Journal of Computer Science

Original Research Paper

A Secure Chat Application Based on Pure Peer-to-Peer Architecture

[1]Mohamad Afendee Mohamed, [2]Abdullah Muhammed and [3]Mustafa Man

[1]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Kuala Terengganu, Malaysia
[2]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia
[3]School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, Kuala Terengganu, Malaysia

Article history
Received: 08-03-2015
Revised: 01-05-2015
Accepted: 28-05-2015

Corresponding Author:
Mohamad Afendee Mohamed
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Kuala Terengganu, Malaysia
Email: mafendee@unisza.edu.my

Abstract: Chat application is increasingly used as an alternative to older communication technologies such as telephony and telegraph. Equipped with advanced features, people can use it for education, business and socialize. Basic requirement for chatting is an ability to exchange text messages, however, recent releases include support for audio and video communications. For some reasons, peer-to-peer now turned out to be a popular architecture and as such, it becomes a choice for developing chat applications such as Skype. Skype however, makes use of centralized server for user registration, login and buddy list. Indeed, this idea could be disastrous in the event of a compromise. In this study, we proposed a chat application that is based on pure peer-to-peer architecture that totally rid of centralized or third party elements. The system is controlled by the users and its security is autonomously managed by the communicating parties. Each user will have their own database for peer's profiles and communication parties authenticate among each other before exchanging messages. The main contribution of this paper is a state-of-the-art chat application having completely been designed with build in security measures.

Keywords: Peer-to-Peer, Decentralized, Buddy List, Authentication, Encryption

Introduction

Communication is a mean for people to exchange messages. It has started since the beginning of human creation. Distant communication began as early as 1800 century with the introduction of television, telegraph and then telephony. Interestingly enough, telephone communication stands out as the fastest growing technology, from fixed line to mobile wireless, from voice call to data transfer.

The emergence of computer network and telecommunication technologies bears the same objective that is to allow people to communicate. All this while, much efforts has been drawn towards consolidating the device into one and therefore indiscriminate the services.

The principle of communication can generally be categorized into two, client-server and peep-to-peer (Daswani *et al*., 2003; Eberspächer and Schollmeier, 2005). In client-server environment, there is a dedicated server while the rest of other nodes are acting as clients throughout the whole communication. Whereby, in peer-to-peer environment, a node can be either a client or a server depending whether it is a requestor or provider of the service at that specific time.

Client-server technology has been around for some times. Some examples of client-server applications are web access, network time and windows login. Specific to user login, user's credentials are stored in the database at the server side. Anyone who has an access to the server and the database can easily access user information. This creates the idea of single point of failure which may cause a devastated damage in case of breaks in. Another downside of client-server is the resource consumption which always concentrates on the server side. This gives rise to the infrastructure cost for the provider.

Chat application is becoming an important digital communication tools, people use it for business, education and socialize. It offers users an ability to identify online friends, a user then just selects the one

that he/she wants to communicate before begins exchanging messages at will. Basic function of chat application is to be able to exchange a text-based message although today's application is equipped with more advanced features such as audio, video conferencing to the least. The application can be developed based on either client-server or Peer-to-Peer (P2P) architectures. However, P2P now turned out to be the popular architecture as it offers many advantages such as low maintenance cost for server side (considering client-server scenario) and high availability content distribution system. For this reasons, many popular applications like Skype, BitTorent and eMule rely on P2P network.

Being one of the most successful computer-based chat applications, Skype offers many user-friendly functions such as buddy list, video conferencing and file transfer. Skype is built on top of hybrid peer-to-peer architecture (Baset and Schulzrinne, 2006). The terms hybrid is used to refer to some element of client-server embedded within P2P architecture.

By and large, before using Skype, a user needs to register to the centralized Skype server for setting up username and password. It is an easy process and is fully guided by the software itself. Fortunately, registration is only one-time operation. Upon completion, at any time when a user wants to communicate with a friend, he/she is required to authenticate him/herself to the login server for the purpose of self-identification and retrieving buddy list. To make sure the client having the right destination server, Skype hard codes the IP address and port number of the server in its executable. Client side uses this parameter when communicating credentials with the server.

Skype user registration and authentication are considered as client-server like operations. During login procedure, the server side is responsible for validating user account and authenticating peers' profile. For this purpose, the server side maintains a list of existing users in a huge database system. On top of that, the server is also in charge of listing online friends for the newly authenticated user. For once, this centralized storage is susceptible to attacks and it provides a single point of failure for attackers to challenge the security of the system (Vestola, 2010). In case of compromised, the damage can be disastrous. Moreover, in the event of server failure, all communications are halted and user needs to wait for the server side to come back up before being able to communicate again.

In this research, we aim at replacing the idea of centralized server by distributed server resided at every user device. One maintains his/her own peers' credentials and as such no need for centralized server anymore. The proposed chat application is based on a pure P2P architecture within a decentralized environment. This new application can be seen as an alternative to hybrid P2P found in Skype, but with a better security feature for users. This way, we manage to avoid some unnecessary dependencies on the third party and therefore bring up security to one step higher.

Methodology

The whole idea of this proposed application is to avoid a centralized system (registration, login and buddy list) as that found in Skype (Baset and Schulzrinne, 2006; Azab *et al*., 2012). Using Skype, during registration, user profile will be stored in a centralized database and one can use the credential to login at anytime and anywhere as preferred. This certainly provides certain level of robustness although the question arises as to how secure the centralized database can be to prevent from attacks.

Recently, a study on decentralized system was proposed but only for the purposed of improving the buddy list (Kundu, 2012). The idea was about developing a robust index system using distributed hash table for decentralized chat application. An indexing system is responsible for storing IP address and port of all users once they joined the chat. Users initialize their own buddy list by contacting the centralized indexing system once they logged in. When a user A wants to communicate to user B, B will act as a server and authenticate client A. As authentication is one-way, this opens up an opportunity for attackers to masquerade as user B. To cater some of these problems, in our proposed application, we come out with the following principle ideas:

- Pure P2P architecture with no centralized server, peers' profiles is managed locally by user
- User registrations are done among themselves and therefore, multiple registrations are needed for communicating with different users
- User login is done on each peer basis and it follows a two-way authentication protocol
- Message is encrypted prior exchanging between two or more peers

Another important feature of any chat application would be buddy listing. In this application, during initialization process, the system reads through the buddy list from local hash table and automatically determines their (device) availabilities by contacting them based on the IP address and port number of their devices. However, it could also be the case that someone else is on the device, for that we have an authentication procedure which makes use of username and password. Furthermore, message exchange is only allowed after peers have authenticated to each other. Unlike Skype, when user initializes the application, one is prompted for

login and if successfully authenticated, user will be able to see online buddies and start exchanging messages.

We depict the architecture of our proposed application as in Fig. 1. Every peer is equipped with internet or LAN connection. The chat software is installed and it has a simple database system for storing friends' profiles. Considering Peer1, its index table consists of credentials belong to Peer3 and Peer4 but not Peer2 since there is no agreement between Peer1 and Peer2. Although Peer2 exists, Peer1 will not be able to see it via Peer3 and Peer4 since the database is one-to-one and as such the security of peers' credential is well maintained. In this scenario, Peer1 can only initiate communication with Peer3 since Peer4 is not connectable due to broken links although Peer4 may be online at that specific time.

There are some noticeable drawbacks from this architecture and that are, user needs to register *n* number of times in order to warrant communication with *n* different peers and user needs to authenticate *m* (<*n*) number of times in order to start communicating with *m* different peers. However, the advantage is if the database of one peer is compromised, the damage is limited to communications specific with that peer and the rest of communications with another *n*-1 peers are still secured since a user can use different credentials when communicating with different peers. From this argument, we proposed that this idea is very suitable for those who require high level of security for data communication but probably not well suited for user having so many peers as it requires extra efforts to manage the database. As an example, this application can be implemented for the use within organization local area network.

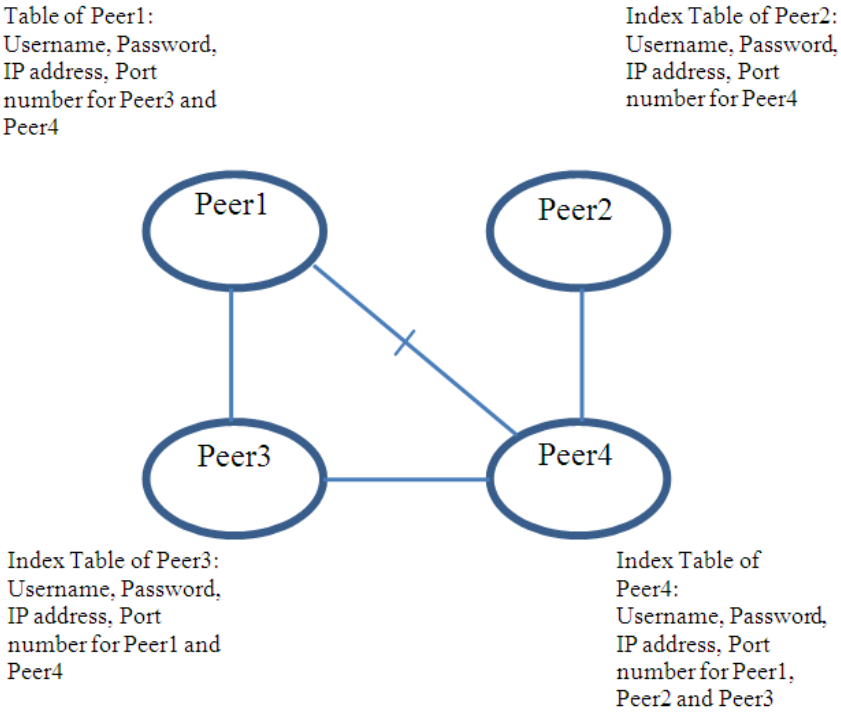

Fig. 1. Proposed architecture

Registration Procedure

In this application, user account registration is left to be done in many ways allowing certain level of flexibility. For examples, a user can visit another user to exchange credentials and manually store into each other database. For distant buddies, both can register to each other from the software and validation can be done via short messages or through email. Successful registration ends up with one's username, hashed password, IP address and port number stored in the other peer's database. Hashing algorithm helps to step up the security of user password in the database. Moreover, IP address and port number is used for the purpose of device identification. Registration between two peers is made discreet; user is encouraged to user different credentials when registering to different peers. This way, it will enhance the security of the application.

Within registration procedure, peers can also agree on secret key for later purpose of message encryption. This secret key can be exchanged together with username and password.

Login Procedure

In order to provide a secure chat, we focus on a simple mutual exclusion in authentication procedure (Li *et al*., 2009; Xu and Li, 2010). Before authenticating to each other, users must have a valid permission to connect, given by a peer, of which authentication will follow. The application captures username, hashed password, Internet Protocol (IP) and port number of a transaction and compares to the one stored in XML database. We have proposed the use of hash function called SHA-256 for authenticating peers (Michail *et al*., 2005). Since the hashed password that is stored in the peer's database, only hashed password needs to be transmitted to that peer.

Figure 2 shows an authentication protocol that we have implemented in this application. As an alternative, user authentication can also be seen as connection establishment phase. In this scenario, Peer1 wishes to communicate with Peer2. This protocol is responsible for asking and granting/rejecting permission from both/to parties. Initially, Peer1 send a request to chat to Peer2 and wait for replies. Peer2 in return have two choices of responses, accept or reject that lead to an end. In case of accept response, Peer1 is required to send their user profiles (username and hashed password) to Peer2. Peer2 will authenticates this profile by comparing to the one exists in its database. In response Peer2 sends success or failure authentication message that lead to an end. Upon receiving authentication successful, Peer1 will ask for Peer2 user profiles. Peer2 sends the user profiles where Peer1 will perform the authentication. Upon successful authentication for both sides, they are ready to exchange messages. This additional procedure is very important to ensure that we are chatting with the correct user and not with someone else that used the computer temporarily.

Data Exchange

Once peers authentication completed, they can start exchanging messages. For security reason, message is encrypted using AES cryptosystem (Bardis and Ntaikos, 2008) employing secret key (Al-Riyami and Paterson, 2003) that we have agreed at the registration phase. The session is valid as long as user does not log out from the chat application. Otherwise, peers need to authenticate one another again.

Implementation

The implementation is subdivided into five main areas that are the Graphical User Interface (GUI), socket programming, user profile's database, user authentication using hash function and message encryption using cryptographic algorithm. Figure 3 shows a simple framework that serves as guidance during an implementation phase. From this framework, we divide our architecture using four modules.

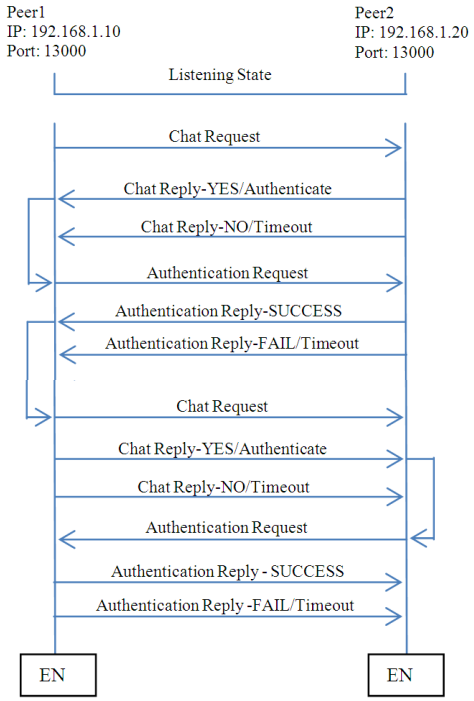

Fig. 2. Authentication protocol

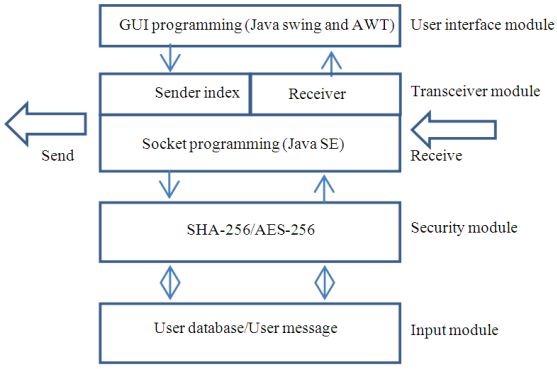

Fig. 3. Proposed framework

The whole project was developed based on Java language. The development of GUI uses Swing and AWT technologies. For peers' communication, simple socket programming was used. For the purpose of user authentication, SHA-256 is used for hashing user password. Peers' IP address and port number are bind together and stored in XML database together with respective username and password.

In authentication procedure, it involved verifying the validity of at least one form of identification. In this research, we have implemented SHA-256 in authentication phase. When a user creates a password, peers system hashes the input using SHA-256 in to hexadecimal value. When a user performs authentication during login procedure, peer system compares the password to the one stored earlier.

To maintain the secrecy of communication, all messages will be encrypted using AES cryptosystem employing 256 bits of key prior transmitting to the other peer. This is the standard key size used by most of the current applications and is acceptable for medium secured application. The use of longer key size is possible but at the expense of some extra computational power.

Results

In this section we show the finish product in respect to the functionalities that we have discussed earlier. The product has been tested in local area network. Next discussion centers on the communication between two peers, namely Peer1 and Peer2. For simplicity, we assume both peers have registered to one another.

Figure 4 shows a screenshot at Peer1 in listening mode. In this case, Peer1 is initiating a connection to Peer2 (192.168.1.20:13000).

At Peer2, the listener will create a pop-up window informing someone (192.168.1.10) has initiated a connection and ask if the user wishes to allow or reject the communication as in Figure 5.

On receiving communication accepted, Peer1 is required to enter the username and password. By clicking the button 'login' in Figure 6, the username and a hashed password will be transmitted to Peer2.

Peer2 compares the received username and hashed password with the one store in its database. Upon successful validation, acknowledgement will be sent to Peer1. Peer1 will in turn request for Peer2 profile. The same procedure happens until both sides have authenticated to each other.

After authentication successfully completed, users can start exchanging messages. Figure 7 shows the two peers successfully exchanged simple messages.

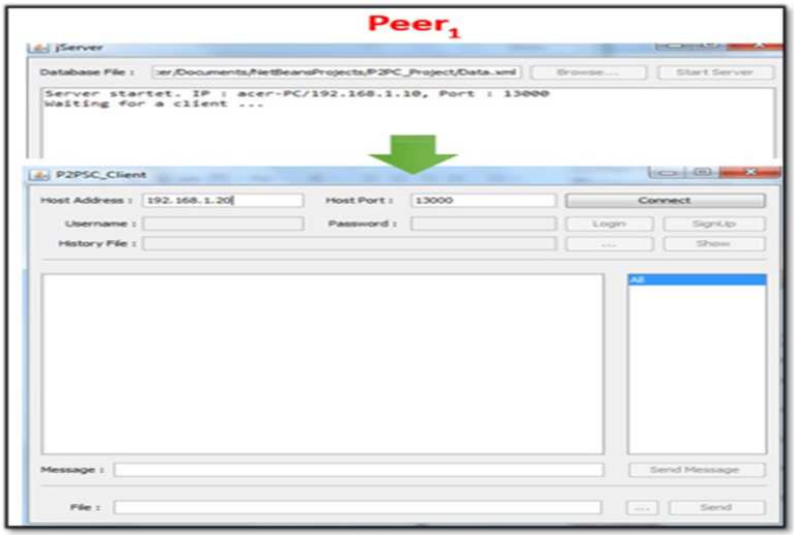

Fig. 4. Initiating connection

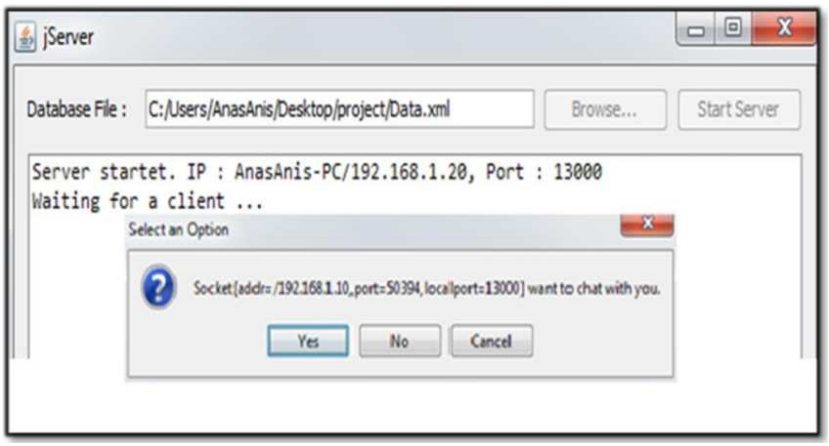

Fig. 5. Accepting connection

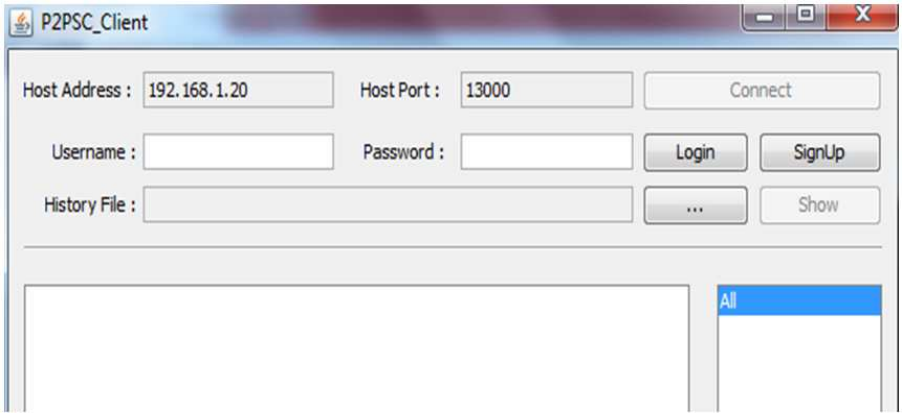

Fig. 6. Profile provision

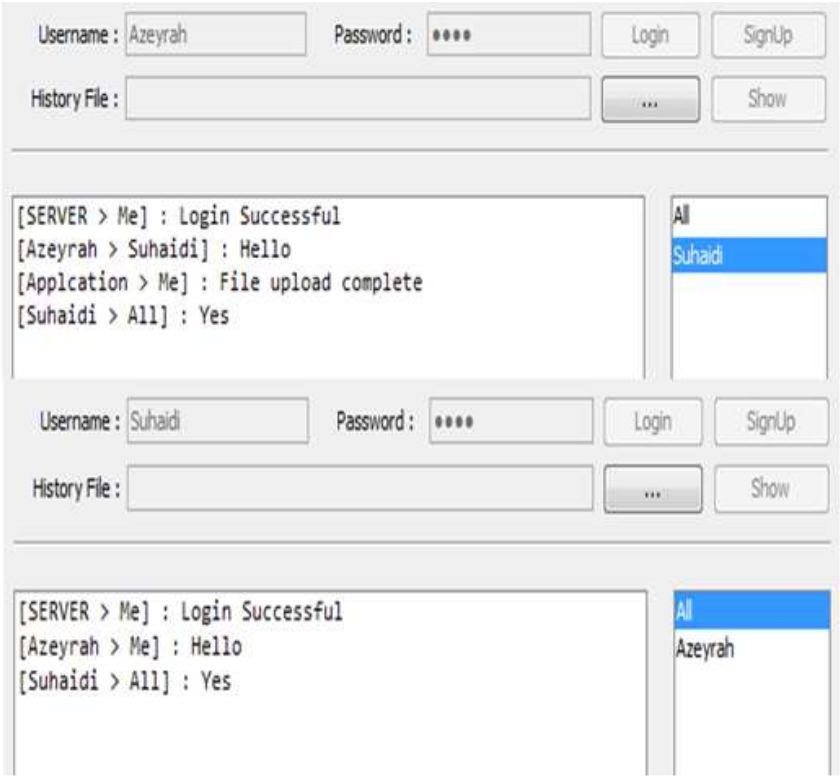

Fig. 7. Message exchange

Conclusion

Analysis has been conducted to investigate decentralized P2P architecture. In this study, we developed a secure chat application that can be implemented in decentralized environment. The main objective was to have a self-secured individual user database without any involvement of third party. Security element was built within user registration, user login, message exchange and database storage.

Possible future works that need to be highlighted includes an update of security tools to the latest version, a variable file size for the purpose of file transfer and incorporating functionalities like voice communication or live chatting.

Moreover, it would be beneficial if we can have a trusted group feature where the member only need a single username and password when communicating with any user within the group. In making the application more secure, we should also offer different authentication and encryption algorithms to be chosen by users. Whenever possible, key size should be allowed to be negotiated manually by the users.

With large migration from PC based to mobile based application, we could also use phone number for device identity as that found in *Whatapps* application.

Acknowledgement

The authors wish to thank anonymous reviewers for their valuable comments and insights to improve the quality of this paper.

Funding Information

The authors have not received any funding corresponding to this research.

Author's Contributions

All authors equally contributed in this work.

Ethics

This material in this article is original and has not been published elsewhere; all authors declare that there is no conflicts of interest.

References

Al-Riyami, S.S. and K.G. Paterson, 2003. Certificateless public key cryptography. Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security, Nov. 30-Dec. 4, Springer Berlin Heidelberg, Taiwan, pp: 452-473. DOI: 10.1007/978-3-540-40061-5_29

Azab, A., P. Watters and R. Layton, 2012. Characterising Network Traffic for Skype Forensics. Proceedings of the 3rd Cybercrime and Trustworthy Computing Workshop, Oct. 29-30, IEEE Xplore Press, Ballarat, pp: 19-27. DOI: 10.1109/CTC.2012.14

Bardis, N.G. and K. Ntaikos, 2008. Design of a secure chat application based on AES cryptographic algorithm and key management. Proceedings of the 10th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems, (TIS' 08), ACM, USA, pp: 486-491.

Baset, S.A. and H.G. Schulzrinne, 2006. An analysis of the skype peer-to-peer internet telephony protocol. Proceedings of the 25th IEEE International Conference on Computer Communications, (CCC' 06), IEEE Xplore Press, Spain, pp: 1-11. DOI: 10.1109/INFOCOM.2006.312

Daswani, N., H. Garcia-Molina and B. Yang, 2003. Open problems in data-sharing peer-to-peer systems. Proceedings of the 9th International Conference Siena, Jan.8-10, Springer, Italy, pp: 1-15. DOI: 10.1007/3-540-36285-1_1

Eberspächer, J. and R. Schollmeier, 2005. First and second generation of peer-to-peer systems. Peer-to-Peer Syst. Appl., 3485: 35-56. DOI: 10.1007/11530657_5

Kundu, A., 2012. Decentralised indexed based peer to peer chat system. Proceedings of the International Conference on Informatics, Electronics and Vision, May 18-19, Dhaka, IEEE Xplore Press, pp: 416-419. DOI: 10.1109/ICIEV.2012.6317378

Li, Z., X. Xu, L. Shi, J. Liu and C. Liang, 2009. Authentication in peer-to-peer network: Survey and research directions. Proceedings of the 3rd International Conference on Network and System Security, Oct. 19-21, IEEE Xplore Press, Gold Coast, pp: 115-122. DOI: 10.1109/NSS.2009.30

Michail, H., A. Milidonis, A. Kakarountas and C. Goutis, 2005. Novel high throughput implementation of SHA-256 hash function through pre-computation technique. Proceedings of the 12th IEEE International Conference on Electronics, Circuits and Systems, Dec. 11-14, IEEE Xplore Press, Gammarth, pp: 1-4. DOI: 10.1109/ICECS.2005.4633433

Vestola, M., 2010. Security issues in structured P2P overlay networks. Helsinki University of Technology.

Xu, Z.B. and Z.W. Li, 2010. Efficient and secure certificateless authentication and key agreement protocol for hybrid P2P Network. Proceedings of the 2nd IEEE International Conference on Information Management and Engineering, Apr. 16-18, IEEE Xplore Press, Chengdu, pp: 272-276. DOI: 10.1109/ICIME.2010.5477831