International Journal of Scientific and Innovative Research 2014; 2(1): 136-141,
P-ISSN 2347-2189, E- ISSN 2347-4971

A DHT ORIENTED PEER TO PEER NETWORK WITH NEW HASH FUNCTION

***VIVEK SAINI[1], K P YADAV[2]**

[1]Research Scholar, Department of Computer Science & Engineering, Sai Nath University, Ranchi, India

[2] Director, MIET, Greater Noida , Uttar Pradesh, India

*Address for correspondence : Vivek Saini, Research Scholar, Department of Computer Science & Engineering, Sai Nath University, Ranchi, India

ABSTRACT

Now a days P2P networks are widely used for voice and video communications and also in many transactions like file sharing. In P2P networks DHT (Distributed Hash Table) oriented routing protocols gives an efficient way to search the contents (like files) between various peers. A DHT protocol works on a hash function for the P2P application that provides a key and searches the responsible nodes for the particular key. In this paper the core focus is targeted to identify a new hash function that can enhance the working of DHT by generating a new identifier for p2p nodes. The Chord protocol is also chosen as the routing protocol for the numerous reasons those are conferred in this paper.

Keywords: P2P *Networks, Distributed Hash Tables, Hash Functions ,Cryptography, Key Distribution Center*

INTRODUCTION

To find the resources rapidly in decentralized distributed systems, DHT protocols are very suitable for searching the contents efficiently. In a distributed system, resources like any type of file, several messages and directories, or various contents can be backed up or fetch from the any nodes at anytime. DHT has a collection of joining nodes, wherever every node have small information of other nodes to make the difference to each other in the system and also has a route searching requests to identify a best path to move towards an appropriate target[1].

Message integrity is among the major requirements in many network protocols that we use today. As, today transmission on any network is achieved at very high speed, the processing for encryption, authentication and integration must also be done at very high speed. Currently, many hash functions are being used for this purpose, say- MD4[2], MD5 [3] and SHA-1[4], Tiger, Whirlpool etc.. These hash functions are one-way hash functions, in the sense that original input message can not be obtained back from the hash value generated from it. In general, Hash Function is defined as a function that takes arbitrary length input and produces output of fixed length, which is known as hash value or the message digest [5]. For a hash function to be useful for network security, it is desirable to have no two different input messages for which the digest is same.

Almost all major cryptographic protocols depend on the security of hash functions [3]. Although, there is a variety of hash functions available in the market, only MD5 [4] and SHA-1 [5] are widely used all over the world. Both hash functions are derived from MD4 [6], which has been proven to be weak. It therefore says that all functions based on MD4 may have common weaknesses. In P2P network every resource is associated with a key, with the help of that key, DHT can find the responsible node for the linked resource rapidly usually in O (log n) hops, here n is the quantity of nodes in the peer to peer network.

There are various DHT protocols like CAN[8], Chord[9], and Pastry[10] but in this paper Chord is targeted protocol on which new hashing will be applied.

DESIGN CONSIDERATIONS FOR THE NEW HASH FUNCTION

For message integrity, the hash function should be fast and one-way functions, that is, it should be practically impossible to find the input message from the digest. This feature gives security against data modification by adversary, but, message integrity alone does not guarantee sufficient security against few of the attacks [7]. For example, general error detecting codes are not sufficient enough because, if the adversary knows algorithm for generating the code, he can easily generate the correct hash code again after forging the message. Intentional modification is undetectable with such codes. i. e. suppose a sender sends the message M along with its hash value h, and intruder changes this message M into M', he also intercepts h and calculates h' for new message M', and transmits it to the receiver. When receiver will recalculate hash on M' it will result in verified one, which is not true. However, encryption techniques, that use a key, can be used to produce a cryptographic checksum that can protect against both accidental and intentional modification in message, and unauthorized data modification also. A hashing scheme can be made more secured and strong by combining it with a block cipher encryption algorithm either symmetric or asymmetric. The function h is defined such that h(M) can be calculated from the message M easily, but if only h(M) is known, finding even one message M that will generate this value is "difficult". Moreover, calculating another message M' that produces the same hash value, i.e., h(M) = h(M'), must be infeasible. The hash value may then be given to encryption function, whose key is known to sender for final calculations of keyed hash. The corresponding key will be used by the receiver to invert the transformation and restore the value h(M). At the receiving end, the function his applied to the received message M, and the two values of h(M) are compared. The message is considered original and unaltered if the two values are equal.

Designing and implementing a new keyed hash function includes two constructs- a compression function that operates on input strings of a fixed length and then to use the cascade function to extend the compression function to string of arbitrary length[10].

P2P ROUTING ALGORITHM (CHORD)

Each resource has a key linked with it. Given a key, a DHT can quickly locate the node responsible for the associated resource, typically within O (log n) hops, where n is the total quantity of nodes in the system.

In Chord, both nodes and keys are given numerical identifiers. The identifier for a key is obtained by hashing that key with some hash function that is used by all of the nodes in the system which returns integers of some bit length m. A node is assigned an identifier by hashing its IP address. Nodes and keys are then arranged in an identifier ring modulo 2m. Each key's value is stored on the first node with an identifier equal to or following that key's identifier in the ring [8].

This aspect is illustrated in following Figure 1.In the Chord ring; the hash bit length m is 6. There are 10 nodes in the network (shown with N prefixes followed by the nodes identifier) and 5 keys (shown with K prefixes followed by the keys identifier) are being stored. Each key is shown being stored on the first node that succeeds the keys identifier in the ring, as indicated by the arrows. In order to find nodes that are responsible for keys, each node has to store some routing information.In Chord, this routing table is called a "finger table". The Chord finger table for a node with identifier id contains m entries, numbered from 0 to m-1. For finger table entry i, the node stored in that entry is the first node whose identifier succeeds id + 2i (mod 2k). It is possible (and often probable) to have duplicate entries in the finger table.

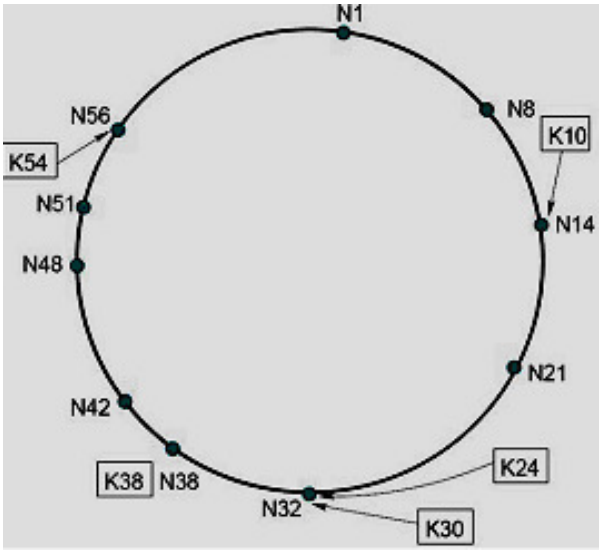

Figure 1: The Chord Ring

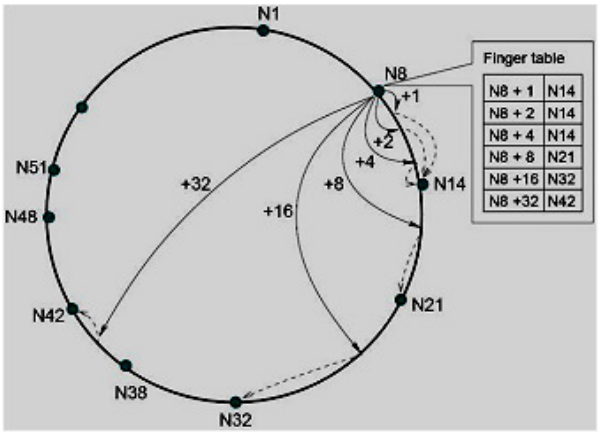

Figure 2: The Chord Ring with Finger Table

Figure 2 shows a sample finger table with an illustration of how the finger table is derived for node N8. N8's last finger table entry should be the node that succeeds 8+25. This node is N42, so a reference to N42 is stored in the last finger table entry of N8's finger table. The rest of the finger table entries are filled in with the same process for i =0, 1, 2, 3, and 4. As figure 2 illustrates, each node only has information about a subset of the nodes in the overall system. As the system gets much larger, the number of unique nodes in each node's finger table becomes a smaller fraction of the overall number of nodes. The size of the finger table has been shown by [8] to be O (log n) where n is the number of nodes in the system. The advantage of the finger table is that when performing a lookup the request can jump about half of the remaining distance between the node doing the routing and the node responsible for the key. This divide and conquer approach to routing lookup requests has been shown by [8] to use O (log n) hops for each route. The algorithm for routing a lookup request from a node is simple: forward the request to the last finger table entry that precedes the identifier of the key. The node preceding the destination node will detect that the key falls between itself and its successor and return information about its successor to the node performing the lookup.

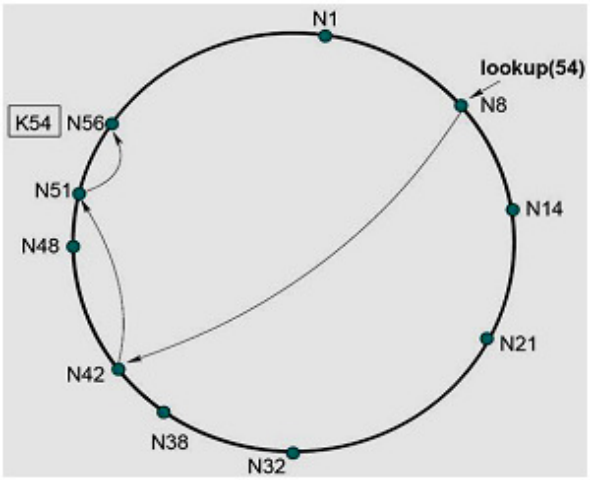

Figure 3: An example of route taken by a lookup in a Chord network [8]

Figure 3 shows an example of the route a lookup request might take through a Chord network. In this figure, N8 is performing a lookup request for key K54.

CURRENT SCENARIO FOR HASH FUNCTION WITH KEY COMBINATION

We assume that two parties A and B wish to communicate with each other, and for message integrity they may use any available hash function such as SHA family or MD family. At the time of session establishment, a user A, may generate random 512-bit value, HSH, A then sends this HSH to the second party in communication, B, in secret manner. This secret key may be further used for keyed Hash Function in any of the following two ways [11]:

Similar problems arise with suffix technique. Still, secret suffix is secure with respect to padding attacks. This is because a message digest is computed with a secret suffix as the last input block. Without knowing the secret, the

adversary cannot, with any surety, append (or prepend) to the message. One major disadvantage of the secret suffix is that it is prone to Birthday attack. In brief, this attack consists of the intruder generating a message trial pool of size R and recording a sample pool of genuine messages of size S. The probability of at least one message in a trial pool hashing to the same MD5 value as one of the messages in the sample pool is roughly expressed by P = (R * S/N) where N is the range (N = 2128 in our case) . According to the Birthday paradox, there is a 50% chance of two messages hashing to the same MD5 value for R = 264 and S = I.

a) **Secret Prefix Method**- When A needs to send a message M to receiver B, A prefixes the random 512 bit value HSH to message and calculates the digest for this combination. It can be represented as h = MD 5(HSH||M) and sends this h to B. Since B possess HSH, it can re- compute MD5 (HSH||M) and verify h. This is known as prefix technique.

b) **Secret Suffix Method**- When A needs to send a message M to receiver B, A postfixes 512 bit value HSH to message and calculates the digest for this combination. It can be represented as h= MD5(M||HSH) and sends this h to B. Since B possess HSH, it can re-compute MD5(M||HSH) and verify h. This is known as suffix technique.

PROPOSED KEYED HASH SOLUTION

There could be numerous solution to the above scenario and thus to strengthen previous designs. Few of such solutions may be to increase the number of rounds (as in MD5); add some coding or scrambling steps (as in SHA-1); increase the buffer size and make the mixing step vary with each round. An example of such an assumption is any ideal-cipher model that uses a key.

An adversary who wishes to discover the secret prefix would first record a message M accompanied by its integrity value, MD5(HSH||M). He would then need to try on the average 2128 possibilities before discovering a prefix string S where S = HSH.

But these methods are not resistant to attacks. To possess the goal of composing original messages, the adversary needs only the intermediate hash value of HSH, i .e. MD5(HSH). If, the MD5(HSH) is 128 bits long, the number of operations required to find MD5(HSH) is computationally 2128.

One more drawback of the secret prefix method is its vulnerability to padding attacks. A padding attack is successful when the adversary is able to either prepend or append few extra bits to an authentic message and pass the resulting message off as original one. An adversary can capture a message M along with its secret prefix-based check MD5(HSH||M). He can then append arbitrary data, M, to the end of M and compute the digest of the resulting message, using MD5(HSH||M) as the initialization value. The legitimate receiver is then fooled into accepting the fraudulent message because the accompanying secret prefix value is "correct". [12]

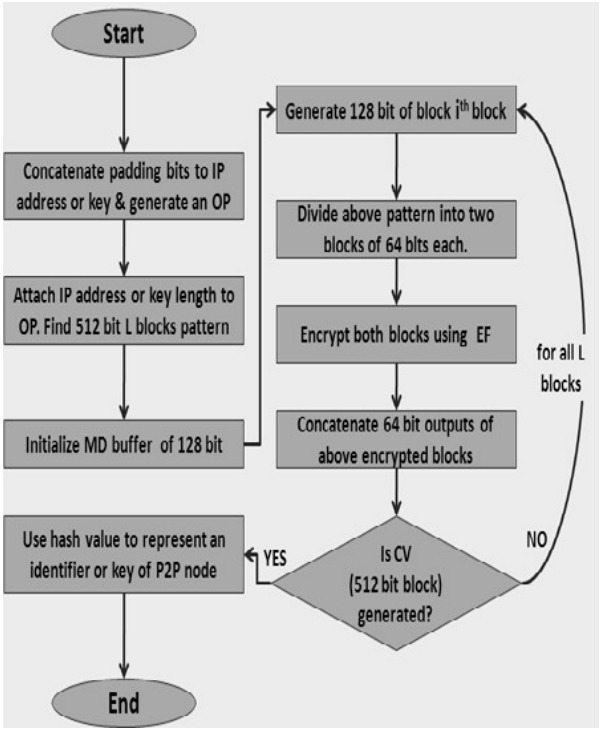

Figure 4: Proposed Hash function algorithm for identifier of resource key generation.

Building hash functions based on block ciphers is the most popular and established approach till date. In this approach, the hash

compression function is a block-cipher with its two inputs representing a message block and a key. At present, a protocol requiring 2128 operations to defeat is considered strong and secure. Nevertheless, it is conceivable that a need for stronger protocols may arise in the future. This paper proposes integration of a keyed encryption function such as DES along with hash function such as MD5. Because the keyed encryption function such as- DES works on Symmetric Key Cryptography, both sender and receiver work on the same key. This common key is shared between them using an encrypted link between them by Key Distribution Center (KDC) [11]. It is responsibility of KDC to send the common session key to both parties involved in communication, using their corresponding public keys. As, only these parties possess their private keys, so other user in the network may decode and use this session key. No other user, apart from KDC and both parties, involved in the message transmission, has any idea of the shared secret key. Thus, it may validate source identity, as receiver is now having the key that same as that of sender's key.

In this solution, the working of hash function HF is integrated with encryption function EF. The output of hash operation in each block will be used as input for encryption function. The hash function HF gives output of 128 bits and the encryption function EF accepts input block of 64 bits at a time. So, first, the output of HF is divided into two blocks each of 64 bit long, first with left 64 bits and second with right 64 bits. Then apply EF on both blocks respectively. The output will be again of 64 bit (total of 128 bit). This overall 128 bit output will then be used as 128 bit CVq for HF processing of next block of input [14].

CV0=IV

CVq= E (K,B1) || E (K,B2) Where,

IV= Initialization value of 128 bit buffer E= Encryption function EF

B1= Left 64 bits of output of HF digest value B2= Right 64 bits of output of HF digest value K= EF key.

The proposed algorithm may be stated in figure 4.

The overall processing with this scheme is shown in the Figure 5.

The strength of this variant is difficult to estimate. The only observation that can be made with certainty is that it is stronger than using only MD5, MD4 or SHA family hash function. The brute force attack on this method requires, on the average, 2196 trials. (Assuming that the intruder has perform attack that equivalent to both attack on HF plus attack on EF at the same time).

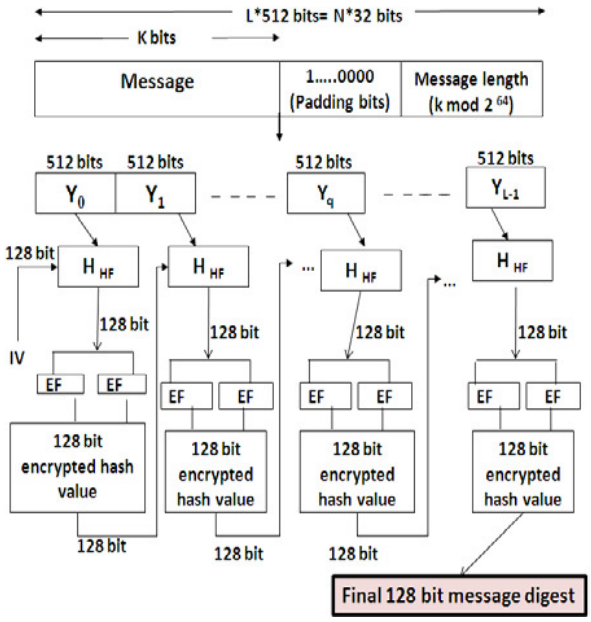

Figure 5: Processing of Hash Function HF having Encryption EF in between

CONCLUSION AND FUTURE SCOPE

DHT protocols are available for routing the packets in P2P networks. Careful and secure identification of a machine in p2p network is very challenging in such type of distributed network. Security may be provided using message integration and authentication techniques between peers. The paper proposed involvement of a newly designed hash technique that makes use of a key while hashing. This key is the main reason behind authenticity of message between peer entities.

REFERENCES

1. A. Saroliya, U Mishra, A. Rana. "A pragmatic analysis of peer to peer networks and protocols for security and confidentiality", IJCCR. Volume 2 (6), November 2012.

2. R.L. Rivest. MD4 Message Digest Algorithm. RFC 1186, October 1990.
3. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.
4. National Institute of Standards and Technology, U.S. Department of Commerce. Secure Hash Standard, 2002. FIPS PUB 180-2.
5. P. Rogaway, T. Shrimpton. Cryptographic Hash - Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. Springer-Verlag 2004.
6. M. Stanek. Analysis of Fast Block Cipher Based Hash Function. Computational Science and its Applications. 2006, vol- 3982/2006, pp- 426-435. DOI: 10.1007/11751595_46.
7. J. Walker, M. Kounavis, S.Gueron, G.Graunke, Recent Contribution to Cryptographic Hash Function, Intel Technology Journal, 2009, Vol 13, issue 2, pp 80-95.
8. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, "A Scalable Content Addressable Network", In Proceeding of SIGCOMM'01, ISSN:0146-4833, Volume 31, Number 4, San Diego, California, USA, pp. 161–172, August 27-31, 2001
9. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications", In: Proc. ACM SIGCOMM'01, San Diego, California 2001.
10. R. Antony, D. Peter, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems", In Proc. of IFIP/ACM International Conference on Distributed Systems Platforms, ISBN:3-540-42800-3, Heidelberg, Germany, pp. 329–350, Nov. 2001
11. L. Harn and C. Lin, Authenticated Group Key Transfer Protocol Based on Secret Sharing. Computers, IEEE Transactions on, June 2010, vol 59, issue- 3, pp-842-846.
12. G. Tsudik, Message Authentication with One-Way Hash Fnctions, Newsletter, ACM SIGCOMM Computer Communication, Volume 22, Issue 5, Oct.1992 pages 29-38.
13. B. Preneel, P.C. Van Oorschot, "On the security of iterated Message Authentication Codes," IEEE Transactions on Information Theory, Vol. 45, No. 1, January 1999, pp. 188-199.
14. R. Purohit, U Mishra, A. Bansal. "Design and Analysis of a New Hash Algorithm with Key Integration". International Journal of Computer Application. Vol 81 (1), pp 33-38, 2013.