Blocking Misbehaving Users in Anonymizing Networks Using Client Puzzle

Ms. Malarvizhi.K[1], Ms. M. Anitha[2]

[1]*Assistant Professor, Department of Information Technology, Sri Ramakrishna Institute of Technology, Coimbatore*

[2]*Assistant Professor, Department of Computer Science and Engineering, Sri Krishna College of Engineering & Technology, Coimbatore*

***Abstract*: - Anonymous network such as Tor, is used to protect user's identity and it hides client's IP address from server through series of routing. The users can authenticate services anonymously by using some credential systems. Since anonymity can give users the license to misbehave, some variants allow the selective deanonymization (or linking) of misbehaving users upon a complaint to a trusted third party. The misbehaving users are collected in the Blacklist. To address this problem, Nymble with client puzzle system is proposed, a system which consist of the following advantages, (1) honest users remain anonymous and their requests unlinkable; (2) a server can complain about a particular anonymous user and gain the ability to blacklist the user for future connections; (3) this blacklisted user's access remain anonymous before the server complains; and (4) users are aware of their blacklist status before accessing a service. In proposed system, Nymble with client puzzle eliminate the need for Pseudonym Manager and the users can receive nymbles directly from the Nymble Manager and by using nymbles, user is connected with server for accessing the services.**

***Keywords:* - Anonymous network, blacklist, Authentication, Misbehaving**

I. INTRODUCTION

Network security starts with authenticating the user, commonly with a username and a password. It involves the authorization of access to data in a network, which is controlled by the network administrator. Users choose or assigned with an ID and password or other authenticating information that allows them access to information and programs within their authority. It consists of provisions and polices adopted by the network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of the computer network and network-accessible resources. The term anonymous is used to describe both (wired/wireless) kinds of network because it is difficult if not impossible to determine whether a node that sends a message originated the message or is simply forwarding it on behalf of another node.

Tor [3] is a system intended to enable online anonymity. Original data is encrypted and re-encrypted multiple times, then sent through successive tor relays, each one of which decrypts a "layer" of encryption before passing it on to the next relay and ultimately the destination. This reduces the possibility of the original data being unscrambled or understood in transit. Some of the networks commonly referred to as "anonymous P2P" are truly anonymous, in the sense that network nodes carry no identifiers. Others are actually pseudonymous: instead of being identified by their IP addresses, nodes are identified by pseudonyms such as cryptographic keys.

1.1 Cryptographic Hash Functions

A cryptographic hash function [4] is, an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an (accidental or intentional) change to the data will (with very high probability) change the hash value. The data to be encoded is often called the "message," and the hash value is sometimes called the message digest or simply digests**.** Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. Cryptographic hash functions are typically used to compute a message digest when making a digital signature. Instead of encrypting the whole message with the secret key, only the message digest is encrypted [5]. A hash function compresses the bits of a message to a fixed-size hash value in a way that distributes the possible messages evenly among the possible hash values. A cryptographic hash function does this in a way that makes it extremely difficult to come up with a message that would hash to a particular hash value.

Cryptographic hash functions typically produce hash values of 128 or more bits. This number is vastly larger than the number of different messages likely to ever be exchanged in the world. The message digest algorithm used in PGP is the MD5 [4] Message Digest Algorithm, placed in the public domain by RSA Data Security. The level of security provided by MD5 should be sufficient for implementing very high security hybrid digital signature schemes based on MD5 and public-key cryptosystems. Some commonly used cryptographic hash functions include MD5 and SHA-1, though many others also exist.

1.2 Message Authentication (MA)

MA is the process of digitally signing a message provides proof of the authenticity of the document or information object, with far greater certainty and precision than paper signatures. Because the verification process comparing the digital representation (hash) of the message or document made at signing with the one created during the verification process discloses whether the message is the same as when signed. MACs are a security code that is typed in by the user of a computer to access accounts or portals. This code is attached to the message or request sent by the user. Message authentication codes (MACs) attached to the message must be recognized by the receiving system in order to grant the user access.

In cryptography, HMAC [4] (Hash-based Message Authentication Code) is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message. Any cryptographic hash function, such as MD5 or SHA-1 [5], may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA1 accordingly.

1.3 Cryptographic Symmetric-Key

Semantic relatedness between two terms can be computed based on highest value path connecting any pair of the terms [3]. Symmetric-key [5] cryptography is sometimes called secret-key cryptography. The most popular symmetric-key system is the Data Encryption Standard (DES). Symmetric-key algorithms are a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. Symmetric key encryption uses same key, called secret key, for both encryption and decryption. Users exchanging data keep this key to themselves. Message encrypted with a secret key can be decrypted only with the same secret key. Strength of the symmetric key encryption depends on the size of the key used. Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way. Symmetric-key encryption can use either stream cipher or block ciphers. Stream ciphers encrypt the digits (typically bits) of a message one at a time. Block ciphers take a number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size. Blocks of 64 bits have been commonly used.

II. LITERATURE SURVEY

2.1 Nymble with Pseudonym Manager

Patrick P. Tsang et al.[1] introduced Anonymous credential system called Nymble with Pseudonym Manager.A system in which server can blacklist individual misbehaving user, thereby efficiently blocking user without compromising their anonymity while maintaining their privacy. Drawback of this anonymous credential system is time complexity.

2.2 Client Puzzles

Juels et al. [2] proposed a Client Puzzles protocol, a measure for connection depletion attacks. There is no general framework for client puzzle schemes. The Repeated Squaring Puzzle is the best client puzzle and also has low time complexity among, Hash-based Puzzle, Discrete Logarithm-based Puzzle, Subset Sum-based Puzzle. When a server comes under attack, it distributes small cryptography puzzle to client making service request and the client must solve its puzzle correctly.

2.3 The Second-Generation Onion Router

Dingledine et al. [3] introduced Tor, a circuit-based low-latency anonymous communication service. It works on real world Internet requiring no special privileges or kernel modifications, little synchronization or coordination between nodes and providing a reasonable tradeoff between anonymity, usability and efficiency. Tor frustrates attackers from linking communication partners or from linking multiple communications to or from a single user.

2.4 Keying Hash Functions for Message Authentication

Bellare et al. [4] presented NMAC and HMAC, message authentication schemes based on cryptographic hash function. This scheme is efficient and also retains almost all the security of the underlying hash function. They utilize any cryptographic hash function of the iterative type and enjoy security and practicality features.

2.5 Symmetric Encryption

Desai et al. [5] proposed schemes for symmetric encryption in concrete security framework. They have been presented with four different notions of security against chosegtn plaintext attack. The concrete complexity of reductions is analyzed among them using Cipher Block Chaining and Counter Mode, provided they are given with both upper and lower bound and obtains tight relation.

2.6 NYM: Practical Pseudonymity for Anonymous Networks

Holt et al. [6] introduced Nym, a simple way to allow pseudonymous access to Internet services via anonymizing networks like Tor, without losing the ability to limit vandalism using Blocking owners of offending IP or email addresses. To create a pseudonymity system with extremely low barriers to adoption, the Nym used application of blind signature. Nym provide a complete solution to be able to deploy with bare minimum amount of time.

III. METHODOLOGY

3.1 Nymble with Client Puzzle System

The Client Puzzle scheme contains several methods such as

discrete logarithmic-based Puzzle, Subnet Sum-based Puzzle, and Repeated Squaring Puzzle [2] etc. In the Proposed system, Repeated Squaring Puzzle of Client Puzzle is used. Fig.1 describes that the Nymble with Client Puzzle Architecture. Repeated Squaring Puzzle is performed between Nymble Manager and User. Using Repeated Squaring Puzzle in Nymble Manager, User registration is performed and ticket is generated. Using the generated ticket, the user can communicate with the server for accessing server's services. In this system, three types of Misbehaving activities are identified such as Password authentication, rewriting the files and downloading files.

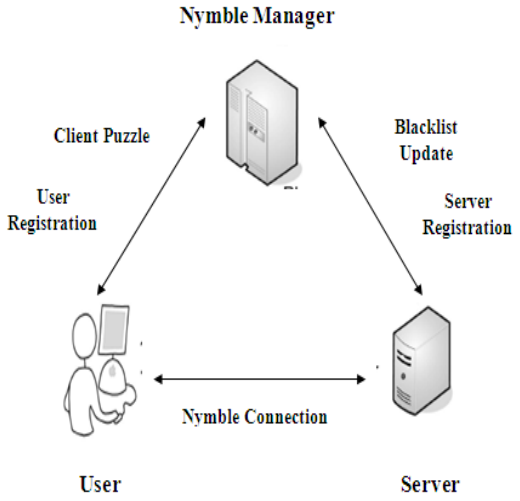

Figure 1. System Architecture

Fig.2 describes that, when a user request the Nymble Manager for user registration, Nymble Manager registers the users. Nymble Manager sends puzzle to the user, for ticket generation. The users solve the puzzle and send backs the solution to the Nymble Manager. The puzzles are verified by the Nymble Manager. If the puzzles are correct then the valid ticket is generated and given to user. If the puzzle is incorrect then the user is invalid. The user sends the ticket to the server for verification. The server checks the misbehaving of the user. If misbehaving is noticed, the server doesn't provide the services to the user. If the user is perfect without any misbehaving, the server provides the services. In this system user can acquire an ordered collection of nymbles, to connect user with server. Without additional information, these nymbles are computationally hard to links. Users access the services by using the stream of nymbles. If the misbehaving of user is noticed by the server, then the services are disconnected immediately.

IV. System Design

4.1.1 Server and User Registration

Server Registration:

The Nymble Manager [1],[6] participates in the server registration. In this process server id (*sid*) is sent from Server to Nymble Manager and using *sid* Nymble Manager registers the server. Server registers it. Server can be registered at most once and NM makes sure that the server has already registered or not.

User Registration:

The Nymble Manager participates in the user registration. A user with identity (*uid)* registers with the NM. NM checks if the user is allowed to register. NM infers the registering user's IP address and makes sure that the IP address has been already registered or not. By using *uid*, NM generates the pseudonym for user registration.

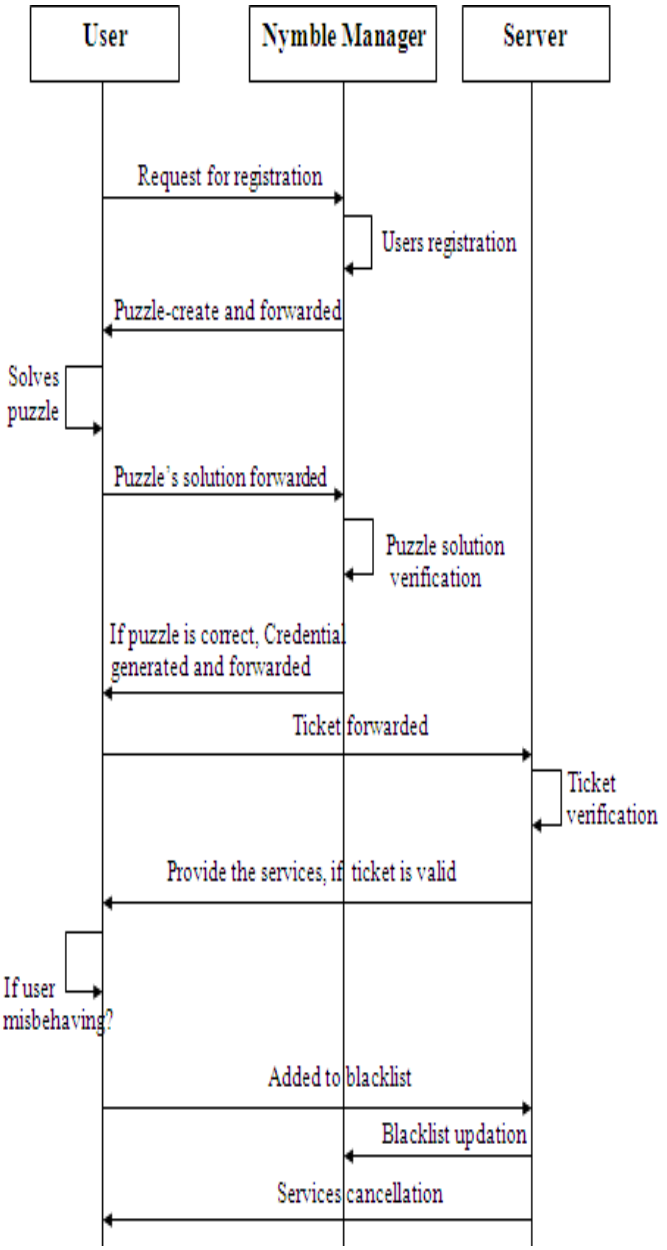

Figure 2. Overall Process of Nymble with Client Puzzle system

4.1.2 Client Puzzles Generation

In Client Puzzles generation [2] NM plays the major role. Repeated Squaring Puzzle is used by the NM. NM generates the puzzles and which are sent to the user. User solve the puzzle and the solution is sent to NM.NM verifies the solution, if solution is correct, NM generates the credential for the user. If the solution for the puzzle is incorrect, NM does not allow the user for further process.

The steps followed for Client Puzzle process are as follows,

Step1: Puzzle Generation

This algorithm selects two random large primes p, q and creates n = p * q out of this. Then it chooses a generator g out of the range [2; n - 1] and outputs the puzzle puz = {n,g} and info = {g,p,q}.

Step2: Puzzle solution

$$Sol=g^{2^{\wedge}l} \bmod n.$$

Step3: Puzzle verification

$$Sol=g^{2^{\wedge}l \bmod (n)} \bmod n.$$

4.1.3 Nymble Tickets and Credentials Generation

The NM generates pseudonyms to users. A pseudonym[1] *pnym* has two components *nym* and *macs*. *nym* is a pseudorandom mapping of the user's identity (IP address).

Pseudonym generation:

Input: User Id-> *uid*

Specific time duration-> *w*

Process:

Step1: Extract *nymKeyP* and *macKeyNP* from Pseudonym state.

Step2: Using *nymKeyP* and *uid* || *w* as input, message authentication code computation algorithm generates *nym*.

Step3: Using *macKeyNP* and *uid* || *w* as input, message authentication code computation algorithm generates *mac*.

Output: Pnym->(nym,mac)

nymKeyP, macKeyNP are generated secret keys.

By using this *pnym* NM generates the credential. A credential contains all the nymble tickets of a particular user present in the server. For nymble ticket generation *f* and *g* are two distinct cryptographic hash functions used. A ticket contains a nymble specific to a server, time period, and linkability window. This credential is sent to the user for the server connection.

Credential generation:

Input: Pnym->(nym,mac)

Server Id->*sid*

Specific time duration-> *w*

Process:

Step1: Extract *macKeyNS, macKeyN, seedKeyN*, and *encKeyN* from Nymble State.

Step2: p*nym* || *sid* || *w* and s*eedKeyN* are used as input for message authentication code computation to generate *mac2* which is given as input for *seed evaluation function(f) which generates seed0.*

Step3: resulting seed0 is given as input for *nymble evaluation function (g)* which generates *nymble.*

Step4: for *t* from 1 to L do

- *seedt=f(seedt-1)*
- *nymblet=g(seedt)*
- Using *nymble* || *seedt* and *encKeyN* as input for encryption function *ctxtt* is generated.
- *tickett=sid* || *t* || *w* || *nymblet* || *ctxtt*
- Using *tickett* and *macKeyN* as input, message authentication code computation algorithm generates *macNt.*
- Using *tickett* || *macNt* and *macKeyNS* as input, message authentication code computation algorithm generates *macNSt.*
- *Tickets[t]=(t, nymblet , ctxtt , macNt, macNSt)*

Output: cred->(nymble , tickets)

macKeyNS, macKeyN, seedKeyN, encKeyN are generated secret keys

4.1.4 Ticket Verification

For ticket verification user sends the ticket and *RmacNS* to sever. Server generates secret key named $macKey_{ns}$. By using this secret key server generates *macNST* and compare *macNST* with *RmacNS.* If both are not equal then server needs a nymble manager for verification, if both are equal then the user will be provided with a valid ticket for access of service.

Server side ticket verification:

Input: Ticket-> ticket

Received secret key-> RmacNS

Process:

Step1: Extract *sid* and *macKeyNS* from Server state.

Step2: Content = ticket || mac_N

Step3: Using content and *macKeyNS* as input, message authentication code computation algorithm generates *macNST.*

Step4: If *RmacNS* = *macNST*

Return *true*

else

false

Output: true or *false*

NM extract ticket and *RmacNS* from sever. NM generates secret key called *macKeyN*. By using this secret key NM generates *macNST* and compare *macNST* with *RmacNS*. If both are equal then NM sends 'true' to the server and server send the user that the ticket is valid for access of service.

Nymble side ticket verification:

Input: Ticket-> ticket

Received secret key-> RmacNS

Process:

Step1: Extract *macKeyN* from Nymble state.

Step2: Using ticket and *macKeyN* as input, message authentication code computation algorithm generates *macNST.*

Step3: If *RmacNS* = *macNST*

Return *true*

Else

false

Output: true or *false*

macKeyN is generated secret key.

4.1.5 Blacklist Verification and Updation

NM and Server participates in blacklist verification. Server gets request from the user and sends the *blist* to NM for generating the *cert, seed* and it is sent back to the server. The Server sends *cert* and *blist* to the user for verification, i.e., if the user is already in blacklist or not.

If user is not in the blacklist then the user can access the services. The user currently in process if misbehaves then the user is automatically added to the blacklist and service is immediately disconnected.

Generation of cert:

Input: Blacklist (db)->blist

Process:

Step1: Extract sid from Server state and macKeyn, signKeyn from Nymble state.

Step2: Content = sid || t || w || blist.

Step3: Using content and *macKeyN* as input, message authentication code computation algorithm generates *mac.*

Step4: Using content and *signKeyN* as input, message authentication code computation algorithm generates *sig.*

Step5: cert-> (mac, sig)

Output: cert

macKeyn, signKeyn are generated secret keys.

Generation of seed:

Input: Specific time duration-*>t*

Blacklist(db)-> *blist*

Process:

Step1: Extract *decKeyN* from Nymble state.

Step2: Using *ctxt* and *decKeyN* as input, message authentication code computation algorithm generates *nymbleseed.*

Step3: If *blist* contains *nymbleseed*

Return *nymbleseed*

Else

Nymbleseed = fconvert(*nymbleseed*)

Output: nymbleseed

decKeyN is generated secret key.

Blacklist Update:

Input: Received secret key-> *RmacNS*

Blacklist (db)-> *blist*

Process:

Step1: Extract *blist* from sever state.

Step2: If *blist* contains *RmacNS*

Return blacklist present

Else

Not present

Output: Blacklist or not.

V. Experimental Results

5.1 Implementation of Client Puzzle System

Web pages are taken as input. In Nymble with Client Puzzle System is the user can only read the file. Apart from reading the file, if the user writes or downloads the file then the users are considered as misbehaving users and the users can be updated in blacklist.

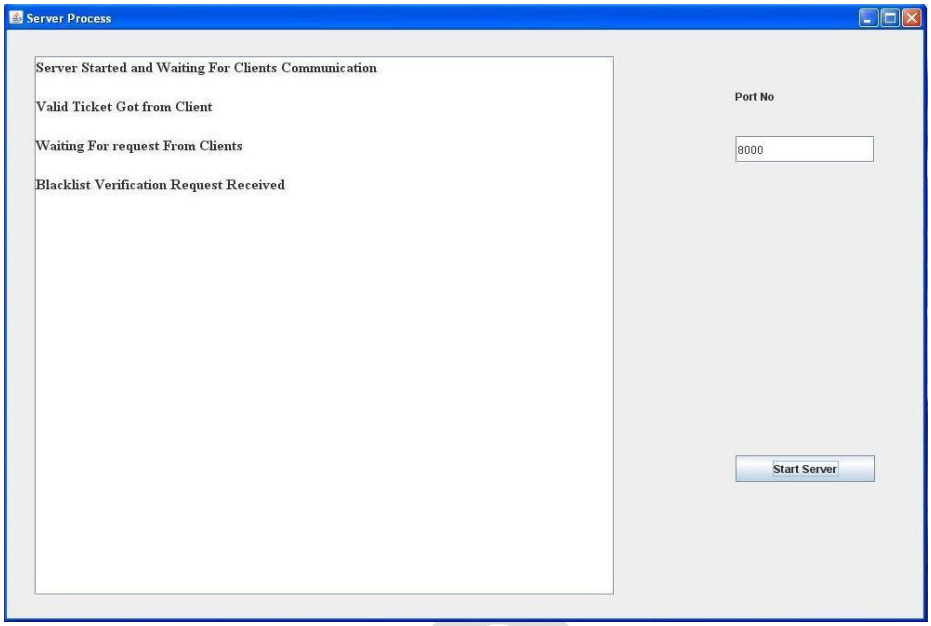

Figure 3.Server Process.

Fig.3 shows users request to the server for its validation, whether they are having valid tickets and verify they are presents in blacklist or not.

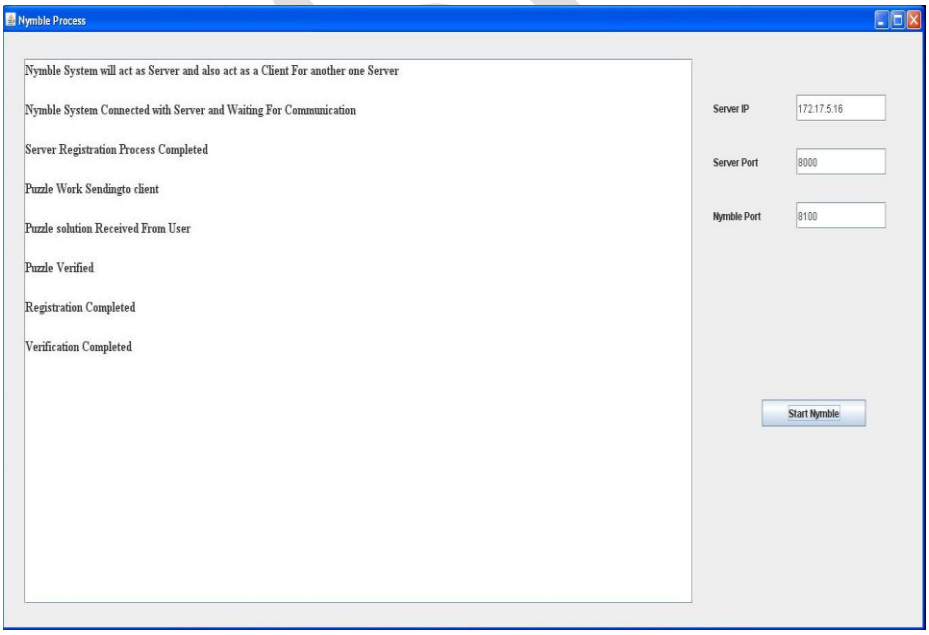

Figure 4.Nymble Process.

Fig.4 shows that user and server registration .When users is connected with NM it generates puzzles and send it to the users. The users solve the puzzles and send it back to the NM, where it checks whether the solution of puzzles are correct or not. If the puzzles are correct then user's registration and verification are completed.

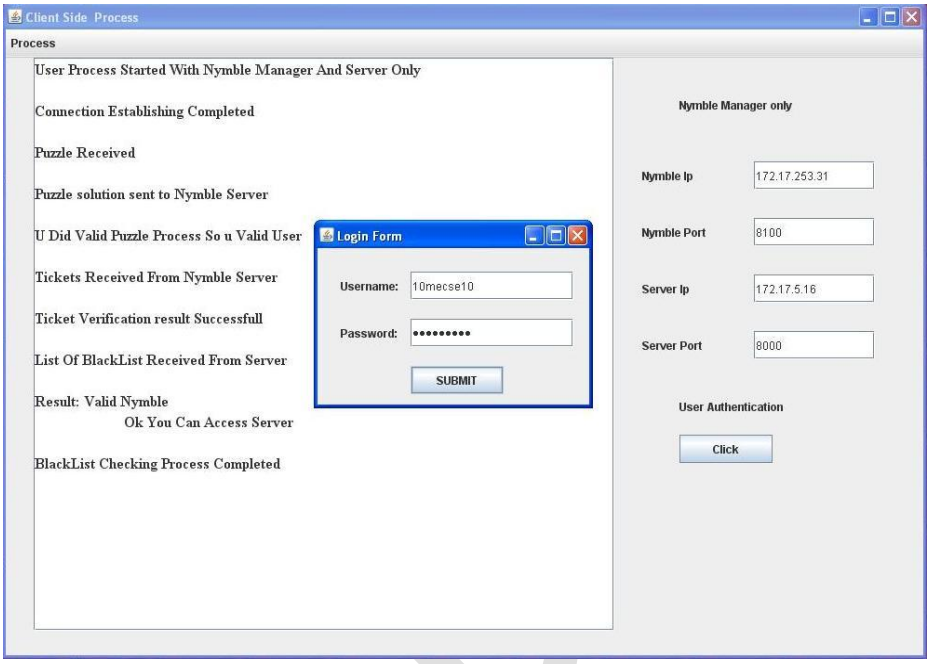

Figure 5.Client Process-Validation of User Authentication.

Fig.5 shows the validation of user authentication, where it checks whether the user is present in blacklist or not. If the user is valid, user authentication procedure is met.

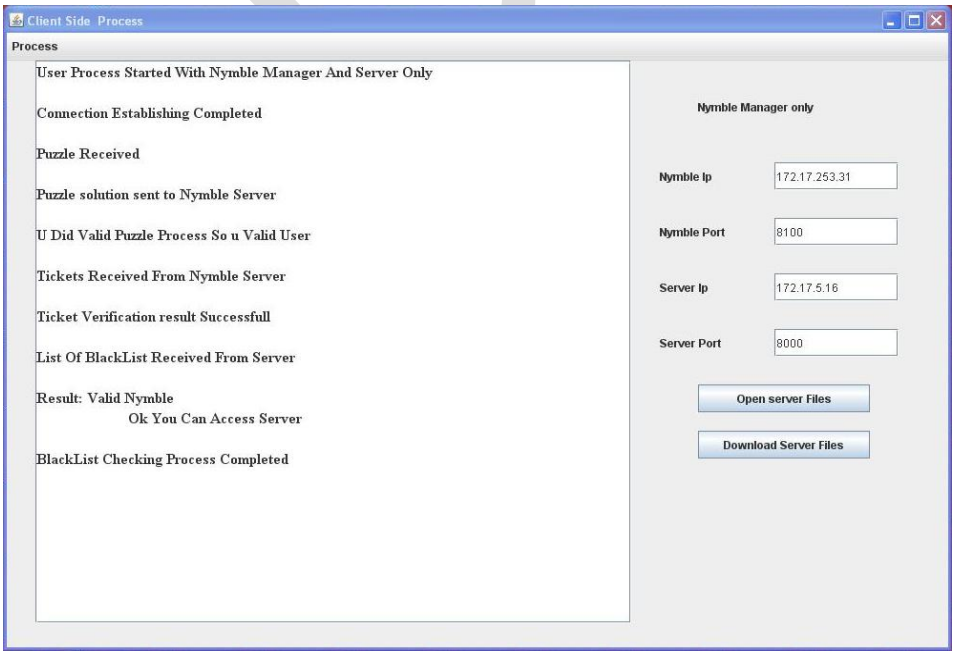

Figure 6.Client Process-Services of Server.

Fig.6 describes the services provided by the Server. If the user name and password are correct then the services are provided to users.

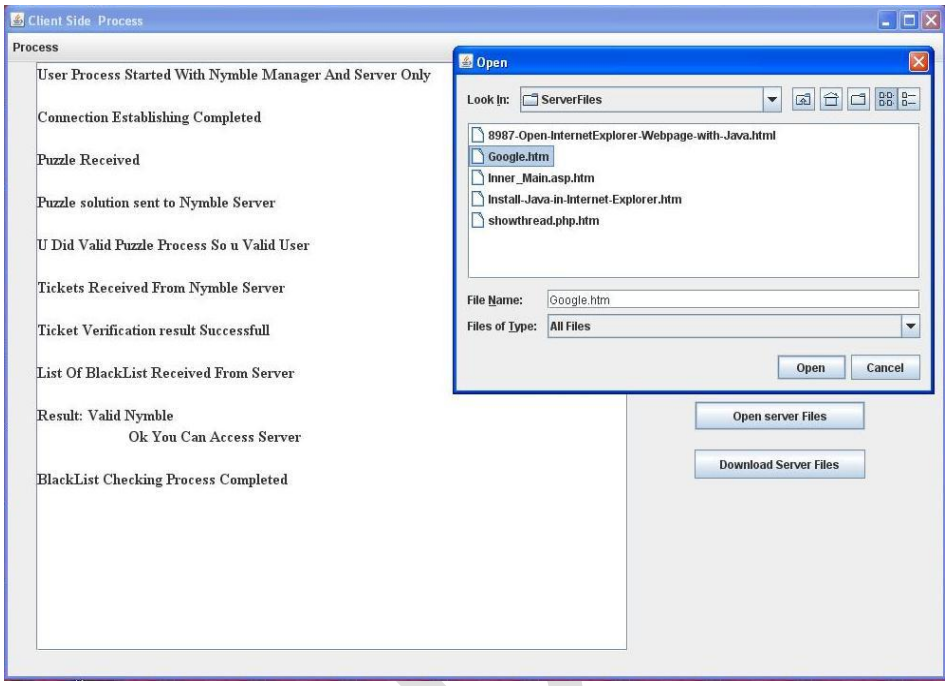

Figure 7.Client Process-Open Server Files.

Fig.7 shows the processes of open server files. When a selected webpage is read, then it is considered that the user is not misbehaving. But if the file is modified (write) then the user is misbehaving and they are blacklisted.

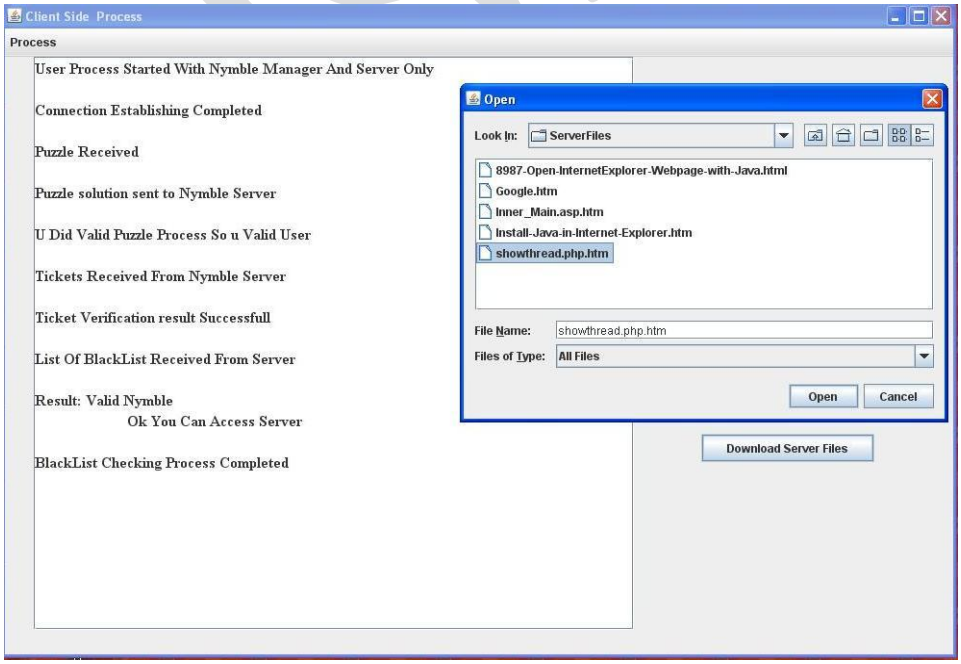

Figure 8.Client Process-Download server files.

Fig.8 describes the processes of downloading a server files. When a selected webpage is downloaded then the user is misbehaving and they are blacklisted.

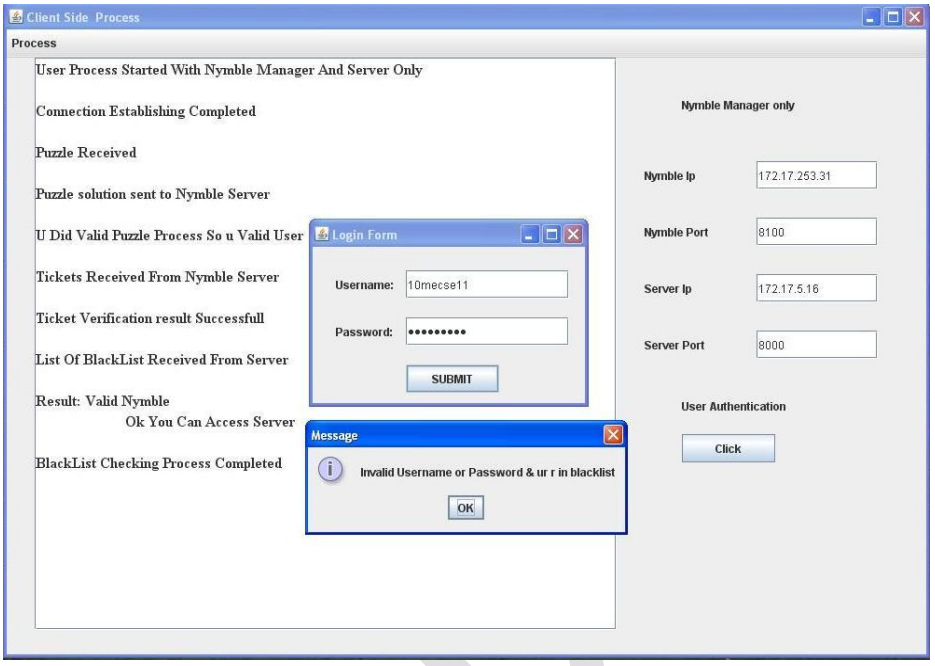

Figure 9.Client Process-Invalid User Authentication.

Fig.9 shows the invalid user authentication. If the user name and password are correct then the services are provided. If user name or password is incorrect more than three times then the user is misbehaving and they are blacklisted.

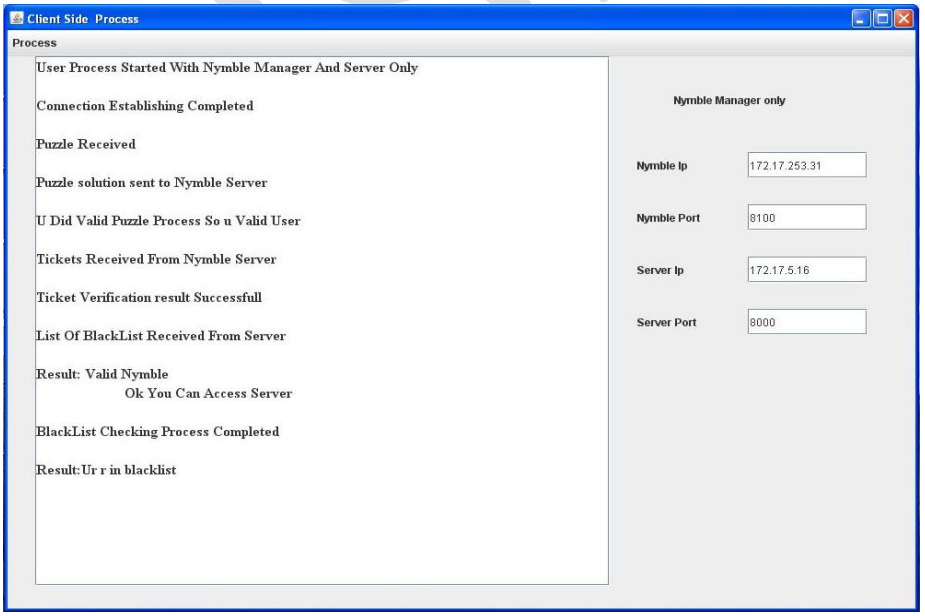

Figure 10.Client Process-Valid User Misbehaved.

Fig.10 shows when a user misbehaves by incorrect user name or password, write or download a server file then their services are disabled and they are included in blacklist.

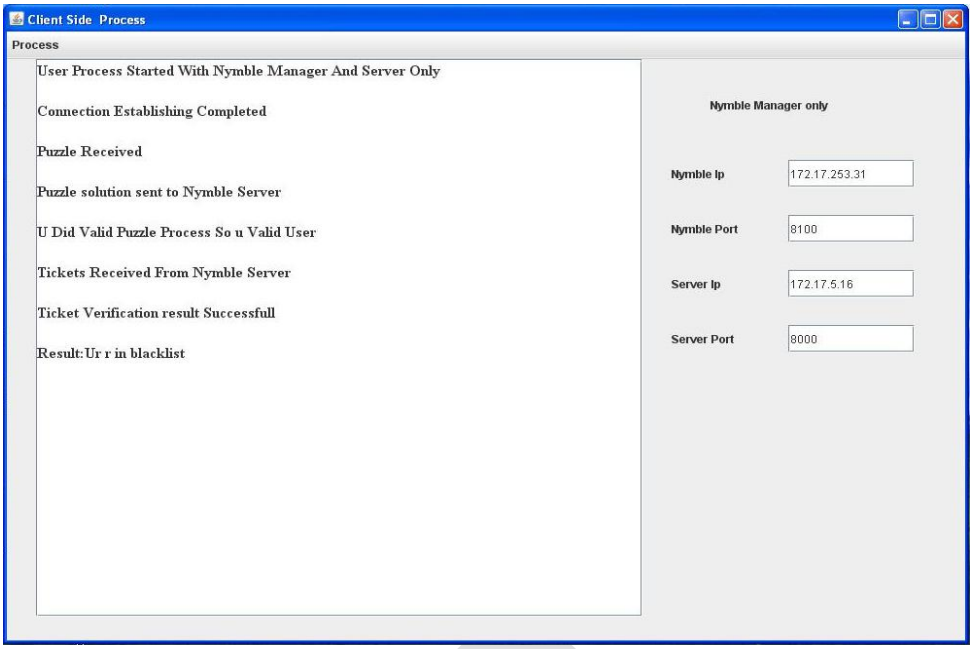

Figure 11.Client Process-Blacklisted User.

Fig.11 describes whether user is blacklisted or not. If blacklisted, user tries to access, they are not allowed to access the service due to above reason.

5.2 Performance Evaluation

In existing system, Pseudonym Manager [1] is used for generating pseudonym. Nymble Manager generates nymble using the pseudonym. In proposed system, user solves the puzzle and by using the user id Nymble manager directly generates the nymble. By comparing with the existing system, proposed system is more advantageous and it consumes less time. Fig.12 shows comparison between existing and proposed nymble generation.

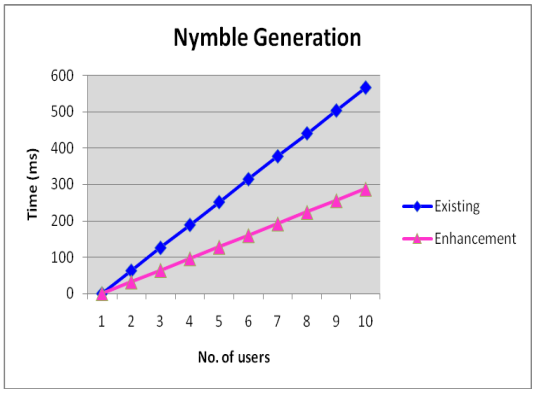

Figure 12.Comparison of Nymble Generation.

After nymble generation, ticket verification is performed in server side. By comparing proposed system with existing system, it consumes less time for ticket verification. Fig.13 shows comparison between existing and proposed ticket verification.

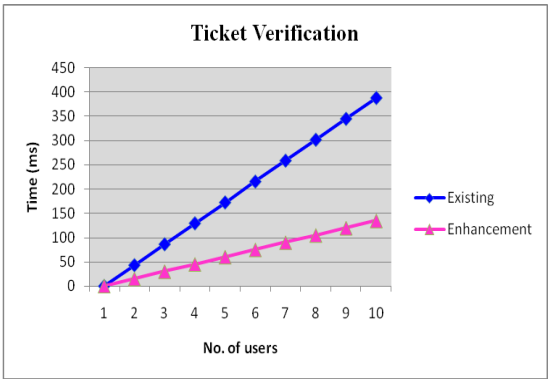

Figure 13.Comparison of Ticket Verification.

Initially, user performs the blacklist verification and then server service is being accessed. By comparing blacklist verification of both systems, proposed system consumes less time. Fig.14 shows comparison between existing and proposed blacklist verification.

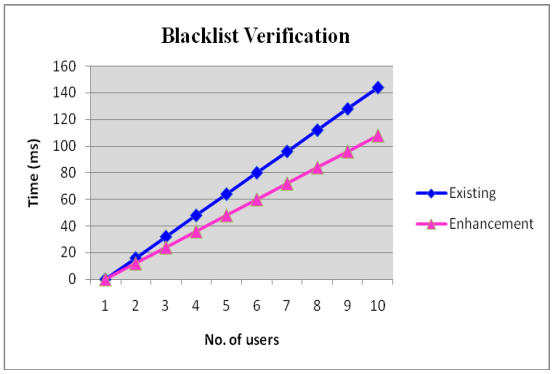

Figure 14.Comparison of Blacklist Verification.

Fig.15 shows the overall performance of Nymble with Client Puzzle system.

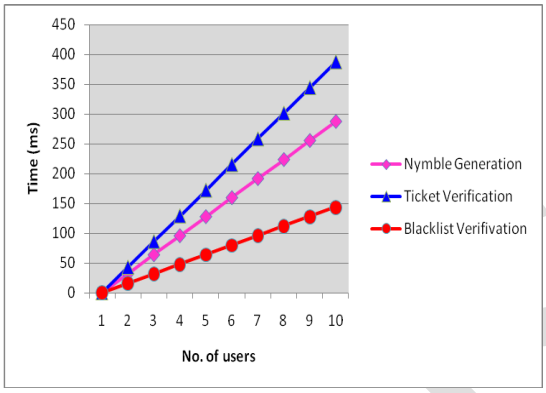

Figure 15.Overall Performance.

VI. CONCLUSION & FUTURE WORK

A comprehensive credential system called Nymble with Client Puzzles, which provides the properties of anonymous authentication, backward unlinkability, subjective blacklisting, and fast authentication speeds, rate-limited anonymous connections are proposed. In Nymble, servers can therefore blacklist anonymous users without the knowledge of their IP addresses while allowing behaving users to connect anonymously. In addition, the system consumes less amount of time and space complexity is much reduced when compared to existing systems.

In Future Nymble is generated using advanced technique HMAC-SHA256 [4] and it is stronger against an attack which is more secure than HMAC-MD5 and HMAC-SHA1.

REFERENCES

[1] Patrick P. Tsang, Apu Kapadia, "Nymble: Blocking Misbehaving Users in Anonymizing Networks", *IEEE Trans. Dependable and Secure Computing, vol. 8, no. 2, March 2011.*

[2] A. Juels and J.G. Brainard, "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks," *Proc. Network and Distributed System Security Symp. (NDSS), 1999.*

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," *Proc. Usenix Security Symp., pp. 303-320, Aug. 2004.*

[4] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," *Proc. Ann. Int'l Cryptology Conf. (CRYPTO), Springer, pp. 1-15, 1996.*

[5] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concret Security Treatment of Symmetric Encryption," *Proc. Ann. Symp.Foundations in Computer Science (FOCS),pp. 394-403, 1997.*

[6] J.E. Holt and K.E. Seamons, "Nym: Practical Pseudonymity for Anonymous Networks," *Internet Security Research Lab Technical Report 2006-4, Brigham Young Univ., June 2006.*

[7] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation," *Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT), Springer, pp. 93-118, 2001.*

[8] D. Boneh and H. Shacham, "Group Signatures with Verifier-Local Revocation," *Proc. ACM Conf. Computer and Comm. Security, pp. 168-177, 2004.*

[9] J. Camenisch and A. Lysyanskaya, "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials," *Proc. Ann. Int'l Cryptology Conf. (CRYPTO), Springer, pp. 61-76, 2002.*

[10] T. Nakanishi and N. Funabiki, "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps," *Proc. Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT), Springer, pp. 533-548, 2005.*

[11] A. Lysyanskaya, R.L. Rivest, A. Sahai, and S. Wolf, "Pseudonym Systems," *Proc. Conf. Selected Areas in Cryptography, Springer, pp. 184-199, 1999.*