NodeId Verification Method against Routing Table Poisoning Attack in Chord DHT

[1]Avinash Chaudhari, [2]Pradeep Gamit

[1]L.D. College of Engineering, Information Technology,
Ahmedabad India
[1]Chaudhari.avi4u@gmail.com, [2]pradeep_gamit@yahoo.com

***Abstract*— Structured P2P networks are growing very fast because of their scalability, efficiency and reliability. Chord is one of the popular DHTs which generate its NodeId by hashing the IP address of node. As there is no mechanism to verify whether the NodeId is generated correctly or not, malicious nodes can generate multiple NodeIds easily. So genuine node's routing tables can be easily poisoned with fake NodeIds. In this paper we proposed the NodeId verification mechanism which will verify whether NodeId is generated correctly or not before updating the routing table entry. We will then analyse and compare the results of mitigation mechanism under attack.**

***Index Terms*— peer-to-peer, overlay network, distributed hash table, poisoning attack, Sybil attack.**

1. Introduction

Because of the lake of centralize authority, Peer-to-peer technologies are vulnerable to different kind of attacks. These networks are overlay networks which are organized on top of underlay network like internet. In overlay network, NodeId is generated to uniquely identify each node. In Chord DHT, this NodeId is generated by hashing the IP address of the node. Each node itself generate the NodeId by hashing its IP address and maintains the routing tables having its neighbor node's NodeIds.

In existing Chord protocol, there is no mechanism to verify the NodeId whether it is generated correctly or not. So that malicious nodes are allowed to generate multiple Sybil identities. As a result, genuine nodes can be poisoned by fake Sybil identities. This reduces the successfulness of lookup process and make whole network vulnerable. To resolve this issue, we proposed the NodeId verification mechanism. In this mechanism, NodeId is generated by hashing the combination of IP address and public key of that node. Furthermore, before updating routing table entry, node will verify whether NodeId is generated correctly or not. So Sybil identities will be identified and routing table poisoning can be prevented.

In this paper, we included the simulation and analysis of Chord protocol under Sybil attack and routing table poisoning attack. Our proposed mechanism is implemented and simulation results are shown which proves that the impact of routing table poisoning attack can be decreased by using this mechanism. The rest of the paper is organized as follows: Section 2 gives brief introduction about structured P2P networks and Chord protocol. In Section 3 related work will be discussed. Section 4 will describe our proposed mechanism. Section 5 shows the simulation results and analysis. Section 6 describes the future work and operational issues, and section 7 concludes.

2. Background

2.1 Structured P2P Network

DHT based p2p is a structured P2P. DHT is said to construct structured overlay network over existing network, such as internet. Figure 1 shows the exact picture of how nodes are connected to each other in overlay networks over underlay network. DHTs are completely decentralized, while DNS based systems creates hierarchy, in which some of the nodes have more central role and others have less. DHTs in contrasts to DNS, dynamically decides which node is responsible for which items. If node responsible for some items leave the network, DHTs self manages by giving the responsibility of those items to another nodes.

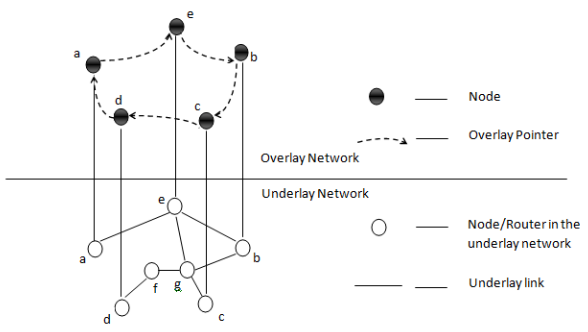


Figure 1 Overlay network over underlay network [8]

2.2 Chord DHT

Chord is one of the popular DHTs which support just one operation: given a key, it maps the key onto a node [1]. Figure 2 shows the chord ring in which the nodes are organized into the identifier space. Chord uses **Consistent Hashing** for assigning an m-bit unique identifier to each node by hashing the IP Address of the node. Each file in the DHT is also assigned a unique identifier in the same address space. There are two parts in identifier: key and value. Key is the hash value of file name and value is the hash value of file content.

Consistent hashing assigns keys of files to the nodes as follows. Node identifiers are ordered in the identifier circle modulo 2^m. Key k is assigned to the first node whose identifier is equal to or follows the identifier of k in identifier circle. This node is called the successor node of key k and denoted by *successor (k)* [1]. Figure 2 shows the identifier circle with m = 3. In this circle there are three nodes with identifier 0, 1 and 3. The successor of identifier 1 is node 1, so file with this identifier will locate at node 1. Same way the file with key 2 is located at node 3 and key 6 at node 0.

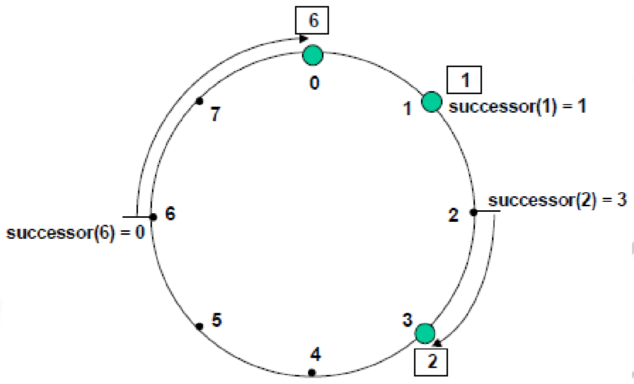


Figure 2: Chord ring consisting of three nodes 0, 1 and 3 [1]

Chord also manages the key look up in the minimum hop count of *log n*. If node 1 requests for key 6, the request will be go to the node 3. As the node 3 is not responsible for the key 6, without responding to node 1, the request will forwarded to the next node 0. Finally node 0 responds to the node 1 with successful look up.

3 Related work

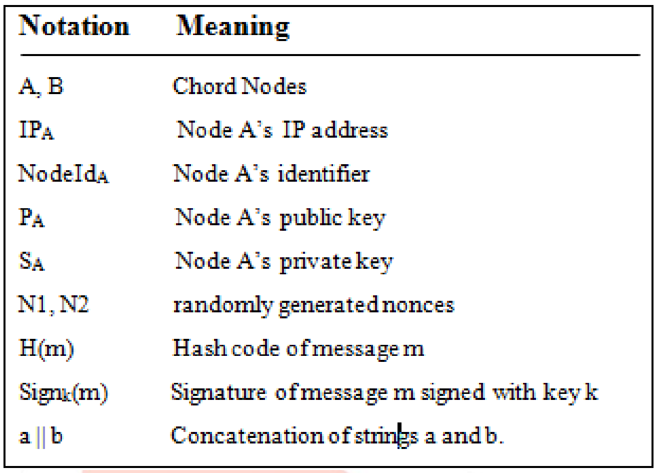
Securing DHTs has always been a challenging task [2,3,4]. Douceur [6] introduces Sybil attack as peer can obtain multiple Sybil identities simultaneously in P2P networks. He proves that without centralize certification authority it is hard to prevent nodes from obtaining pseudo-identities. He also shows that requiring all nodes to obtain certificate is too hard to be practical in P2P networks. Existing solutions to this attack requires participants to obtain certificates [2] or perform some computational puzzles [7]. This compromises the decentralize nature of P2P systems and does not fully solve the issue of Sybil attack.

To address this issue, recent researches propose to use social network trust relationship for routing [8, 9, 10]. These systems require large number of repeated lookup to maintain state. So these systems are suffering from high overhead. Also these systems require peers to reveal social contact information as some of these systems require global distribution of friend lists. This is not fortunate, because social contacts are considered to be private information. Another security concern in these systems is peers need to directly communicate with random peers revealing their IP address which allows attacker to perform traffic analysis and compromise user privacy [11]. However, some mechanisms protect privacy of social relationship by ensuring that the peer's relationship information is revealed only to the peer's immediate friends [12]. But this mechanism does not defend against attacker who targets a specific subset of users. As well as experiences higher latencies compared to traditional systems.

4 Proposed Mechanism

4.1 Assignment of Verifiable NodeId

To prevent the malicious nodes to generate multiple NodeIds, we need to restrict NodeId generation by binding some identity of node to NodeId. Binding IP address to NodeId is common solution which cannot work in NAT(Network Address Translation) environment because nodes behind the NAT box appear to have same IP address. Therefore we used public key cryptosystem and bind public key and IP address to NodeId. So that nodes behind the NAT box and having same IP address but different public key can also generate different NodeId. In existing Chord system NodeId is generated by hashing the IP address. This cannot work in verification session to ensure the integrity of message exchanged with signature techniques.



Notation	Meaning
A, B	Chord Nodes
IP_A	Node A's IP address
$NodeId_A$	Node A's identifier
P_A	Node A's public key
S_A	Node A's private key
N1, N2	randomly generated nonces
H(m)	Hash code of message m
$Sign_k(m)$	Signature of message m signed with key k
a || b	Concatenation of strings a and b.

5

Figure 3 Notations used throughout this paper

In our mechanism node will generate its NodeId as follows:

$$NodeId_A = H(IP_A \,||\, H(P_A))$$

NodeId generated this way will call the correct NodeId.

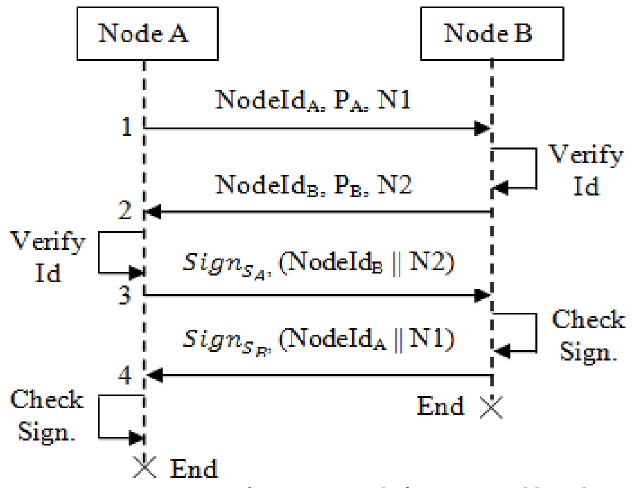


Figure 4 Verification session before routing table updating

In structured peer-to-peer networks, random distribution of NodeId is the essential property that must be maintained. The nested call of hash function in our mechanism helps to maintain this property. In this method every node can verify whether node A generated its NodeId in correct way, by computing hash code with A's IP address and public key. If $NodeId_A$ is generated correct way, then node A will treated as a valid node otherwise it will be ignored.

5.1 Verification Before Routing Table Update

A Chord node updates its routing table when receiving unwanted messages like lookup request and find node calls. So it is very easy for adversarial node to inject bogus NodeId into genuine node's routing table. Existing Chord protocol does not possess such mechanism to restrict this misbehavior. In order to circumvent the routing table poisoning, we suggest that when node A exchanges messages with node B and want to add B's information into its routing table, node A must verify whether node B is a valid node which generates its NodeId in a correct way. Routing table must be updated in a verifiable way as shown in figure 4.

Figure 4 shows the message exchanges between two nodes during verification session. In step 1 and 2 recipients verifies the legality of the sender's NodeId by calculating the hash code from node's IP address and public key. This will ensure that NodeIds are generated in a correct way and whether node is valid or not. In step 3 and 4 recipient verifies the signature, NodeId and equality between nonce. Verification of signature and NodeId ensures that the sender is same entity throughout the session and recipient is the correct entity to receive messages. Checking the equality of nonce prevents from replay attacks. If session ends successfully, node A and node B will insert each other's information into their routing tables; otherwise messages are discarded.

6 Simulation And Analysis

Haiman Lin and Ruilin Ma proved that the longer the poisoning duration, the higher the poisoning effect. Without RTP(Routing Table Poisoning), the poisoning effect is almost linear. But whenever the RTP attack is performed, it increases the impact of the attack and poisoning level increases very effectively. The experiments performed on existing mechanism are also performed on proposed mechanism. The results are collected and taken under examination.

Figure 4 shows the comparison of impacts of attack on both existing and proposed mechanism. In this case 3 hrs of average node interval is taken and without RTP, malicious routing table entries are counted. As 10% nodes are Sybil nodes in simulation, 10% malicious routing table entries are obvious. It is clearly shown that in proposed mechanism, the routing table entries are prevented as compared to existing mechanism. Figure 5 also shows the comparison results between two mechanisms. In this case RTP is launched and 3 hrs of average node interval is taken.

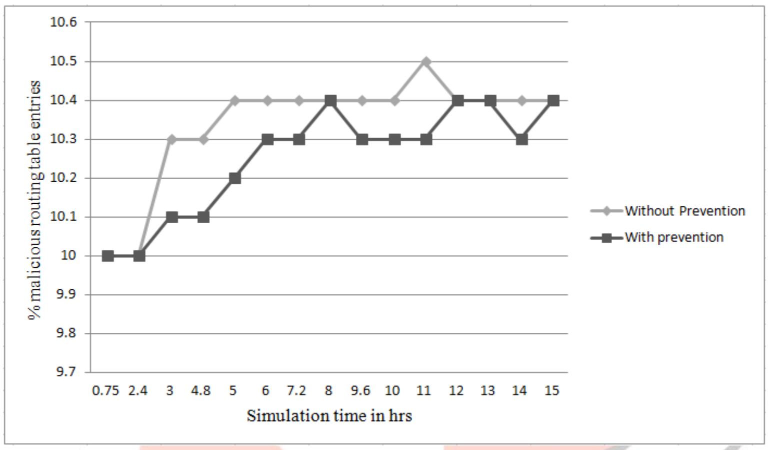


Figure 4: 3 hrs average node interval without RTP

Although average node interval is same as in above case, the malicious routing table entries are increased as the simulation time increase. In this case, RTP increase the impact of Sybil attack which leads to poison almost half of the network with malicious routing table entries. In existing mechanism, 50% network is poisoned whereas in our proposed mechanism, only 15% network is poisoned. So it is clear that only 5% routing table entries are done by RTP attack in our proposed mechanism.

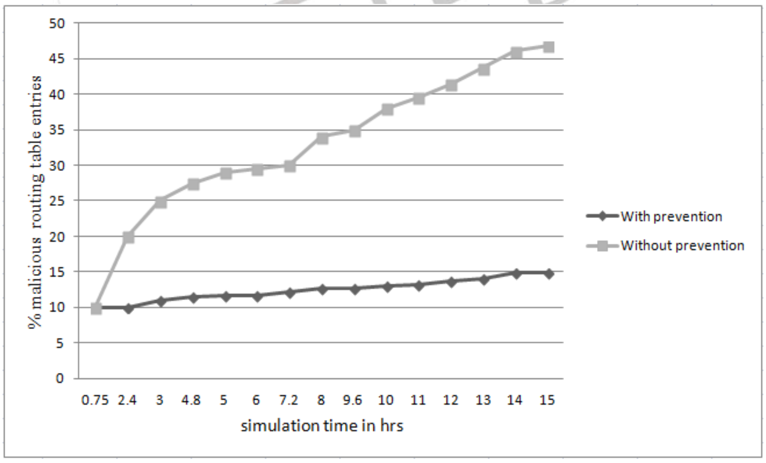


Figure 5: 3 hrs average node interval with RTP.

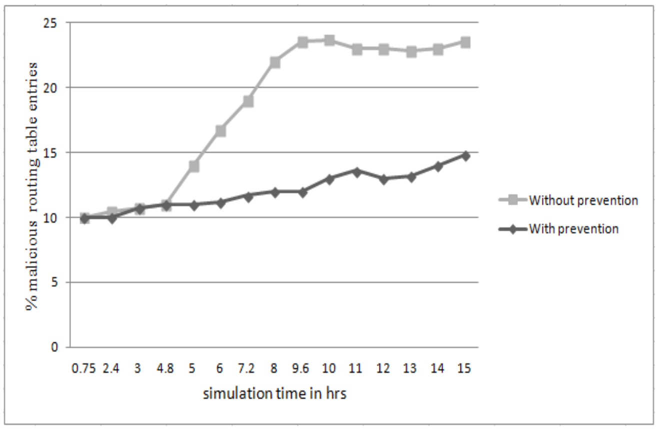


Figure 6: 15 hrs average node interval without RTP.

Figure 6 illustrates the graph of the malicious routing table entry with respect to simulation time in hours. In this case 15 hrs of average node interval is taken without RTP attack. Average node interval is an important parameter which directly affect on number of malicious routing table entries. So here also the malicious routing table entries are increase as the simulation time increases. But in proposed mechanism entries are less nearly about 15%. In this case also our proposed mechanism works effectively and decreases the impact of the RTP attack.

In last case with average node interval of 7 and using RTP attack, simulation is carried out and results are measured. Figure 16 shows the analysis of fourth experiment which shows the percentage of malicious routing table entries with respect to simulation time in hours. In this case, more than 50% of network is poisoned with malicious routing table entries in traditional mechanism. However, in our proposed mechanism, this effect is decreased to about 20%. In this case also our mechanism works effectively.

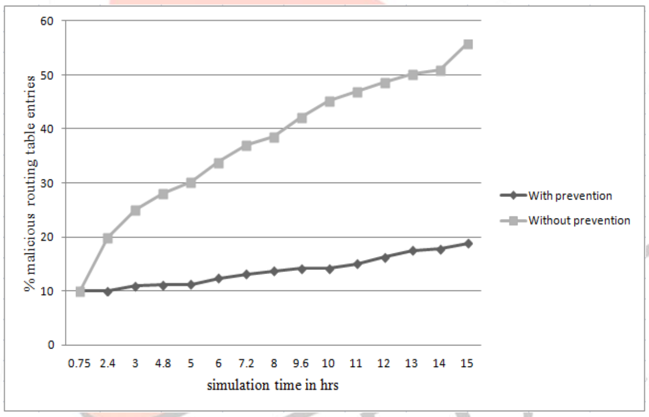


Figure 7: 15 hrs average node interval with RTP

7 Conclusion and Future Work

As shown in simulation analysis, this mechanism is effectively prevents the malicious routing table entries. It decreases the effect of RTP attack from 55% to 20%. As 10% nodes are malicious, 10% malicious entries are obvious. So it is clear that 10% malicious entries are still not prevented. Verifiable NodeId assignment and routing table updating ensure that each Chord node has unpolluted routing table, in which all fingers have verifiable NodeIds which is associated with their public key and IP address. But there is still considerable room for an adversary to pollute the genuine nodes routing tables. To inject itself into the routing tables of other nodes, the malicious node appears as genuine node when generating its NodeId and make contacts to other nodes. But when receiving lookup requests, it forwards this requests to faulty or non-existent nodes.

As a future work, we plan to introduce the secure message forwarding mechanism and distributed way of generating public keys.

REFERENCES

[1] Ion Stoica, David Karger, Robert Morris, Hari Balakrishnan, M. Frans Kaashoek, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," MIT Laboratory for Computer Science, 2011.
[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI*, 2002.
[3] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *IPTPS*, 2002.
[4] D. Wallach. A survey of peer-to-peer security issues In *International Symposium on Software Security*, Tokyo, Japan, 2002.
[5] Jochen Dinger and Hartentein. Defending the Sybil Attack in P2P networks: Taxonomy, Challenges, and a Proposal for self-Registration. *IEEE, proceedings of the First international Conference on Availability, Reliability and Security, 2002*.

[6] J.Douceur. The Sybil Attack. *Proceedings of the First International Workshop on Peer-to-peer Systems*. Springer, March 2002.

[7] N. Borisov. Computational puzzles as sybil defenses. In *IEEE P2P*, 2006.

[8] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil nodes using social networks. In *NDSS*, 2009.

[9] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. In *IEEE Security and Privacy*, 2008.

[10] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *SIGCOMM*, 2006.

[11] K. Bauer, D. Mccoy, D. Grunwald, and D. Sicker. Bitstalker: Accurately and efficiently monitoring bittorent traffic. In *Proceedings of the International Workshop on Information Forensics and Security*, 2009.

[12] Prateek Mittal, Matthew Caesar, Nikita Borisov. X-Vine: Secure and Pseudonymous Routing in DHTs using Social Networks, In *ACM 2012*.

[13] Haiman Lin, Ruilin Ma, Li Guo, Peng Zhang, Xiaolun Chen. Conducting Routing Table Poisoning Attack in DHT Networks. In 2010 IEEE.

AUTHOR

[1] Avinash Chaudhari received the degree of B.E in Information Technology from Vishwakarma Govt. Engg. College Chandkheda, Gandhinagar(GU) in 2011. During 2011-2012, he worked in Creative Glance Technologies as iOS Developer. He is currently pursuing M.E from L.D College of engineering, Gujarat Technological University.

[2] Pradeep Gamit received the degree of B.E Computer Engineering from Vishwakarma Govt. Engg. College Chandkheda, Gandhinagar (GU) in 2010 and degree of M.E. from L.D. College of Engg.(GTU) in 2012. He is currently working as Assistant professor in L.D College of Engg.