A Distributed Login Framework for Semi-structured Peer-to-Peer Networks

Xiali Hei, Xiaojiang Du
Department of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
Email: {xiali.hei, dux}@temple.edu

Binheng Song
Department of Mathematical Sciences
Tsinghua University
Beijing, 100084, China
Email: bsong@math.tsinghua.edu.cn

***Abstract*—In Peer-to-Peer (P2P) networks, security is a challenging issue due to decentralization. In this paper, we propose an effective distributed login framework for P2P networks. User profile availability is a critical issue in P2P networks. Within the distributed login framework, we propose a new Reed-Solomon erasure code scheme leveraging the Pascal matrix that can guarantee user profile availability. Our performance and security analyses show that: (1) the distributed login framework provides high availability and low redundancy rate; and (2) the new erasure code has low computation and memory overheads.**

***Index Terms*—peer-to-peer networks; Reed-Solomon erasure code; Pascal matrix**

I. Introduction

A Peer-to-Peer (P2P) system is built as an overlay on the existing Internet infrastructure to provide file sharing service to a highly transient population of users (peers). Structured peer-to-peer systems typically use distributed hash table-based (DHT) indexing, such as in the Chord system [1]. Unstructured peer-to-peer systems do not provide any algorithm for organization or optimization of network connections. Three models of unstructured architecture have been defined: pure peer-to-peer systems, hybrid peer-to-peer systems (with supernode), and centralized peer-to-peer systems. The first prominent and popular peer-to-peer file sharing system - Napster, was an example of the centralized model. Gnutella [3] and Freenet [4], on the other hand, are examples of the decentralized model. Kazaa [16] is an example of the hybrid model. Our system is a semi-structured P2P system because we utilize one of the most prominent DHT implementations - the Pastry Protocol [2]; at the same time, we have supernodes to store the popular files.

A login process authenticates a user to the system and allows the system to provide service to the user. Login is a frequently used operation. Some Instant Message software utilizes a central login control server to process login operations. It causes privacy concerns due to both inside and outside attacks. Staffs with access to the login server may get user credentials and profiles. Outside hackers may attack the central server and disrupt services. Also, they may be able to prevent some selected users from accessing their services. The login operation of such a system should not depend on the availability of any particular participating peer, or of any central component, if the latter exists. To address the drawbacks of the centralized model, we propose a distributed login framework.

The login process should have high availability because users may need the service at any time. According to [5], less than 4% of the peers have an uptime of over 10 hours. It is a challenging issue to provide high availability when peers have short uptime.

In this paper, we design a novel distributed login framework for p2p networks. We develop a new erasure code scheme that can assist user profile splitting and combination operations. Our scheme utilizes the Pascal matrix instead of the Vandermonde matrix [9]. Compared with the current schemes that use the Vandermonde matrix, our scheme is faster in encoding phase and it has similar performance in decoding phase.

The rest of the paper is organized as follows. In Section II we discuss related work. In Section III we provide the system model. In Section IV, we present the distributed login framework. We evaluate the performance in Section V, and conclude the paper in Section VI.

II. Related Work

There are a lot of literatures [17], [18], [19], [20], [21], [22], [23] focused on authentication schemes in P2P network. Unlike some P2P authentication schemes, our scheme is not based on PKI [21] or ID-PKI [22], [23]. Since P2P network allows sharing of resources and services by direct interactions among multiple users. To achieve high availability of a file, existing methods use multiple copies and the erasure code scheme [10]. The most popular erasure code scheme is Reed Solomon (RS) code [9], which utilizes Vandermonde matrix V on Galois Fields (2^w). This matrix has a serious drawback: the generation of itself needs multiplication operations on Galois Fields. These operations are time-consuming. Many literatures focused on how to efficiently encode and decode. However, few literatures considered alternative matrices.

The Vandermonde matrix has the problem of fast coefficient expansion. That is, as the dimension of the matrix - t increases, the coefficients (elements of the matrix) increases dramatically [15]. In our research, we propose an alternative matrix - Pascal matrix. The generation of this matrix on Galois Fields (2^w) only needs addition operations. Furthermore, it's a symmetric

matrix, which can reduce 50% operations and memory space comparing to the asymmetric Vandermonde matrix.

III. System Model

We assume that a user runs the client software on his office or personal computer. In the rest of the paper, we use the terms user and node interchangeably.

A user's profile is only stored on a set of trusted nodes such as personal nodes and friend nodes [6]. The set of trusted nodes of a user is referred to as HNs or friend nodes, we will also use them interchangeably. The friends of a user are properly selected with respect to the availability and performance goals. We give priority rankings to friend nodes. For example, the longer the uptime, the higher is the ranking; the closer in distance, the higher is the ranking.

Each user is identified by a unique ID - nodeID. We use HN to denote the nodeID of a friend node. Our system employs a distributed hash table (DHT). This DHT is used for storing the index of the user profile and other meta information, e.g., the current IP address of a user. A user and his friend-node mapping is stored in the DHT in the form of *(ID, value)* pair with ID being the nodeID and value being a friend node. The user to friend node mapping in the DHT is useful for distributing and retrieving a user's profile. Using our proposed storage methods, one can ensure that the data stored inside DHT is highly available and secure in spite of node churn. As a trusted storage is not required by the system design, such a DHT could be hosted at a highly available cloud storage or in publicly available OpenDHT-like services [7]. In our techniques, most of the data is stored on supernodes.

IV. A Distributed Login Framework

A. Security Requirements

The login functionality enables the user to announce his status in the network. During the login process the user authenticates himself with his credential information obtained during the registration phase. After successful login, the joining of the node is announced in the network, and other nodes can contact it directly. There are four security requirements that a distributed login framework needs to satisfy:

Privacy requirement: A friend node should not be able to read the user's profile.

Integrity requirement: A friend can not modify the user's profile.

Storage persistency: If a user has successfully registered, his profile will be kept all the time until the user is revocated.

Availability requirement: If a user provides correct user name and password, he should be able to login successfully.

In addition to the above security requirements, we need to address the issue of how to distribute and retrieve a user's profile.

B. A Distributed Login Framework based on Pastry

To enhance file's availability, existing P2P file sharing methods use multiple copies and the erasure code theory. Consider a user with a file. We split user's file into k pieces. Then we distribute the original k pieces to k peers, while distributing the mixture of the original k pieces to other $m(=n-k)$ peers. Collecting any k shares from these n peers will enable one to reconstruct the original file. Our scheme requires an index server to assist the file piece searching, we choose the Pastry as the DHT protocol in this paper.

1) The Pastry Protocol [2]: Pastry is a distributed lookup protocol which can efficiently locate a node that stores a particular data item. Every node in Pastry Network has a unique 128-bit node identifier(nodeId). A Pastry node with a message and a key can efficiently routes the message to the node, which is numerically closest to the key. Each Pastry node maintains its immediate neighbor list in the nodeId space, and notifies applications of new node arrivals, node failures and recoveries. Pastry nicely takes into account network locality and it automatically adapts to the arrival, departure and failure of nodes. Because nodeIds are randomly assigned, the set of nodes with adjacent nodeId is diverse in geography, ownership, jurisdiction, etc. Leveraging this, Pastry can route to one of l nodes that are numerically closest to the key.

2) The Distributed Login Model: In Pastry network, each node maintain a routing table, neighborhood set and leaf set. Assume that friend nodes are trusted. Every user's friend is a node in the Pastry network. When a user has a new friend, the friend is added to the user's Pastry network as a direct neighbor. We choose friend nodes with higher priority in the Pastry network to store the user's profile pieces. The user has an assisted data server on his own PC or on Cloud. The assisted server distributes and combines the profile pieces. After registration, the assisted server obtains the user's profile. After the user add n friends, the assisted server distributes the profile to n friends. Any k ($k < n$) online friends can recover the user's profile.

3) Profile Distribution and Storage: Existing methods [10-14] have low efficiency and availability because they need to search and then obtain the user profile. Based on the characteristic of a user's profile, we propose a more efficient method for profile management.

In the model, each node on Pastry stores an empty filedb during initialization. Because the user's ID is unique, we can get a 128-bit $resID$ by hash $nodeID$ using SHA-1. During the registration and password update process, the profile needs to be distributed. During registration, user distributes the profile to the assisted server. After the user adding friends to the nodes on Pastry, the assisted server distributes the user's profile to friend nodes according to $resID$. The assisted server (also called the bootstrap node) keeps an index of the friend nodes. The password update process is similar to the registration process. The user sends a command to change the password, the friend nodes will modify the filedb after they receive the command.

When friend nodes (FNs) receive the corresponding profile pieces, they backup their profile pieces on his leaf nodes. For a user that has logged in, if one friend node goes off-line, then one of the nearest leaf node will probably become the new friend node (FN). If there is one node (say A) added before

the FN goes off-line, A may be the new FN. This method guarantees the quick search of nodes with profile pieces.

After the backup node receives the profile piece, first he will check the integrity of the piece. If the check passes, the node store it into filedb. However, he will not back up it to his leaf nodes.

4) The Login Process: First, a user finds the FNs through the bootstrap node (i.e., the assisted data server). Then bootstrap node sends the request to these FNs. The FN checks whether it owns the user's profile piece or not. If the piece does exist, it returns the piece to the user after the integrity check passes. If the integrity check fails, the FN will delete the profile piece from his filedb.

If the FN does not find the user's profile piece, it will inquire the profile piece from his leaf nodes. If successfully finding the profile piece, it will retrieve the profile piece from the successor and forward it to the user. The user then combines all the needed profile pieces, if it matches his original profile, the user passes. Otherwise, the user cannot login.

5) Securing User Profile Pieces: To secure user profile pieces, a straightforward method is to encrypt the profile pieces using user's public key. We can also use MD5 checksum of encrypted pieces to detect falsifications.

C. The Profile Distribution Scheme

We present the profile distribution scheme below.

Step 1: The server evenly split the file into k pieces: $x_1, x_2, ..., x_k$. Each partner (peer) i gets one piece x_i.

Step 2: The server distributes the file pieces to the (k+1)th party to the $(k+m)$th party as follows:

The $(k+1)$th party gets: $y_{k+1} = x_1 + x_2 + ... + x_k$

The $(k+2)$th party gets: $y_{k+2} = x_1 + 2x_2 + 3x_3 + ... + kx_k$

... .

The $(k+m)$th party gets: $y_{k+m} = x_1 + C_m^1 x_2 + C_{m+1}^2 x_3 + ...C_{k+m-2}^{m-1} x_k$

Step 3: To reconstruct the file from any k parties, we only need to obtain the exact solution to the k linear equations (from the above $(k+m=n)$ equations).

We design the coefficients of the equations according to the following rules: $d_{p,q} = d_{p,q-1} d_{p-1,q}$ ($p > 1$, $q > 1$), with $d_{1,i} = 1$ and $d_{i,1} = 1$ for integer i.

$$\begin{aligned} &y_i = x_i, \quad i = 1, 2...k; \\ &y_{k+1} = x_1 + x_2 + x_3 ... + x_k \\ &\cdots \\ &y_{k+m} = x_1 + C_m^1 x_2 + C_{m+1}^2 x_3 + ... + C_{n+m-2}^{m-1} x_k \end{aligned} \tag{IV-1}$$

The coefficients of the linear equations form a matrix A:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & . & 0 \\ 0 & 1 & 0 & 0 & . & 0 \\ \cdots & & & & & \\ 0 & 0 & 0 & 0 & . & 1 \\ 1 & 1 & 1 & 1 & . & 1 \\ 1 & 2 & 3 & 4 & . & k \\ \cdots & & & & & \\ 1 & m & C_{m+1}^2 & C_{m+2}^3 & . & C_{k+m-2}^{m-1} \end{pmatrix} \tag{IV-2}$$

We split matrix A into two matrices I and D. Matrix I is a $k \times k$ indentity matrix, and the rank of it is k. According to our design, in matrix D, $d_{p,q} = d_{p-1,q} + d_{p-1,q}$ ($p > 1$, $q > 1$). Hence, the last (right-most) element in the ith row is not less than the sum of the last numbers from the 1st row to the (i-1)th row, while the first number in the ith row is equal to the first number in the first (i-1) rows). Hence, the ith row cannot be represented by a linear combination of the first (i-1) rows. So, the rank of D is $min(k,\ m)$. If we assume $m < k$, then it means the m row vectors of matrix D are linearly independent. Actually, D is a Pascal matrix, and the elements of the symmetric Pascal matrix are the binomial coefficients, i.e. $d_{pq} = C_z^r$, where $z = p + q - 2$, $r = i - 1$.

$$A = \begin{pmatrix} I \\ D \end{pmatrix} \tag{IV-3}$$

$$D = \begin{pmatrix} 1 & 1 & 1 & 1 & . & 1 \\ 1 & 2 & 3 & 4 & . & k \\ 1 & 3 & 6 & 10 & . & C_{k+1}^2 \\ \cdots & & & & & \\ 1 & m-1 & C_m^2 & C_{m+1}^3 & . & C_{k+m-3}^{m-2} \\ 1 & m & C_{m+1}^2 & C_{m+2}^3 & . & C_{k+m-2}^{m-1} \end{pmatrix} \tag{IV-4}$$

D. Proofs of Correctness

The rank of an $m \times k$ matrix is at most min(m, k). A matrix that has a rank as large as possible is said to have full rank. The solution is unique if and only if the rank of the augmented matrix is equal to the number of variables. Otherwise the general solution has w free parameters where w is the difference between the number of variables and the rank. This theorem is discovered by Rouch's and Capelli. To prove the correctness, the main challenge is to prove that any k dimension row vectors consisted of coefficients are linearly independent.

Lemma 1: The $m \times k$ matrix D is full rank.

Proof: Here we suppose m=min(m,k) to simplify the proof. Let A_{dt} is a matrix made up of the first t rows and first t columns of D.

$$A_{dt} = \begin{pmatrix} 1 & 1 & 1 & 1 & . & 1 \\ 1 & 2 & 3 & 4 & . & k \\ 1 & 3 & 6 & 10 & . & C_{k+1}^2 \\ \cdots & & & & & \\ 1 & t-1 & C_t^2 & C_{t+1}^3 & . & C_{k+t-3}^{t-2} \\ 1 & t & C_{t+1}^2 & C_{t+2}^3 & . & C_{k+t-2}^{t-1} \end{pmatrix} \tag{IV-5}$$

Step 1: Perform the row transformations on A_{dt}: we let the $(i+1)$th row subtract ith row, (for $i = t-1, ..., 1$), then the A_{dt} turn out to be:

$$A_{dt} = \begin{pmatrix} 1 & e \\ f & A_{d(t-1)} \end{pmatrix} \tag{IV-6}$$

where f is a zero column vector, e is a row vector with nonzero elements, $A_{d(t-1)}$ is a submatrix of A_{dt} by deleting the first row and the last column. The reason is that $d_{i,j} - d_{i-1,j} = d_{i,j-1}$, so every element becomes its left neighbor. Because

f is a zero column vector while the first element in first row vector is 1, the first row vector cannot be represented by the row vectors after it. At this time, the problem turns out how to prove that $A_{d(t-1)}$ is full rank.

Step 2: We iteratively do Step 1, eventually we get one matrix with only one nonzero element 1, which is the most left and lowest element. Since it is full rank, this is trivial.

Till now, the proof of Lemma 1 is done. So that means m row vectors in D are linearly independent. If we pick any $t < m$ row vectors in D, they are linearly independent, too. We pick the first k columns from A, and randomly pick k row vectors from these first k columns to form a new matrix E.

Lemma 2: E is a full rank $k \times k$ matrix consisted of k row vectors in A. If any row vector of E is replaced with a row vector in $A \setminus E$, the new matrix is still full rank.

Proof: here we use mathematical induction on m.

Base case: when $m = 1$, by observation method, a k dimension row vectors consisted of coefficient is matrix E_1, and it has two cases:

Case 1.1: In E_1, $(k-1)$ row vectors are from I and one row vector is from D, it is $(1, 1, 1, ..., 1)$, because I is $k \times k$ identify matrix, so these (k-1) row vectors made up of coefficients are linearly independent, and the k dimension row vector $(1, 1, 1, ..., 1)$ cannot be represented by these $(k-1)$ row vectors because the rank of the matrix consisted of these $(k-1)$ row vectors is $(k-1)$ rather than k, so these k row vectors consisted of coefficients are linearly independent.

Case 1.2: In E_1, (k) row vectors are from I, because I is $k \times k$ unit matrix, so these k row vectors consisted of coefficients are linearly independent.

Combined these two cases, when $m = 1$, the Lemma 2 is correct. Then, we assume that for all positive integers $m = s$, any k row vectors in A are linearly independent.

Suppose these k row vectors in A constitute a new matrix E_s. We also assume that i row vectors in E_s are from matrix B, the other $(k-i)$ row vectors are from matrix D.

When $m = s + 1$, if we replace one row vector in E_s with the new row vector $(1, (s+1), C_{s+2}^2, C_{s+3}^3, ..., C_{k+s-1}^s)$, we called the new matrix is E_{s+1}.

Case s.1: The replaced row vector is from I.

Because the remaining $(i-1)$ row vectors in E_{s+1} from matrix I are unit vector, so they can not represent the new vector $(1, (s+1), C_{s+2}^2, C_{s+3}^3, ..., C_{k+s-1}^s)$. Thus, the new vector is linearly independent to these remaining $(i-1)$ row vectors from matrix I in E_{s+1}. By Lemma 1, the new vector is linearly independent with the other $(k-i)$ row vectors from matrix D in E_{s+1}. According to the induction hypothesis, the unreplaced $(k-1)$ vectors are mutually linearly independent because the all k vectors are mutually linearly independent. So the the rank of new matrix E_{s+1} is still k. That means, the new k row vectors are linearly independent.

Case s.2: The replaced row vector is from D.

Because the other i row vectors in E_{s+1} from matrix I are unit vector, so they cannot represent the new vector $(1, (s+1), C_{s+2}^2, C_{s+3}^3, ..., C_{k+s-1}^s)$. Thus, the new vector is linearly independent to these remaining i row vectors from matrix I in E_{s+1}. By Lemma 1, the new vector is linearly independent to the other $(k-i-1)$ row vectors from matrix D in E_{s+1}. According to the induction hypothesis, the remaining $(k-1)$ vectors are mutually linearly independent because the all k vectors are mutually linearly independent. So the rank of new matrix E_{s+1} is still k. That means, the new k row vectors are linearly independent.

Hence, by the second principle of induction, Lemma 2 is true.

Theorem 1: Any k row vectors in matrix A are linearly independent.

Proof: Initially, we pick k row vectors are from I, they are linearly independent. Then, we replace one row with row vector from D, this is case 1.1, so it is correct. After that, we replace one row in the updated k dimension row vectors with one row vector from D, so it is true according to Lemma 2. After we finish this, we delete the picked row vector in D. We iteratively do this until there are no more rows left in D, and the k row vectors are still linearly independent. So, any k row vectors in matrix A are linearly independent. We are done.

V. Performance Evaluation

A. Computation Complexity

Reed Solomon erasure code scheme is based on Galois Field GF(2^w). This is to avoid file expansion due to the coefficients increasing with the k. We will discuss computation complexity in two phases: encoding phase and decoding phase.

Encoding Phase: Adding a new element in Pascal matrix just involves addition operation while multiplication operation in Vandermonde matrix. Vandermonde matrix V have a fatal drawback that is the generation of it needs unavoidable multiplication operations on Galois Fields. These operations are time-consuming. While generation of Pascal matrix on Galois Fields (2^w) just needs addition operation. What's more, it's symmetric that can reduce 50% operations and memory space compared with the asymmetric Vandermonde matrix. In the encoding phase (also known as computing checksum phase), Pascal matrix is faster than Vandermonde matrix.

Decoding Phase: During the decoding phase, we need to get the inverse matrix of the matrix E. E is not a Pascal matrix, neither is Vandermonde matrix. We assume E has $k - s$ unit vectors and other s row vectors from Pascal matrix or Vandermonde matrix. For this case, there is no fast method to get the exact solution to the corresponding linear equations due to exploiting the Fast Fourier Transform to solve linear equations needs strict condition constrains. So the efficient methods are Guass elimination, LU decomposition and Cholesky decomposition. Guass elimination is the most common-use method. If we utilize this method, the computation complexity of decoding phase is $k + ks + s^3$. When $s \ll k$, it is roughly equal to k, where E is a sparse matrix. If $s \approx k$, then it is roughly equal to s^3.

In a summary, Pascal matrix can be used to replace the Vandermonde matrix in Reed-Solomon erasure code scheme.

Compared with Vandermonde matrix, it is faster than Vandermonde matrix in encoding phase, while having similar performance in decoding phase.

B. Storage Requirement

Assume the size of the whole file is S, now the size of every piece is almost S/k, there are nl pieces totally. So the redundancy rate is $(n*l)/k-1$, where l is the number of leaf nodes.

C. Availability Analysis

Availability here means when friends (nodes) join or quit, the profile won't be lost and the user can login successfully at any time.

Hence, there should be nodes that can locate the needed profile pieces successfully in Pastry. The redundancy backup can address the lost profile pieces when the user's friend joins or leaves. If there are no malicious nodes, one independent backup node will store the profile pieces on his leaf nodes. If in a stabilization cycle, all of his leaf nodes quit, the profile piece will be lost. However the other profile pieces can recover the original file. Another approach is increasing the number of backup pieces on assisted server. This will increase the backup messages linearly while availability almost exponentially.

D. Security Analysis

Here we focus on the security of the profile. We assume that malicious nodes are possible but the malicious nodes cannot modify or access the data without authorization. At the same time, the tampered or falsified data pieces can be found timely and discarded. We assume traditional encryption is secure. The user ($resID$) can generate pieces a_1 and a_2 consistently. Since the distribution and combination profile need the user ID, the assisted server can combine $hash(ID)$ with $resID$ to determine the a_1 or a_2 firstly, then determine the respective location I_1 and I_2 according to Pastry protocol. Because both I_1 and I_2 do not know that the user's profile are stored on them, this method does work. Thus, the security of system depends on the hash function of user ID.

E. Feasibility Analysis

In this model, if a user has few friends, this may affect the availability. We can use an assisted data server (without authentication and control functions) to store the encrypted profile pieces. The user distributes the original profile to the assisted data server during the registration. Then the assisted data server distributes the profile to the Pastry periodically. This method decreases the number of backup nodes and guarantees the availability of the system.

VI. Conclusion

In Peer-to-Peer (P2P) networks, security is a challenging issue due to decentralization. In this paper, we proposed an effective distributed login framework. User profile availability is an important issue in P2P networks. Within the distributed login framework, we proposed a new erasure code scheme. This new erasure code scheme uses Pascal matrix instead of Vandermonde matrix. Compared with Vandermonde matrix, our scheme only uses 50% of the time for encoding, while has similar performance in decoding phase. Our performance and security analysis showed that the distributed login framework provides high availability and low redundancy rate.

VII. Acknowledgment

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-0963578, CNS-1002974, CNS-1022552, and CNS-1065444, as well as the US Army Research Office under grant W911NF-08-1-0334.

References

[1] I. Stoica, R. Mprris, D. Karger et al., "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proc. of ACM SIGCOMM 2001*, August 2001, pp. 149-160.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001, pp. 329 - 350.

[3] http://www.gnu.org/philosophy/gnutella.html

[4] https://freenetproject.org/index.html

[5] J. A. Pouwelse, P. Garbacki, D.H.J. Epema et al., "The Bittorrent P2P filesharing system: Measurements and analysis," in *Proc. of IPTPS'05*, Feb 2005, vol. 3640 of LNCS, pp. 205-216.

[6] A. Datta, "Tutorial: Peer-to-Peer Storage Systems: Crowdsourcing the storage cloud," in *Proc. of ICDCN 2010*, Kolkata, India, 2010.

[7] G. Urdaneta, G. Pierre, and M. V. Steen, "A Survey of DHT Security Techniques," *ACM Computing Surveys*, 2009.

[8] NIST. SECURE HASH STANDARD. http://csrc.nist.gov/publications /fips/fips180-2/fips180-2withchangenotice.pdf.

[9] J. Plank, "A tutorial on reed-solomon coding for faulttolerance in raid-like systems," *Software Practice and Experience*, Vol. 27, No.9, Sept. 1997, pp. 995-1012.

[10] J. Kubiatowicz et al., "OceanStore: An Architecture for Global-Scale Persistent Storage," in *Proc. of ASPLOS'00*, 2000, pp. 190-201.

[11] W. G. Yee and L. T. Nguyen, "A view of the data on P2P file-sharing systems," Jounal of The american society for information science and technology, Vol. 60, No. 10, 2009, pp. 2132-2141.

[12] J. Leu and M. Huang, "Comparison of Piece-Based File Sharing Schemes over a Peer-to-Peer Network in a Heterogeneous Network Environment," in *Proc. of GLOBECOM'09*, 2009, pp. 1-6.

[13] W. Sang and D. Qiu, "On The Efficiency of Peer-To-Peer File Sharing," in *Proc. of ICME'07*, 2007, pp. 32-35.

[14] J. Lu and J. Callan, "Content-based retrieval in hybrid peer-to-peer networks," in *Proc. of ACM CIKM'03*, 2003, pp. 199-206.

[15] X. Hei, X. Du, B. Song. Two matrices for Blakley's Secret Sharing Scheme. ICC 2012.

[16] http://en.wikipedia.org/wiki/Kazaa

[17] Z. Li, X. Xu and L. Shi et al, "Authentication in Peer-to-Peer Network: Survey and Research Directions, in proc. of Third International Conference on Network and System Security, 2009, pp. 115-122.

[18] Z. Min, "P2P Software Security Research Based on Key Issues, Shanghai Jiaotong Universitu, Doctoral thesis, 2008.

[19] L. Lu, J. Han and Y. Liu et al, "HuaiLionel M. Ni, Jian Ma, Pseudo Trust: Zero-Knowledge Authentication in Anonymous P2Ps, IEEE Tran. On parallel and distributed systems, vol. 19, no. 10, 2008, pp. 1325-1337.

[20] Y. Zhang and D. Zhang, "Authentication and access control in P2P network, in proc. of GCC2003, 2004, pp. 468-470.

[21] K. Berket, A. Essiari and A. Muratas, "PKI-Based Security for Peer-to-Peer Information Sharing,, in proc. of the Fourth International Conference on Peer-to-Peer Computing, 2004, pp. 45 -52.

[22] S. Ryu, K. Butler and P. Traynor et al, "Leveraging indentity-based cryptography for node ID assignment in structured p2p systems, in proc. of IEEE AINAW07, 2007, pp. 1803-1815.

[23] K. V. Nguyen, "Simplifying peer-to-peer device authentication using identity-based cryptography, in proc. of the International conference on Networking and Services, 2006, pp. 43-46.