{n_1, n_2, ...n_m\}$, then the distance between n_k and n_{key} is at most half the distance between n_{k-1} and n_{key}. Considering that the largest distance between n_0 and n_{key} is $2^m - 1$, n_0 can reach n_{key} by a maximum of m hops.

Assuming that n_0 is not the direct predecessor of n_{key}, i.e. n_{key} is not among the successors of n_0, then there are q nodes belonging to $\{n_{0,} n_{key}\}$. Because $q \geq m+1$, there must exist intermediate nodes $n_1' \notin \{n_{0,} n_{1...} n_m\}$ which can choose n_1' as the next route node for the querying the neighbors of n_{key}.

If n_{key} is not the successor of n_1', then there must exist $n_2' \notin \{n_{0,} n_{1...} n_m\}$.

These procedures are repeated until n_t',or one of the neighbors of n_{key}, is reached during the query. Then, $\{n_1', n_2', ... n_t'\}$ constitutes the intermediate nodes passed by the route which starts with the query initiator and ends with one of the neighbors of n_{key}. And that it is exclusive of $\{n_1, n_2, ..., n_m\}$.

To further improve query efficiency, the pointer table is used in the selective routing algorithm to obtain the next routing node. If the next route in the pointer table does not meet the conditions, then we choose the next route from the neighbor table. After receiving the information regarding node n_t', the detection center can acquire theinformation on all neighbors of n_{key} and check the consistency of regional information to determine whether n_{key} is the correct root key. The pseudo code of the detecting algorithm for the detection center is in **Fig. 3**.

```
procedure _check _rootkey(key,{n_0, n_1, ... n_m, n_key})
n = n_0
while(n_key ∉ neighbor(n))
// look up the next routing node from down to up in pointer table
if(n' ∈ finger(n) and n_1' ∈ (n, key] and n_1' ∉ {n_1, ... n_m})
n = n_1'
get _other _neighbor _info(n)          // acquire other neighbor information of n_key
analyse _area _info(n_key, neighbor(n_key))   //check consistency of regional information
return check_rootkey_valid(n_key)              //return the result
```

Fig. 3. The pseudo code of the algorithm for checking the authenticity of the root

In summary, on completion of the first symmetric lookup-based routing process, a comparison between the two query results can verify the accuracy of the routing process in a simple and effective manner. If the query results are inconsistent, this suggeststhat problems exist with the query results obtained in eitherdirection. The proposed algorithm designs a mechanism for verifying the query root, and is furthermore capable of detecting malicious nodes and correct roots through verification. Therefore, even if given the circumstances that the symmetric routing process has been attacked by malicious nodes, we can offer protection through the comparison of query results and the use of the verification mechanism. For example, an effective defense can be providedagainst the malicious nodes colludingwhile

forwarding routes or intentionally pretending to be the root key. This ensures prompt discovery of malicious attacks before the submission to a higher layer for the acquisition of the resource list, resulting in greater accuracy of key query and improved routing security.

Malicious nodes can make the common node fail by causing an inconsistency between the system's reality and the node specifiedby the pointer table or the neighbor table of the common node. Becausethe route forwarding error caused by the failed node is identical to the route forwarding error caused by the malicious node, we can thus use the same security mechanism for the detection and processing of the failed nodes and malicious nodes in this paper.

3.Performance Analysis and Simulations

3.1 Performance analysis

In Symmetric-Chord, while initiating the query, the query initiator needs to perform a query in both the clockwise and anticlockwise directions. Let $Delay_{clockwise}$ and $Delay_{anticlockwise}$ denote the query delays in the two directions. The total query delay has a maximum delay of max($Delay_{clockwise}$, $Delay_{anticlockwise}$). This will decrease query efficiency. Details of the analysis are given below.

Assume the possibilitythat the root of the queried resource is located at a position within the ring that follows the uniform distribution, i.e. $P = 1/2^m$. In the case of the structure consisting of sparse nodes, DHT is highly efficient in query, addition, exit and maintenance. Therefore we can assume that the node distribution approximates to the level of saturation, and the obtained result represents the theoretical lower boundary of system efficiency.

First, we compute the average length of the path (i.e. number of hops) for the one-wayquery in the original Chord system. Starting with the current query node n, we regard the Chord ring as the n-ary tree rooted on n. The root of each sub-tree is the node specifiedby the pointer table, and each sub-tree is similar to the n-ary tree. The preorder search process of the m-ary tree is consistent with the Chord search process. From the analysis, one can observe that C_m^k nodes can be reached via k hops from n. According to the above assumptions, the positions of the queriedsource nodes in the ring follow a uniform distribution. Hence, the average length of the path for the query in the one-way Chord system is $E_{hop} = \sum_{i=0}^{m} i * C_m^i * 1/2^m = m/2$.

For the Symmetric-Chord query, the length of the path in each query is $\max(d_{clockwise}, d_{anticlockwise})$. According to the properties of the Chord query algorithm, we have $d_{clockwise} + d_{anticlockwise} <= m+1$. By using a combined number computation method, the average path length can be calculated as follows:

(1) When m is even:

$$E_{hop(even)} \le \sum_{i=1}^{m/2} (m+1-i) * C_m^i * 1/2^m + \sum_{i=m/2+1}^{m} i * C_m^i * 1/2^m$$

$$= \left[\sum_{i=1}^{m/2} (m+1) * C_m^i - \sum_{i=1}^{m/2} i * C_m^i + \sum_{i=m/2+1}^{m} i * C_m^i \right] * 1/2^m == \left\{ (m+1) * \left[2^{m-1} + m!/(m/2)!/(m/2)! - 1 \right] - m * 2^{m-2} + m * 2^{m-2} \right\} * 1/2^m$$

According to the Stirlingapproximate formula, we have:

$$E_{hop_{(even)}} \le (m+1)/2 + (m+1)/\sqrt{2\pi m} + o(1)$$

$$E_{hop_{(odd)}} \le \sum_{i=1}^{(m-1)/2}(m+1-i)*C_m^i *1/2^m + \sum_{i=(m+1)/2}^{m} i*C_m^i *1/2^m$$

$$= \left[\sum_{i=1}^{(m-1)/2}(m+1)*C_m^i - \sum_{i=1}^{(m-1)/2} i*C_m^i + \sum_{i=(m+1)/2+1}^{m} i*C_m^i\right]*1/2^m$$

$$= \left\{(m+1)*(2^{m-1}-1) + m!/[(m-1)/2]!/[(m-1)/2]!\right\}*1/2^m$$

Similarly, accordingto the Stirlingapproximate formula, we have:

$$E_{hop_{(odd)}} \le (m+1)/2 + (m)/\sqrt{2\pi(m-1)} + o(1)$$

From the computation above, it can be seen that the Symmetric-Chord performs a symmetric search, and therefore its number of hops is on average larger than that of the one-way query by $1/2 + (\sqrt{m/2\pi} + o(1)$. For the Symmetric-Chord security system that uses the SHA-1 hash algorithm to generate 160-digit node IDs, it on average incurs an additional 5.5 hops. The average number of hops in the original Chord system, based on one-way query, is $m/2 = 160/2 = 80$. Compared with the query delay increment of 6.88%, the cost incurredis acceptable.

3.2 Simulations

In this paper, BRITE[16] is used to generate AS network topology, with the nodes following the Heavy Tailed distribution. HOP is used to represent the query efficiency of the simulations. The bandwidth distribution is ofthe Constant type. Connections between nodes are considered to be the same. Byzantine is chosen as the malicious node model. Let N denote the number of nodes, and f ($0 \le f < 1$) denote the proportional coefficient of malicious nodes. These malicious nodes are partitioned into independent sets.The malicious nodes in each are able to collude with one another. The size of the set can be modified bycoefficient c ($1/N \le c \le f$). If $c = f$, all malicious nodes can collude with each other, thus posing the greatest threat against the system. During the simulations, these sets of malicious nodes damage the correct routes by randomly forwarding the wrong routes or colluding.

Fig. 4 shows the possibility of successful routing for the Symmetric-Chord and the original Chord systems with varying proportions of malicious nodes. Experimental results show that when the proportion of malicious nodes f, is less than 20%, query accuracy in the Symmetric-Chord system can reach 98%, approximating to 100%. This is because the possibility of suffering malicious attacks during the routing process is low in the case of the proportion of malicious nodes being small. So the query accuracy is high for both the Symmetric-Chord and the original Chord systems. However, the Symmetric-Chord system provides the root validating mechanism, which enables the detection center to validate without using the intermediate routing nodes previously acquiredas soon as any inconsistency in query results is detected. The root can be determined effectively during the validation process, guaranteeing query accuracy. Conversely, in the original Chord system, once malicious nodes are encounteredduring the routing process, the message that the query has failed is returnedafter the query is complete. As a consequence, query accuracy significantly decreases with an increase in the number of malicious nodes in the system.

If the proportion of malicious nodes exceeds 30%, and assuming there is a need for the validation mechanism, due to the increased possibility of again encountering malicious nodes in the process of obtaining the information on the neighbors of the to-be-validated root, and to

the high possibility of the query initiator emerging asthe malicious node, the validation process is severely affected, andthus, resulting in a decreased accuracy of the query nodes returned by the validation process. Hence, the query accuracy of the Symmetric-Chord system substantiallydecreases.However, taking all matters into consideration, the Symmetric-Chord system remains superior to the original Chord system.

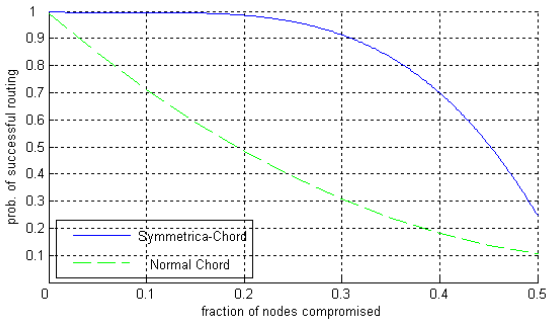


Fig. 4. Comparison of routing success rate between the Symmetric-Chord system and the original Chord system

In the Symmetric-Chord system, following the detection of malicious nodes, the security mechanism in place will insulate the malicious nodes from theP2P system. Therefore, the proportion of malicious nodes will gradually decrease over time. **Fig. 5** shows the relationship between the proportion of malicious nodes in the Symmetric-Chord system and the system operation time. Following a decrease in the proportion of malicious nodes,collusions between malicious nodes will also decrease. In this case, their separate malicious behaviors can be promptly detected and the malicious nodes are insulated from the P2P system, resulting in a quicker decrease of the proportion of malicious nodes within the system.

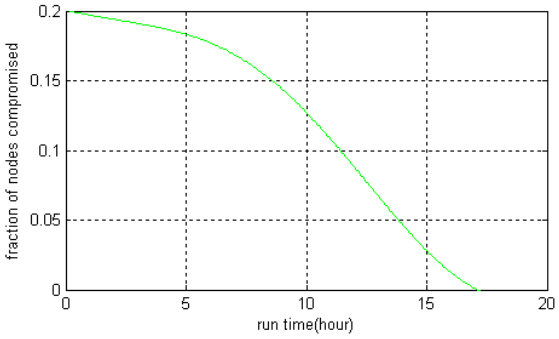


Fig. 5. Changes in the proportion of malicious nodes in the Symmetric-Chord system over time

The symmetric lookup-based Chord system performs a two-way query, with query efficiency depending on the direction of which query is slower. Its overall efficiency is thereforecompromised. **Fig. 6** compares the average length of the query path in the Symmetric-Chord system and the original Chord system. From the figure, it can be observed that the average length of the query path in the original Chord system generally follows $(\log N)/2$, with its average path length increasing linearly with the logarithm of its respective number of nodes. In the case of the number of nodes being small, the average length of the query path in the Symmetric-Chord system increases quickly with an increase in the number of

nodes. However, when the number of nodes approximates to the saturation point, the average length of the query path increases slowly and has an overall fluctuation around $1/2+(\sqrt{m/2\pi}+m/2)$. This is consistent with the analysis result on the algorithm's performance.

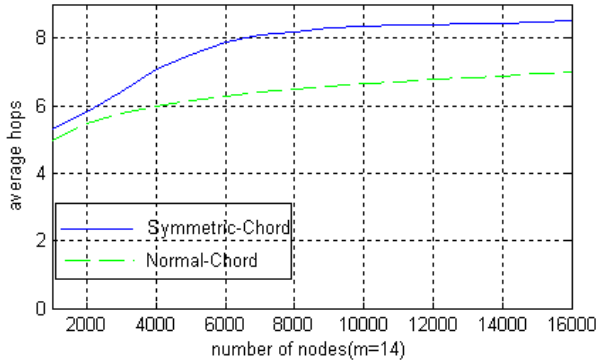


Fig. 6. Comparison of the average length of the query path between the Symmetric-Chordsystem and the original Chord system

4. Conclusions

In this paper, we analyze the security threats to routing query for P2P applications based on the distributed hash table. The Symmetric-Chord algorithm that relies on symmetric lookup and the selective routing method is proposed. Symmetric-Chord is actually a redundant routing algorithm, which checks the accuracy of the query by building different paths to the destination. In Symmetric-Chord, information on neighbors is used to check the authenticity of the root key. Details of the proposed algorithm are described and the impact on the query efficiency is analyzed. Simulations are performedwith the results demonstrating that the symmetric lookup-based algorithm is capable of guaranteeing the securityof the route based on the distributed hash table.

The distributed properties of the P2P applications determine that its security check solely relies upon the information about the nodes within the system. However, this information may either provide a correct message or sometimes even a wrong message owing to malicious nodes. Investigation on how to choose nodes which store the query information from numerous other nodes and how to obtain the correct information from the collected data, is a highly complex procedure. In future work, we will try to address the routing security problem by building redundant routes for other DHT algorithms. We will also study the introduction of resource duplication as a potential approach to resolving the P2P routing security problem.

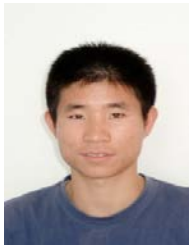
References

[1] Rowstron A, Druschel P, "Pastry: Scalable, decentraized object location and routing for large-scale peer-to-peer systems," in *Proc. of the 18th IFIP/ACM International Conference*, pp. 329-350, November 13-18, 2001. Article (CrossRef Link)

[2] Stoica I, Morris R., Karger D, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol.31, no.4, pp.149–160, Octorber, 2001. Article (CrossRef Link)

[3] Ratnasamy S, Francis P, Handley M., "A Scalable Content Addressable Network," *ACM SIGCOMM Computer Communication Review*, vol.31, no.4, pp.161-172, Octorber, 2001. Article (CrossRef Link)
[4] Zhao B Y, Kubiatowicz J D, Joseph A D., "Tapestry:An infrastructure for fault-tolerant wide-area location and routing," *Technical Report* ,UCB/CSD-01-1141, 2001.
[5] Sheng W Y, Ranasinghe Q Z., "P2P object tracking in the internet of things," in *Proc. of Parallel Processing (ICPP*), pp. 502-511, September 13-16, 2011. Article (CrossRef Link)
[6] Zheng L, Zhang H, Han W, et al., "Technologies, applications, and governance in the Internet of Things," *Internet of things-Global technological and societal trends*, Vermesan, Ovidiu and Friess, Peter, pp:141-175, 2011.
[7] Liu Y T, Qiu X F, Ji Y, "A Novel Security Mechanism to Defend Cross-Layer Security Threats in P2P Network," in *Proc. of Internet Technology and Applications (iTAP)* , pp. 1-4, August 16-18, 2011. Article (CrossRef Link)
[8] Shen H Y, Xu C Z, "Elastic Routing Table with Provable Performance for Congestion Control in DHT Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol.21, no.2, pp.242-256, April, 2010. Article (CrossRef Link)
[9] Xu X., "Providing Efficient Secure DHTs Routing," in *Proc. of Computational Intelligence and Security*, pp. 510-514, December 11-14, 2009. Article (CrossRef Link)
[10] Villanueva. R, Villamil. MD, Arnedo M, "Secure Routing Strategies in DHT-Based Systems," *Data management in grid and peer-to-peer systems*, vol.6265, pp. 62-74, 2010. Article (CrossRef Link)
[11] Bender A, Sherwood R, Monner D, "Fighting Spam with the Neighborhood Watch DHT," in *Proc. of INFOCOM*, pp. 1755-1763, April 19-25, 2009. Article (CrossRef Link)
[12] Xu J, Yan J, He L, "CloudSEC: A Cloud Architecture for Composing Collaborative Security Services," in *Proc. of IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 703-711, November 30-December 3, 2010. Article (CrossRef Link)
[13] Mishra M, Tripathy S, Peri S., "SEPastry: Security Enhanced Pastry," *Advances in Computing and Information Technology*, vol.176, pp. 789-795, July 13-15, 2012. Article (CrossRef Link)
[14] Ganguly. A, Boykin. Po, Wolinsky. Di, "Improving peer connectivity in wide-area overlays of virtual workstations," *Cluster computing-the journal of networks software tools and applications*, vol. 12, no.2, pp.239-256, June, 2009. Article (CrossRef Link)
[15] Chen HW, Ye ZW, "BChord: Bi-directional routing DHT based on chord," in *Proc. of the 12th International Conference on Computer Supported Cooperative Work in Design*, pp.410-415, April 16-18, 2008. Article (CrossRef Link)
[16] Brite Network topology generator [EB/OL]. http://www.csbu.edu/brite/.

Luo Bingqing was born in Ju Rong, Jiang Su Province, in 1987. She is now a Ph.D candidate in the college of Internet of Things, Nanjing University of Posts and Telecommunications. Currently her research interests include IP wireless sensor network, wireless sensor network communication, mobile Internet, etc.



Jin Yiai graduated from Jilin University in 2005. He has been working in ZTE Corporation for more than ten years, and engaged in cloud computing and bigdata security research.

Luo Shengmei received his Master degree in Harbin Instituted of Technology in 1996，majoring in communication and electronic. Now he is the chief architect of ZTE Corporation, his research interests include cloud computing, network storage and big data.

Sun Zhixin was born in Xuan Chen, An Hui Province, in 1964. He is now a professor and Ph.D. supervisor of the college of Internet of Things in Nanjing University of Posts and Telecommunications.Currently his research interests include computer networks and security, multimedia communication, mobile Internet, etc.