Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090134

Entity Authentication Protocol for P2P Cloud Storage based on Ternary Huffman Merkle Hash Tree

*Ningning Song**, *Xianwei Zhou and Qian Liu*

School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing 100083, China

Received: 3 Apr. 2014, Revised: 4 Jul. 2014, Accepted: 5 Jul. 2014
Published online: 1 Jan. 2015

Abstract: With the development and application of cloud computing, cloud storage is becoming more and more popular with many organizations, but traditional principal-minor structure for cloud storage is at risk of single point of failure. Moreover, it has brought safety risk. Cloud storage security has become the key problem of restricting the development of cloud computing. The paper has a deep research of cloud storage structure and cloud storage security, proposes a entity authentication protocol for P2P cloud storage based on ternary Huffman Merkle hash tree (HuffMHT).This method using the concept of ternary HuffMHT can obtain an effective safe strategy. At the same time, symmetrical key algorithm and public key algorithm are just combined to reduce the authentication delay effectively and increase the network lifetime and enhances the security of the networks. Moreover, signatures of knowledge is used in this paper, which notes number of hash is reduced, the power which notes consume is debased, the network load is decreased. The performance of the system is discussed, and the encryption method has high efficiency and high security.

Keywords: P2P cloud storage, Network security, Entity authentication, Ternary HuffMHT

1 Introduction

With the development of information technology, information data is increasing by blasting. The most challenge we are facing now is how to realize the security of Big Data. Cloud Storage is extension and development of Cloud Computing, which has economy, high reliability and scalability. Appearance of Cloud Storage makes users be easily satisfied without affording the increasing high cost in data center management [1]. Now, there are many Cloud Storage systems, such as Google GFS, Amazon EC2, HDFS, etc. All of them are based on Master-Slave Structure, which causes a range of problems [2]: (1) Cloud Storage Provider has absolute control over users' data, users are apprehensive for the security of data. (2) Single Point of Failure is likely to occur within the Master-Slave Structure. (3) As more users join in cloud storage system Master-Slave Structure has constraints based on bandwidth. P2P is *a* non-centralized architecture computing model. Each node in P2P network can both provides services and be serviced, all nodes have equal status. Thus, the structure of a peer-to-peer is the progress of Cloud Storage structure. DHT technology is one of the main realization methods of P2P at present. DHT provides precise matching of information, but it can't support content or semantic meaning or other complex search, which brings great inconvenience to users' information. So far, data security has always been the most troublesome problem in promotion of cloud storage, such as data storage security, transmission security, access control and so on [3].

Currently researchers have made ??a number of cloud storage entity authentication schemes: authentication mechanism based on trust management [4,6], authentication mechanism based on public key certificate [5,8]. Authentication scheme based on trust mechanism mainly based on own experience to judge node and maintain the safe operation of the network. Authentication mechanism based on public key certificate is not suitable for large-scale network environment. Therefore, existing authentication scheme and cannot meet the special circumstances of the cloud storage network. Literature [9] gives the Merkle hash tree based authentication protocol entity, the calculation procedure to perform a complete binary tree. However, the actual calculation is only valid with the leaf nodes of a relationship. Literature [10] gives the hash tree based on Huffman entity authentication protocol, but it still exists the problem of high maintenance cost.

* Corresponding author e-mail: 6221294@163.com

In order to solve security threats in cloud storage, we must design efficient and flexible cloud storage authentication mechanism based on Semantic P2P structure. A entity authentication protocol for P2P cloud storage based on ternary Huffman Merkle hash tree (HuffMHT) is proposed. This method using the concept of ternary HuffMHT can obtain an effective safe strategy. At the same time, symmetrical key algorithm and public key algorithm are just combined to reduce the authentication delay effectively and increase the network lifetime and enhances the security of the networks. Moreover, signatures of knowledge is used in this paper, which notes number of hash is reduced, the power which notes consume is debased, the network load is decreased.

2 System model

We constructed a prototype system named SP2CPS based on Semantic P2P structure, and it can be divided into five layers: physical layer, semantic layer, router layer, management layer and application layer, seen in the figure 1.

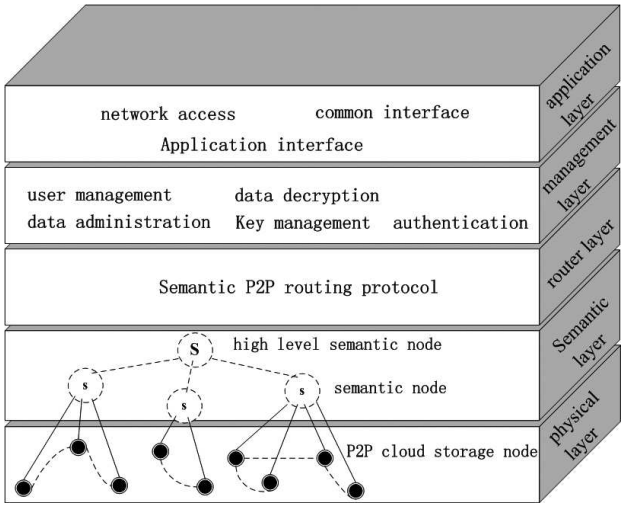


Fig. 1: Semantic P2P structure.

(1) Physical layer
Physical layer is composed of P2P cloud storage nodes which can provide idle storage resources and computing resource in a certain area. P2P cloud storage nodes provide and demand service simultaneously, they have the same status. P2P cloud storage nodes are the basic elements of the system.

(2) Semantic layer
P2P cloud storage nodes with similar in semantic, connecting each other, compose diffident semantic cluster. Each cluster has a semantic node to take charge of the organization and administration of the cluster. The semantic node just needs to management those nodes that register?in?cluster. Choose nodes with stronger storage capacity, high - speed calculation and long online time to be the semantic node. Lower semantic nodes can compose high semantic cluster though semantic classification.

(3) Router Layer
Router layer is Responsible for nodes for the storage node interoperability, semantic layer topology construction, dynamic monitoring topology, routing conference and other, which uses the semantic-based routing algorithm to put P2P nodes together to form a structured semantic peer networks according to the semantics.

(4) Management Layer
Management layer is responsible for user management, data storage management, create file directory system for each user, use a copy of redundancy, error correction redundancy to ensure data reliability. Encryption, authentication, key management and other security mechanisms are used to ensure data security.

(5) Application Layer
Application layer provides file storage service interface to user, develops different application service interface according to the type of application to provide various application services.

Without loss of generality, in this paper, we assume that the nodes are divided into three layers. The first layer consists of physical layer peer nodes, the second layer is formed by semantic nodes, and the high-level semantic nodes compose the third layer. There are m clusters in the system, every cluster $C_i(i = 1,2,3,\ldots,m)$ has a unique identifier CID_i, we denote by sn_i cluster head within the cluster $C_i(i = 1,2,3,\ldots,m)$, that is, the child semantic cluster node

3 Ternary Huffman Merkle Hash Tree based on access probability

In the peer cloud storage system, peer node's resource and calculation ability is not balanced, so those nodes have more resource and strong computing ability are more likely to be accessed. Some Existing multi- point authentication protocol use the complete binary tree to build merkel hash tree, the average hash path length is too long. Huffman Merkel hash tree use binary Huffman tree to build hash tree, this unbalanced binary tree will drop average hash path length to a certain extent, but for the bottom of the leaf node hash path length issues remain unresolved.

Definition 1. Access probability is probability of being accessed by another node

When we build ternary Huffman merkel hash tree, the node with small access probability is put in a leaf close to the root, and we put the node with high access probability far away from the root. This has the advantage that, depending on the request, the system needs to build

different Huffman key hash tree, in the build process, the node with high access probability appears in almost every Huffman merkel hash tree. Since the hash tree root hash value is calculated from the bottom up, so we put the node with high access probability far away from the root can make the original tree usability enhancements when some nodes join in or left the hash tree.

We assume that there are peer nodes in cluster $C_i(i = 1,2,3,\ldots,m)$, we denote by p_j access probability of node $x_j(j = 1,2,3,\ldots,n)$. A ternary Huffman merkle hash tree based on access probability is constructed with the following rules:

(1) $F = p_1, p_2, \ldots, p_n$ is used to represent a set that has n ternary trees (there is only one root node in each tree);
(2) Three trees with highest access probability are selected to compose a new tree. According to descending order of probability, they are as the new tree's left sub-tree, middle sub-tree and right sub-tree (when two root nodes with equal access probability, numbers of tree are our considerations), and new access probability of the new tree is the sum of three old trees.
(3) Remove those three trees mentioned?above from the set $F = p_1, p_2, \ldots, p_n$, besides, put the new tree in the set.
(4) Process (2) and (3) are repeated until there is only one tree in set $F = p_1, p_2, \ldots, p_n$, the last tree (Ternary Huffman Tree) is result that we want.
(5) The definition of Ternary Huffman Tree indicate that the number of leaf nodes in a Ternary Huffman Tree must satisfies condition $num = 2t + 1$, therefore, when do not satisfy that condition, we can add a virtual node with access probability 0 in set $F = p_1, p_2, \ldots, p_n$ in order to build Ternary Huffman Tree.

Ternary Huffman Merkle Hash Tree comprises Huffman and one-way hash function. A sample Ternary Huffman Merkle Hash Tree is shown in Figure 2. To build the tree, a set of three adjacent nodes at a given level $t = \{1,2,3,4\}$, are combined into one node in the upper level, node that we denote by $h_{1-(2t-1)}$. Then, $h_{1-(2t-1)}$ is obtained by applying hash to the concatenation of the three cryptographic variables. Here s_i is the leaf key stored by leaf node N_i. Then h_i is computed as $h_i = h(s_i) = g^{s_i}$. In this figure, $h_{1-3} = H(h_1 \parallel h_2 \parallel h_3)$, $h_{1-5} = H(h_{1-3} \parallel h_4 \parallel h_5)$, $h_{1-3} = H(h_1 \parallel h_2 \parallel h_3)$, $h_{1-7} = H(h_{1-5} \parallel h_6 \parallel h_7)$, denotes a hash function such as md5 or SHA.

4 Signatures of knowledge

Signature of knowledge [11] is that signer can prove that someone has knowledge of a secret with respect to public information noninteractively and he doesn't leak any secret message. Signatures of knowledge is defined as being, Alice $y = g^x modn$, here x is the private key of Alice, y is the public key of Alice, g and n are public

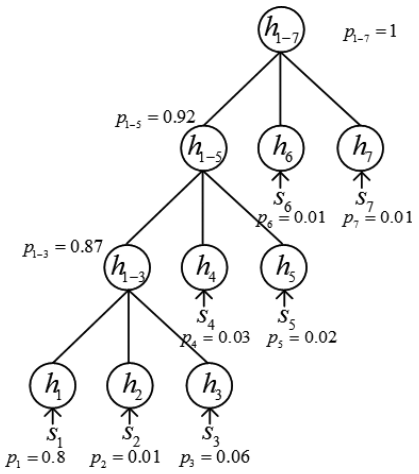


Fig. 2: Ternary Huffman Merkle Hash Tree.

system parameters, m is a message. Alice can prove to others that she has the private key x and she doesn't leak any secret message about x if she can provide a signature of knowledge (c,s) satisfy the following conditions.

$$c = H(m \parallel y \parallel g \parallel g^s \parallel y^c) \tag{1}$$

The process for signatures of knowledge as show below:

(1) Choose a random number r;
(2) Calculate according to the formula $c = H(m \parallel y \parallel g \parallel g \parallel y^r)$;
(3) Calculate according to the formula $s = r - c * x$;
(4) (c,s) is the signature of knowledge about private key x.

If you know the private key x, then perform the above four steps procedure you can easily calculate signature of knowledge (c,s). Otherwise, if you want to calculate (c,s) without the private key x, you must through the following three steps which is computationally infeasible. We can know that if you can give a signature (c,s) that satisfies condition $c = H(m \parallel y \parallel g \parallel g^s \parallel y^c)$, you are able to explain you own the private key x.

$$c = H(m \parallel y \parallel g \parallel g^r) \tag{2}$$
$$= H(m \parallel y \parallel g \parallel g^s g^{xc}) \tag{3}$$
$$= H(m \parallel y \parallel g \parallel g^s g^c) \tag{4}$$

5 A Signature of knowledge-based Multipart Authentication process

In the peer cloud storage system, peer nodes receive cloud storage services while providing cloud storage services to other peer nodes. The new node must enforce mutual authentication with peer nodes that has already in the cloud storage system. It is assumed that peer nodes in the network have already built a trust relationship with cluster head node (semantic node). There are several basic steps to achieve mutual authentication.

5.1 Network Initialization

The generation of cluster symmetric key is directly relevant to economic progress. Based on multi-Diffie-Hellman protocol, peer nodes exchange keying material $h_j = H(s_j) = g^{s_j} \ mod \ p$ with each other, in order to generate cluster key K_{C_i} for cluster C_i. According to multi-Diffie-Hellman protocol, we know that $K_{C_i} = g^{\prod_{i=1}^{n} s_i} \bmod p$. K_{C_i} is used to encrypt information within cluster C_i.K_{C_i} will be updated on time, as well as some node enter or left cluster C_i.

V is certification data set, maintained by a network management center, $V = \{(ID_{c_i}, K^*_{c_i}, h_j, (c,s)) \ i = 1,2, \cdots, m\}$, here symmetric key K_{c_i} is generated by cluster head node and is used for authentication. We randomly select a peer node to be the authentication node x_j, the authentication key is generated from s_j. We denote by h_j the authentication key of the cluster, and $h_j = g^{s_j} \bmod p$, (c,s) is signature of knowledge about s_j, and satisfies the condition $c = H_k(s_j \parallel h_j \parallel g \parallel g^s g^c)$. $\parallel$ is message connect sign.

5.2 Authentication beginning

Node T sends a request packet $(semantic_T, K_T)$ to the network center via a secure channel when it apply to join the network, here is a symmetric key that is generated by node T, $semantic_T$ is the semantic feature of node T. Network management center inquire cluster ID from semantic node according to the semantic feature $semantic_T$. Moreover, Network management center get certification data set of that cluster, send replying package $\{ID_T, time, E_{K_T}(V),\ mac(K_T, ID_T \parallel V \parallel time)\}$ to node. Here is identity of node. time is time stamping that is used to against replay attack. $(mac(K_T, ID_T \parallel V \parallel time)$ is message authentication code that is used to assure the integrity and reliability of the packet.

Node T decrypts the packet using key K_T, obtaining the certification data set $V = \{ID_{c_i}, K^*_{c_i}, h_j, (c,s)\}$, and send a reply packet to network management center. Then, node sends a certification request $V = \{ID_T,\ time,\ request,\ mac(K^*_{c_i}, ID_T \parallel time \parallel resquest)\}$ packet to cluster head ID_{c_i}.

Cluster head ID_{c_i} receives request packet from node. Firstly, it tests the validity of the time stamping and message authentication code $mac(K_T, ID_T \parallel V \parallel time$ by key $K^*_{c_i}$. If test is successful, cluster head ID_{c_i} broadcast a certification request packet $\{ID_T, time, auth\_start,\ mac(K_{c_i}, ID_T \parallel time \parallel auth\_start)\}$ to all nodes in the cluster, and send reply packet $\{ID_T, time, E_{KT}(h_{1-n}),\ \{mac(K_T, ID_T \parallel h_{1-n} \parallel time)\}$ to node T, here h_{1-n} is root key of the cluster . Otherwise, cluster head discard the packet and refuse certification quest. The process of authentication process is shown in figure 3.

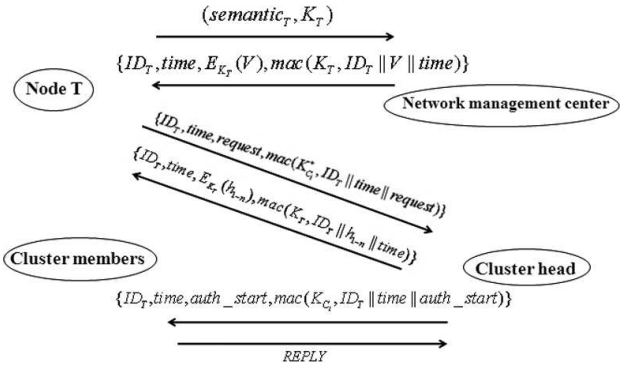


Fig. 3: Authentication Beginning.

5.3 Implementing Authentication

Node T receives request packet from cluster head ID_{c_i}. Firstly, it tests the validity of the time stamping time and message authentication code $mac(K_T, ID_T \parallel h1-n \parallel time)\}$. Then, Node T decrypts the packet using key K_T, obtaining root key h_{1-n}.

Except the node that own authentication key s_j, all others nodes send authentication packet $\{ID_k, time, E_{K_T}(h_k, (c,s)_k),\ mac(K_T, ID_T \parallel (h_k, (c,s)_k) \parallel time)\}$ to node T, here $h_k = g^{s_k} mod p$, $(c,s)_k$ is signature of knowledge about s_k.

Node T decrypts packets using key K_T, obtaining K_T authentication elements. Together with h_j that obtain from network management center, node T recalculates root key h^*_{1-n} according to the structure of Ternary Huffman Merkle Hash Tree. If h_{1-n} and h^*_{1-n} are equal, node T can confirm identity of cluster, then send confirmation packet $\{ID_T, time, E_{K_T}(h^*_{1-n},\ mac(K_T, ID_T \parallel h_{1-n} \parallel time)\}$ to node x_j. Otherwise, authentication fails, node T send alert packet to network management center because there may be some malicious nodes in the cluster.

Authentication node x_j verifies whether h^*_{1-n} and h_{1-n} are equal, if that's so, cluster c_i can confirm identity of node T, node x_j broadcast success packet in the cluster. Otherwise, authentication fails. Implementing authentication process is shown in figure 4.

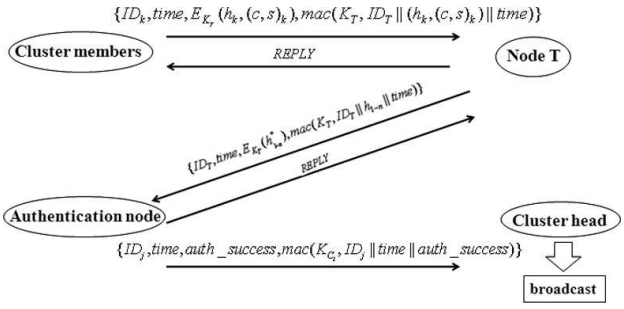


Fig. 4: Implementing Authentication.

The process described above is based on signatures of knowledge, peer nodes only need to seed hash value

about leaf key rather than the leaf key itself, and can prove to others that they have the leaf key s_j and she doesn't leak any secret message about s_j. When there is a new node need to authenticate again, those leaf keys are still safe enough, which can be used for the next authentication process.

6 Dynamic Update protocol of Ternary Huffman Merkle Hash Tree

Cluster nodes changes frequently according to factors such as access probability.

The existing multi-party authentication protocol based on hash tree need to recalculate hash values of leaf key in order to calculate the hash value of the whole tree(root), which result in heavy network load and computing load. In this paper, we design a high-effective dynamic update protocol of Ternary Huffman Merkle Hash Tree based on signatures of knowledge.

6.1 Renewal scheme when a node adds

Ternary Huffman Merkle Hash Tree t is shown in figure 5, we assume there are seven nodes in the cluster. When a new node joins in the cluster after authentication, it is denoted by x_8. Node generates a leaf key s_8, computing the hash value $h_8 = g^{s_8} mod\, p$. We might as well set the access probability of node $p_8 = 0$.

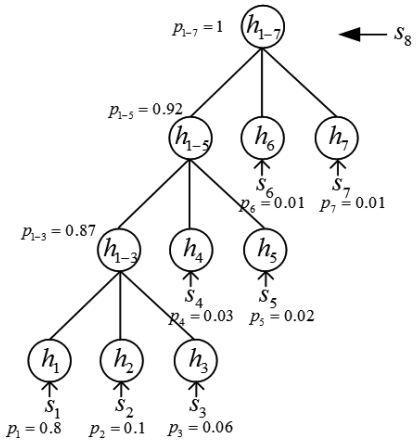


Fig. 5: New node adds.

According to the construction rules of Ternary Huffman Merkle Hash Tree, the cluster generates a new hash tree t^* as shown below after node s_8 joins in. Here $s_9{}^*$ and $h_9{}^*$ are virtual nodes, which are not used to calculate the hash value of the whole tree (root).

The hash value of t is $h_{1-7} = H(h_{1-5} \parallel h_6 \parallel h_7)$, after node s_8 joins in, the hash value of the new tree t^* becomes $h_{1-9} = H(h_{1-7} \parallel h_8)$.

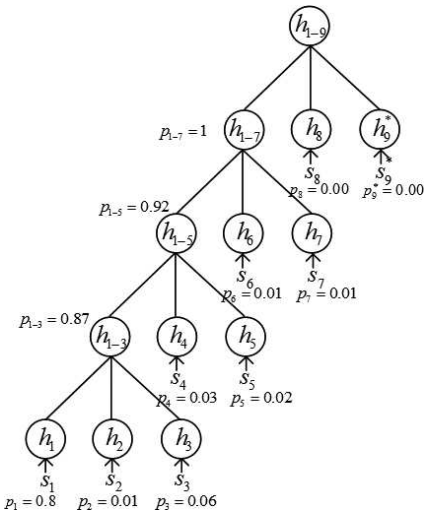


Fig. 6: Hash tree update.

6.2 Renewal scheme when a node exits

Ternary Huffman Merkle Hash Tree t as shown below, we assume node x_k exit the cluster. As the number of nodes of the cluster are reduced, node access probability change. We denote by $p_i (i = 1,2,\ldots,7)$ the access probability of node x_i, afer node x_k exit, the access probability of node x_i change to $p_i{}^* = p_i + p_i p_k/(1-p_k)(i = 1,2,\ldots,7)$. There a rather obvious way of testing this, the sum of all $p_i{}^*$ is still 1.

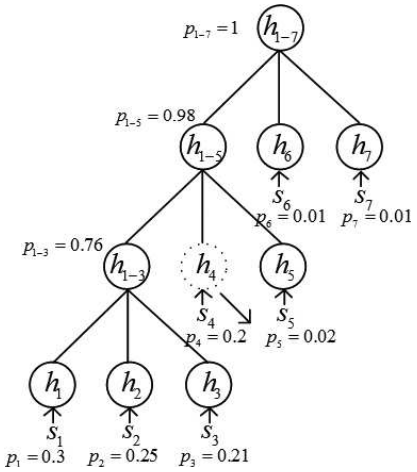


Fig. 7: Node leaves.

Above knowable, after node exits, the access probability of other nodes are recalculated, $p_1 = 0.375$, $p_2 = 0.3125$, $p_3 = 0.2625$, $p_5 = 0.025$, $p_6 = 0.0125$, $p_7 = 0.0125$. According to the construction rules of Ternary Huffman Merkle Hash Tree above, the cluster generates a new hash tree t^* as shown blow. Here s_8^* and h_8^* are virtual nodes, which are not used to calculate the hash value of the whole tree (root). After node joins in, the hash value of the new tree becomes $h_{1-8} = H(H(h_{1-3} \parallel h_5 \parallel h_6) \parallel h_7 \parallel h_8)$.

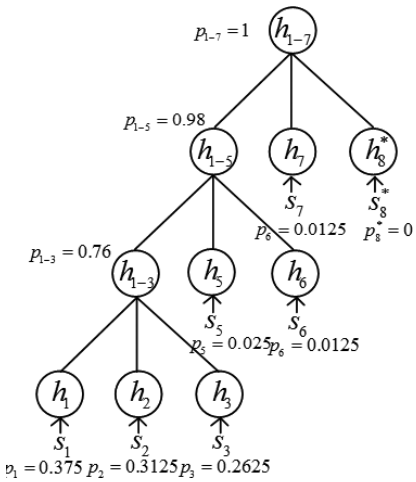


Fig. 8: Hash tree update.

7 Security analysis and performance evaluation

We designed multi-signature cloud storage authentication protocol based on signature of knowledge, the authentication process using symmetric encryption mechanisms to avoid the problem of large public-key cryptographic computations, which can effectively lower network latency and computational load. Certification process only between the node and the target clusters, without the whole network authentication. We adopt the idea of multi-party certification, and each node within the cluster can participate in certification to go, with a strong safety. While the authentication process can be detected within a cluster node or malicious behavior is captured. Security analysis and performance evaluation are given below.

7.1 Security analysis

From there process described above, we can know that sent and received messages are encrypted, it is impossible for malicious nodes to eavesdrop or alter transmitted messages. Also, timestamp is used to prevent replay attack, digital signatures and hash algorithms are mechanisms used to provide data integrity. So, it is hard for malicious nodes to destroy the security and robustness of the system.

Compared with the existing multi-party authentication protocol based on hash tree, our protocol is based on signature of knowledge, nodes send hash value and signature of knowledge of their private keys to the node which to be authenticated during the authentication process, which can maintain the communication's reliability, integration and secrecy and avoid leakage of the private key. The malicious nodes cannot generate valid signature of knowledge without private key even if they get the valid hash value of private key. Timestamp is used with signature of knowledge to against reply attack, which can maintain freshness. Describe no longer about the security for signature of knowledge

7.2 Performance analysis

7.2.1 The number of hash for compute root hash value

As can be seen from the authentication process, multi-signature protocol based on ternary Huffman tree needs a smaller amount of hash computation. We assume there are nodes in the cluster, to meet the requirements of the ternary tree structure, the number of nodes required to meet the conditions. The number of hash for our protocol computer root hash value is, and the number of hash for paper [10] computer root hash value is. Obtained results are shown in figure 9.

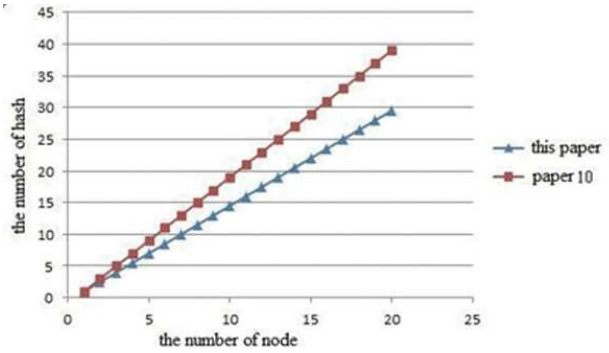


Fig. 9: The number of hash for compute root hash value.

7.2.2 The number of hash for authentication again

From the security analysis we can know that each leaf node do not need send its private key to the authentication node or registered to the network management center. Only need to send the hash value of the key to the authentication node, and through the time-stamped signature of knowledge to ensure their own and their corresponding key freshness. So after a authentication process any new node join in the network, each node only needs to update their signature of knowledge and the signature in certification data set with a signature stamp. The number of hash our protocol required is $sum = 1$, for paper [10] the number is $sum = 2n - 1$. Obtained results are shown in figure 10.

7.2.3 The number of hash when new node adds

Similar to (2), when a new node join in the network we set its access probability 0, and the node is placed at the top of the hash tree. The number of hash our protocol required is $sum = 2$, for paper [10] the number is $sum = 2n - 1$. Obtained results are shown in figure 11.

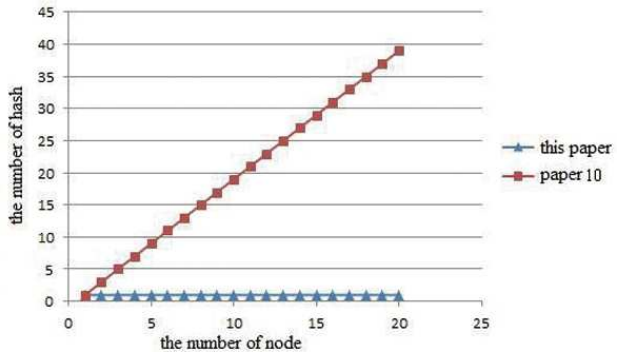


Fig. 10: The number of hash for authentication again.

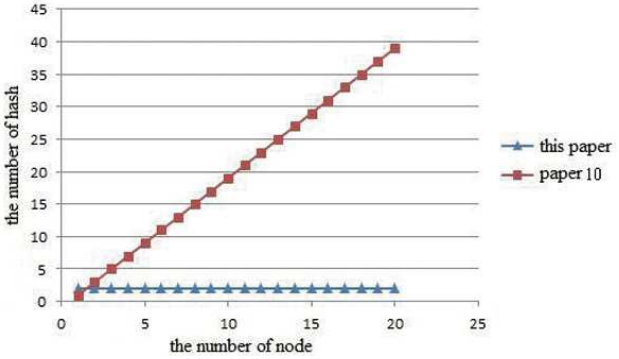


Fig. 11: The number of hash when new node adds.

7.2.4 The number of hash when a node leaves

We assume there are n nodes in the cluster that is divided into m layers, here $m = (n-1)/2$. We suppose that the left node is located in layer j, here $1 \leq j \leq m$. The number of hash our protocol required is $3(n-1)/2 - 3j$, for paper [10] the number is $sum = 2n - 1$ Obtained results are shown in figure 12.

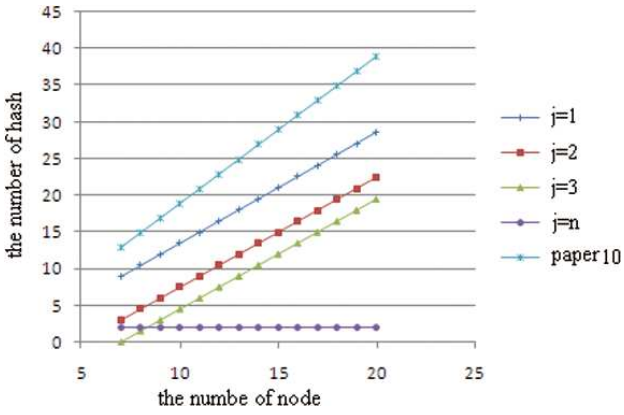


Fig. 12: The number of hash when new node adds 8 Conclusion.

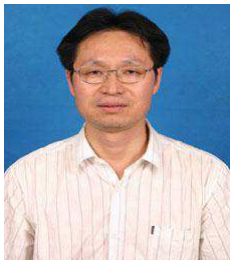
8 Conclusion

Because peer cloud storage network security boundary is not clear, as well as other self-organizing feature makes network face unauthorized access and malicious nodes security threats such as eavesdropping. Therefore, research on other cloud storage network security authentication protocol is very important. In this paper, we use ternary Huffman hash tree, combined with knowledge of the signature, and introduced a tree structure based on access probability algorithm. On this basis, a cloud storage entity authentication scheme is proposed. The agreement has a simple design and high security features, which can resist eavesdropping attacks, replay attacks and forgery attack. Compared with the relevant agreement, our protocol can shorten the certification delay and the calculate load in the same network environment.

References

[1] Kamara, Seny, and Kristin Lauter. "Cryptographic cloud storage." Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2010. 136-149.
[2] Xu, Ke, et al. "A cloud computing platform based on p2p." IT in Medicine & Education, 2009. ITIME'09. IEEE International Symposium on. **1**. IEEE, 2009.
[3] Shin, SeongHan, and Kazukuni Kobara. "Towards secure cloud storage." Demo for CloudCom2010 (2010)
[4] Bowers, Kevin D., Ari Juels, and Alina Oprea. "HAIL: a high-availability and integrity layer for cloud storage." Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009
[5] Kang, Lishan, and Xuejie Zhang. "Identity-based authentication in cloud storage sharing." Multimedia Information Networking and Security (MINES), 2010 International Conference on. IEEE, 2010.
[6] Shraer, Alexander, et al. "Venus: Verification for untrusted cloud storage." Proceedings of the 2010 ACM workshop on Cloud computing security workshop. ACM, 2010.
[7] T Prasath, N Aravindhu, D Rampriya, Augmentation based Secure Storage Authentication and Privacy System in Cloud Environment. International Jouranl of Cloud Computing and Services Science[j]. **2**, (2013).
[8] Shin, Sang-Ho, Dong-Hyun Kim, and Kee-Young Yoo. "A lightweight multi-user authentication scheme based on cellular automata in cloud environment." Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on. IEEE, 2012.
[9] Wang L N, Shi D J, and Qin B P, et al.. Clustering-based merkel hash tree entity authentication scheme [J]. Chinese Journal of Sensors and Actuators, 2007, **20**, 1338-1343.
[10] Ji-wen, Zhou Xian-wei Guo. "HuffMHT-Based Entity Authentication Scheme for Ad hoc Networks." Journal of Electronics & Information Technology, **4**, (2010): 019.
[11] ZENG, Liang-jun, and Wen-feng QI. "Signatures of Knowledge." Journal of Information Engineering University3 (2005): 002.

Ningning Song is engaged in Ph.D. degree study at Department of Communication Engineering, School of Computer and communication Engineering, University of Science and Technology Beijing. His research interests include key management, security and privacy of mobile systems, authentication and secure routing in cloud networks.



Xian-wei Zhou received his B.S. degree in Department of Mathematics from Southwest Teachers' University in 1986 and his M.S. degree in Department of System Science and Mathematics from Zhengzhou University in 1992, and in 1999 he obtained the Ph.D. degree in Department of Transportation Engineering from the Southwest Jiaotong University, P. R. China. He was engaged in postdoctor study at School of Electronic and Information Engineering of Beijing Jiaotong University, P. R. China, from 1999 to 2000. Now, as a professor in Department of Communication Engineering, School of Computer & Communication Engineering, University of Science and Technology Beijing, his research interests include the security of communication networks, next generation networks, mobile IPv6, scheduling theory and game theory.

Qian Liu is engaged in Ph.D. degree study at Department of Communication Engineering, School of Computer and Communication Engineering, University of Science and Technology Beijing. Her research interests include network security, resource allocation of network, scheduling theory, and game theory.