BAJI SHAHEED BABA SHAIK, G CHINNA BABU, UPPE NANAJI/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
www.ijera.com
Vol. 1, Issue 4, pp.1803-1810

IMPLEMENTATION OF P2P REPUTATION MANAGEMENT SCHEME BASED ON DISTRIBUTED IDENTITIES AND DECENTRALIZED RECOMMENDATION CHAINS

BAJI SHAHEED BABA SHAIK[1], G CHINNA BABU[2], UPPE NANAJI[3]

[1]DEPARTMENT OF CSE, ST.THERESSA INSTITUE OF ENGINEERING AND TECHNOLOGY, GARIVIDI, VIZIANAGARM DISTRICT, ANDRA PRADESH, INDIA,
[2]ASST.PROFESSOR, DEPARTMENT OF CSE, ST.THERESSA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GARIVIDI, VIZIANAGARM DISTRICT, ANDRA PRADESH, INDIA.
[3]Associate professor, Department of computer science, St. Theressa Institute of Engineering and Technology, Garividi (Cheepurupally), Andra pradesh, India

Abstract—The motivation behind basing applications on peer-to-peer architectures derives to a large extent from their ability to function, scale and self-organize in the presence of a highly transient population of nodes, network and computer failures, without the need of a central server and the overhead of its administration. P2P networks are vulnerable to peers, who cheat, propagate malicious codes, or peers who do not cooperate. Traditional client-server security models are not sufficient to P2P networks because of their centralized nature. To full fill this ,in this paper we present a cryptographic protocol for ensuring secure and timely availability of the reputation data of a peer extremely at low cost and develop a queuing model to evaluate the time required at each peer to serve its replication requests.

***Index Terms*— Object Oriented Approach; P2P; attack; Encryption and Decryption.**

I. INTRODUCTION

Peer-to-Peer (P2P) overlays like CAN [1], Chord [2], Pastry [3] and Tapestry [4] provide a self-organizing substrate for large-scale peer-to-peer applications. These systems provide a powerful platform for the construction of a variety of decentralized services, including network storage, content distribution, and application-level multicast. Structured overlays allow applications to locate any object in a probabilistically bounded, small number of network hops, while requiring per-node routing tables with only a small number of entries. Moreover, the systems are scalable, fault tolerant and provide effective load balancing. Methods like generating trust and protecting client-server networks cannot be used for pure P2P networks. This is because trusted central authority is not used in the P2P networks.
However, to fully realize the potential of the P2P paradigm, such overlay networks must be able to

support an open environment where mutually distrusting parties with conflicting interests are
allowed to join. Even in a closed system of sufficiently large scale, it may be unrealistic to assume that none of the participating nodes have
been compromised by attackers. All peers in the P2P network are identified by identity certificates. Reputation of the peer is attached to its identity. Identity certificates are generated using self-certification method, and all peers maintain their own identity certificate authority which issues the identity certificate to the peer. Each peer owns the reputation information pertaining to all its past transactions with other peers in the network and stores it locally. A two party cryptographic protocol not only protects the reputation information between the two peers participating in the transaction. Proposing technique not only reduces the percentage of malicious transactions in the network but also significantly reduces the network traffic compared to other reputation systems.
The main contributions of this paper are:

- A self-certification-based identity system protected by cryptographically blind identity mechanisms.
- A light weight and simple reputation model.
- An attack resistant cryptographic protocol for generation of authentic global reputation information of

II. LITERATURE SURVEY

This section briefly reviews some of the existing P2P reputation systems, focusing particularly on the storage and integrity issues. We start by giving an overview of the reputation systems.

Kevin A. Burton designed the open privacy distributed reputation system [5] on p2p, which is derived from the distributed trust model. It proposed the concept of reputation network, which is composed by identities and certificates. Therefore, the trustworthiness of the identities can be estimated from a visible sub-graph of the reputation network.

P2PREP et.al. [6] is a reputation sharing protocol proposed for Gnutella, where each peer keeps track and shares with others the reputation of their peers. Reputation sharing is based on a distributed polling protocol. Service requesters can access the reliability by polling peers.

Karl Aberer et.al. [7] proposed a trust managing system on the P2P system. It integrates the trust management and data management schemes to build a full-fledged P2P architecture for information systems. The reputations in this system are expressed as complaints; the more complaints a peer gets, the less trustworthy it could be. After each transaction, and only if there is dissatisfaction, a peer will file a complaint about the unhappy experience. To evaluate the reputation of a peer involves searching for complaints about the peer.

Kamvar et.al [8] proposed a reputation management system, for P2P file sharing systems such as Gnutella to combat the spread of inauthentic file. In their system, each peer is given a global reputation that reflects the experiences of other peers with it.

Sit and Morris [9] present a framework for performing security analyses of p2p networks. Their adversarial model allows for nodes to generate packets with arbitrary contents, but assumes that nodes cannot intercept arbitrary traffic. They then present a taxonomy of possible attacks. At the routing layer, they identify node lookup, routing table maintenance, and network partitioning / virtualization as security risks. They also discuss issues in higher-level protocols, such as file storage, where nodes may not necessarily maintain the necessary invariants, such as storage replication. Finally, they discuss various classes of denial-of-service attacks, including rapidly joining and leaving the network, or arranging for other nodes to send bulk volumes of data to overload a victim's network connection (i.e., distributed denial of service attacks).

Dingledine *et al.* [10] and Douceur [11] discuss address spoofing attacks. With a large number of potentially malicious nodes in the system and without a trusted central authority to certify node identities, it becomes very difficult to know whether you can trust the claimed identity of somebody to whom you have never before communicated.

Bellovin [12] identifies a number of issues with Napster and Gnutella. He discusses how difficult it might be to limit Napster and Gnutella use via firewalls, and how they can leak information that users might consider private, such as the search queries they issue to the network. Bellovin also expresses concern over Gnutella's "push" feature, intended to work around firewalls, which might be useful for distributed denial of service attacks. He considers Napster's centralized architecture to be more secure against such attacks, although it requires all users to trust the central server. It is worthwhile mentioning a very elegant alternative solution for secure routing table maintenance and forwarding that we rejected. This solution replaces each node by a group of diverse replicas as suggested by Lynch *et al.* [13]. The replicas are coordinated using a state machine replication algorithm like BFT [14] that can tolerate Byzantine faults. BFT can replicate arbitrary state machines and, therefore, it can replicate Pastry's routing table maintenance and forwarding protocols.

In this paper, we investigate Reputation Systems for P2P networks—a more ambitious approach to protect the P2P network without using any central component, and thereby harnessing the full benefits of the P2P network.

III. SYSTEM ARCHITECTURE

In this section we discuss about system architecture. Figure 1 gives a sketch of the system architecture of peer trust. There is no central database. Trust data that are needed to compute the trust measure for peers are stored across the network in a distributed manner. The callout shows that each peer has a trust manager that is responsible for feedback submission and trust evaluation, a small database that stores a portion of the global trust data, and a data locator for placement and location of trust data over the network. Figure 2 shows a simple example of a peer trust network of 6 peers constructed. For such a data location scheme, there is a trust issue associated with it, namely, peers may misbehave by providing false data or random data when responding to a search request.

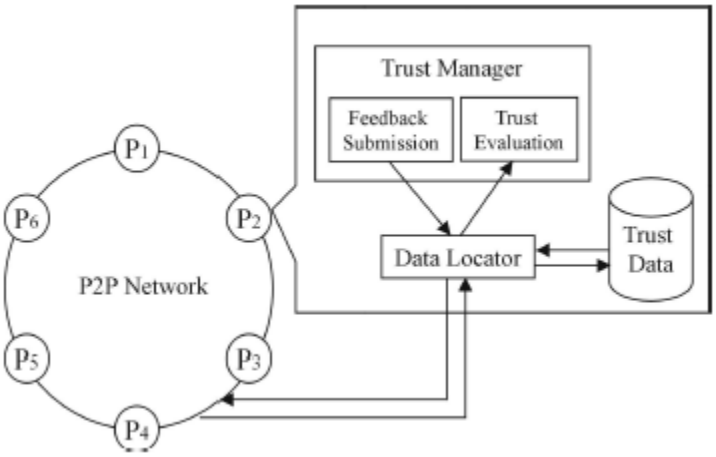

Figure 1 System architecture

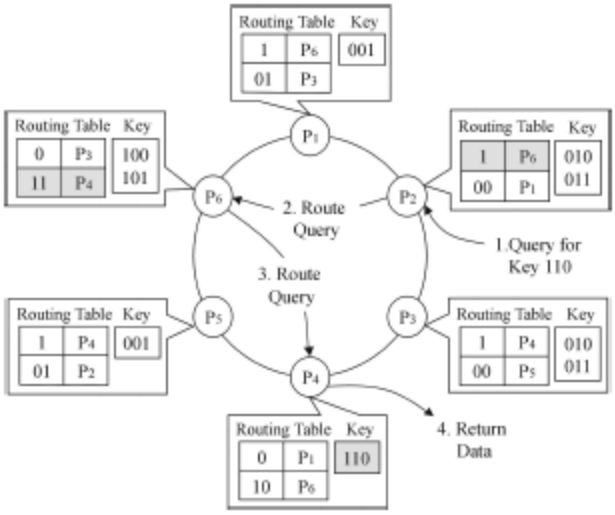

Figure 2 Data location

A. Existing System:

The peers in the P2P network have to be discouraged from leeching on the network. It has been shown in Tragedy of Commons that a system where peers work only for selfish interests while breaking the rules decays to death. Policing these networks is extremely difficult due to the decentralized and ad hoc nature of these networks. Besides, P2P networks, like the Internet, are physically spread across geographic boundaries and hence are subject to variable laws.

The traditional mechanisms for generating trust and protecting client-server networks cannot be used for pure1 P2P networks. This is because the trusted central authority used in the traditional client-server networks is absent in P2P networks. Introduction of a central trusted authority like a Certificate Authority (CA) can reduce the difficulty of securing P2P networks. The major disadvantage of the centralized approach is, if the central authority turns malicious, the network will become vulnerable. In the absence of any central authority, repository, or global information, there is no silver bullet for securing P2P networks.

B. Proposed System:

In this paper, we investigate Reputation Systems for P2P networks—a more ambitious approach to protect the P2P network without using any central component, and thereby harnessing the full benefits of the P2P network. The reputations of the peers are used to determine whether a peer is a malicious peer or a good peer. Once detected, the malicious peers are ostracized from the network as the good peers do not perform any transactions with the malicious peers. Expulsion of malicious peers from the network significantly reduces the volume of malicious activities. All peers in the P2P network are identified by identity certificates (aka identity). The reputation of a given peer is attached to its identity. The identity certificates are generated using self-certification, and all peers maintain their own (and hence trusted) certificate authority which issues the identity

certificate(s) to the peer. Each peer owns the reputation information pertaining to all its past transactions2 with other peers in the network, and stores it locally. A two-party cryptographic protocol not only protects the reputation information from its owner, but also facilitates secure exchange of reputation information between the two peers participating in a transaction.

IV. IMPLEMENTATION

In this paper, we consider seven modules.

- Login Module
- Active Node in Dynamic root
- Group Controller
- Trusted Group Members
- Data Transfer
- Find Group Key
- Block Untrusted Users

A. Login Module

Authentication checking: In this module checks whether the user is authenticated or not if the user is authenticated then they have the permission to process further transactions otherwise they cannot access any transaction in this system.

Registration process: If the new user to this system first they must registered in the register module after they have continue to process in the system. During the registration the user must enter the valid information for create new user name and password if only valid user. Once user registered after they have authorized user of this system.

B. Active Node in Dynamic route

In our communication group have number of client nodes are interconnected in the server. Each group has the separate group key for communication in the group. When a new member joins or leaves the communication group, only it's reflecting for local subgroup. The each group has separate group key for communication in between who are in the communication group in that time.

C. Group Controller

Backward Secrecy: Backward Secrecy is used to prevent a new member from decoding messages exchanged before it joined the group. This property guarantees that a passive adversary who knows a subset of group keys cannot discover the previous group keys.

Forward Secrecy: Forward Secrecy is used to prevent a leaving user or expelled group member to continue accessing the group communication. This property guarantees that a passive adversary who

D. Trust Group Members

Our protocol directly addresses the problem of reducing the overload of the group controller. We divide the multicast communication group into regional subgroups. Each subgroup is independently managed by a subgroup controller (SGC) like a separate multicast group with its own subgroup key. Thus, when a member joins or leaves the communication group, it joins or leaves only its local subgroup. As a result, only the local subgroup communication key needs to be refreshed and the scalability problem is greatly mitigated.

E. Data Transfer

In cryptography encryption is the process of transforming information referred to as plaintext using an algorithm called cipher to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information in cryptography, referred to as ciphertext. In many contexts, the word encryption also implicitly refers to the reverse process, called decryption. In our multicast communication group mainly concentrates on enabling the data transfer among the server and multiple clients in the network communications. The server sends encrypted data and clients receive the decrypted data.

Encryption: Encryption is the conversion of data into a form, called a cipher text that cannot be easily understood by unauthorized people. The translation of data into a secret code. Encryption is the most effective way to achieve data security. In our communication protocol each sub group maintain the separate group key for the communication among the network. Server send data transfer to multiple clients in encrypted data because in between the data transfer unauthorized people cannot see easily what the server sending data to clients. This process is shown in Figure 3 and source code is shown in table 1.

Table 1 source code for File Encryption

```
private void blowfishEncrypt(byte in[], int off, byte out[], int outOff){
int L = (in[off] & 0xff) << 24 | (in[off + 1] & 0xff) << 16 | (in[off + 2] & 0xff) << 8 |
in[off + 3] & 0xff;
int R = (in[off + 4] & 0xff) << 24 | (in[off + 5] & 0xff) << 16 |
(in[off + 6] & 0xff) << 8 | in[off + 7] & 0xff;
 L ^= P[0];
       for(int i = 0; i < rounds;)
       {
          int a = L >>> 24 & 0xff;
          int b = 0x100 | L >>> 16 & 0xff;
          int c = 0x200 | L >>> 8 & 0xff;
          int d = 0x300 | L & 0xff;
          R ^= (sKey[a] + sKey[b] ^ sKey[c]) + sKey[d] ^ P[++i];
          a = R >>> 24 & 0xff;
          b = 0x100 | R >>> 16 & 0xff;
          c = 0x200 | R >>> 8 & 0xff;
          d = 0x300 | R & 0xff;
          L ^= (sKey[a] + sKey[b] ^ sKey[c]) + sKey[d] ^ P[++i];
       }
R ^= P[rounds + 1];
out[outOff] = (byte)(R >>> 24 & 0xff);
out[outOff + 1] = (byte)(R >>> 16 & 0xff);
out[outOff + 2] = (byte)(R >>> 8 & 0xff);
out[outOff + 3] = (byte)(R & 0xff);
out[outOff + 4] = (byte)(L >>> 24 & 0xff);
out[outOff + 5] = (byte)(L >>> 16 & 0xff);
out[outOff + 6] = (byte)(L >>> 8 & 0xff);
out[outOff + 7] = (byte)(L & 0xff);
}
```

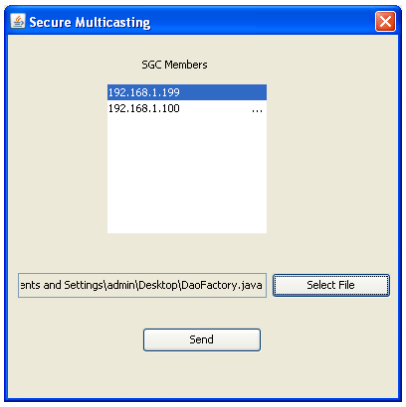

Figure 3 encryption file and send

Decryption: Decryption is the process of converting encrypted data back into its original form. The process of decoding data that has been encrypted into a secret format. Decryption requires a secret key or password. Server send data transfer to multiple clients in encrypted data because in between the data transfer unauthorized people cannot see easily what the server sending data to clients. After receiving encrypted data back into original form in client side. Converting encryption and decryption only for security purpose. This process is shown in Figure 4 and source code is shown in table 2.

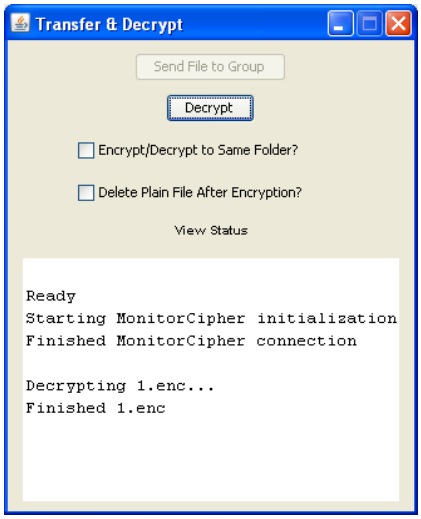

Figure 4 file decryption and receive

Table 2 source code for File decryption

```
private void blowfishDecrypt(byte in[], int off, byte out[], int outOff){
int L = (in[off] & 0xff) << 24 | (in[off + 1] & 0xff) << 16 | (in[off + 2] & 0xff) << 8 |
in[off + 3] & 0xff;
int R = (in[off + 4] & 0xff) << 24 | (in[off + 5] & 0xff) << 16 | (in[off + 6] & 0xff) << 8 |
in[off + 7] & 0xff;
L ^= P[rounds + 1];
        for(int i = rounds; i > 0;)
        {
           int a = L >>> 24 & 0xff;
           int b = 0x100 | L >>> 16 & 0xff;
           int c = 0x200 | L >>> 8 & 0xff;
           int d = 0x300 | L & 0xff;
           R ^= (sKey[a] + sKey[b] ^ sKey[c]) + sKey[d] ^ P[i--];
           a = R >>> 24 & 0xff;
           b = 0x100 | R >>> 16 & 0xff;
           c = 0x200 | R >>> 8 & 0xff;
           d = 0x300 | R & 0xff;
           L ^= (sKey[a] + sKey[b] ^ sKey[c]) + sKey[d] ^ P[i--];
        }
R ^= P[0];
out[outOff] = (byte)(R >>> 24 & 0xff);
out[outOff + 1] = (byte)(R >>> 16 & 0xff);
out[outOff + 2] = (byte)(R >>> 8 & 0xff);
out[outOff + 3] = (byte)(R & 0xff);
out[outOff + 4] = (byte)(L >>> 24 & 0xff);
out[outOff + 5] = (byte)(L >>> 16 & 0xff);
out[outOff + 6] = (byte)(L >>> 8 & 0xff);
out[outOff + 7] = (byte)(L & 0xff);
}
```

F. Find Group Key

We divide the multicast communication group into regional subgroups. Each subgroup is independently managed by a subgroup controller like a separate multicast group with its own subgroup key. when a new member joins in the communication group then we create a new group key for only for its local group as wells as existing member leaves from the communication group after that they don't want to access the local subgroup so only the local subgroup communication key needs to be refreshed. Source code for this is shown in Table 3.

Table 3 source code for Generate Group key

```
private synchronized void makeKey(BlowKey key){
    byte kk[] = key.getEncoded();
    if(kk == null)
      throw new KeyException("Null Blowfish key");
    int len = kk.length;
    if(len == 0)
      throw new KeyException("Invalid Blowfish user key length");
    if(len > 56)
      len = 56;
    System.arraycopy(S0, 0, sKey, 0, 256);
    System.arraycopy(S1, 0, sKey, 256, 256);
    System.arraycopy(S2, 0, sKey, 512, 256);
    System.arraycopy(S3, 0, sKey, 768, 256);
    int i = 0;
    int j = 0;
    for(; i < rounds + 2; i++)
    {
      int ri = 0;
      for(int k = 0; k < 4; k++)
      {
        ri = ri << 8 | kk[j++] & 0xff;
        j %= len;
      }
      P[i] = Pi[i] ^ ri;
    }
```

G. Block Untrusted Users

Trusted users are formed a group. If a new member request to join in group, the IP Address will be validated. IP address will be validate with the help of subnet masking, such as the Class A, Class B, Class C Part of the IP address. If the IP is not matched with the trusted group then it will not be allowed to enter into the trusted group.

V. CONCLUSION AND FUTURE WORK

Peer-to-Peer networks have previously assumed a fail-stop model for nodes; any node accessible in the network was assumed to correctly follow the protocol. However, if nodes are malicious and conspire with each other, it is possible for a small number of nodes to compromise the overlay and the applications built upon it. This paper has presented the design and analysis of techniques for secure node joining, routing table maintenance, and message forwarding in structured P2P networks. These techniques provide secure routing, which can be combined with existing techniques to construct applications that are robust in the presence of malicious participants.

Future work

Scalability is important for any system; various combinations of this procedure may be used for achieving better result. The accuracy of the detecting may be improved by preprocess the data before analysis. So in this way there is scope to the future enhancements.

REFERENCES

[1]. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In Proc. ACM SIGCOMM'01, San Diego, California, August 2001.
[2]. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. ACM SIGCOMM'01, San Diego, California, August 2001.
[3]. Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. IFIP/ACM Middleware 2001, Heidelberg, Germany, November 2001.
[4]. Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.
[5]. M Chen and J.P Singh. Computing and using reputations for internet ratins. In 3rd ACM conference on Electronic Commerce,2001.
[6]. F. Cornelli, E.Damiani, S.D.C. di Vimercati, S.Paraboschi, and P.Samarati, Choosing reputable servents in aP2P network, in 11th International Conference, 2002.
[7]. C. Dellarocas. Immunizing online reputation reporting systesm against unfacir ratins and discriminatory behavior , ACM , 2000.
[8]. E.Friedman and P.Resnick. The social cost of cheap pseudonyms. Journal of Economics and management Strategy,2001.
[9]. Emil Sit and Robert Morris. Security considerations for peerto-peer distributed hash tables. In Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge,Massachusetts, March 2002.
[10]. Roger Dingledine, Michael J. Freedman, and David Molnar. Accountability measures for peer-to-peer systems. In Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly and Associates, November 2000.
[11]. John R. Douceur. The Sybil attack. In Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02),Cambridge, Massachusetts, March 2002.
[12]. Steve Bellovin. Security aspects of Napster and Gnutella. In 2001 Usenix Annual Technical Conference, Boston, Massachusetts,June 2001.
[13]. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI'99), New Orleans,Louisiana, February 1999.
[14]. Nancy Lynch, Dahlia Malkhi, and David Ratajczak. Atomic data access in content addressable networks. In Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02),Cambridge, Massachusetts, March 2002.