International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Enhanced Nymble for Blocking Misbehaving Users over Anonymizing Networks

M.Vanaja
(M.Tech),
Chaitanya Institute of Science and Technology, Kakinada.

M. Vamsi Krishna
Head, Dept.of CSE,
Chaitanya Institute of Science and Technology, Kakinada.

Venkata Ramana
Asst.Professor, Dept.of CSE,
Chaitanya Institute of Science and Technology, Kakinada.

***Abstract* — Anonymity networks such as TOR provide users with a means to communicate privately over the Internet. These networks help to solve the real and important problem of enabling users to communicate privately over the Internet. But some users misuse this service and they tried to deface popular websites like wikipedia. To counter such users, a system called Nymble is used. But there is a possibility of collusion of components of Nymble such as Pseudonym manager and Nymble manager. This results in security problems. Also Nymble is not scalable and robust. In this paper, we propose a method which avoids the chance of collusion between Pseudonym manager and Nymble manager and also this method is scalable and robust.**

***Keywords* — Anonymity, Blacklisting, Pseudonym, TOR.**

I. INTRODUCTION

Anonymizing networks such as Tor[1][2] route traffic through independent nodes in separate administrative domains to hide a client's IP address. Unfortunately, some users have misused such networks—under the cover of anonymity, users have repeatedly defaced popular Web sites such as Wikipedia. Since Web site administrators cannot blacklist individual malicious users' IP addresses, they blacklist the entire anonymizing network. Such measures eliminate malicious activity through anonymizing networks at the cost of denying anonymous access to behaving users.

To counter such malicious users and continuing anonymous access to behaving users, anonymizing networks such as Tor are combined with Nymble system. Nymble[3] is basically a system that intends to bind identity of an anonymous user to a pseudonym, generated from user's IP address, and simulates a service request with a ticket acquisition. This idea enables a server to complain about misbehavior of a user and blacklist his future tickets. Using this system honest users remain anonymous, a server can blacklist future connections of particular users. Moreover, all connections of a blacklisted user before the complaint remain anonymous and finally a user can check whether he is blacklisted or not at the beginning of a connection. Nymble can also counter sybil attack[4]. Nymble offers the following properties:

- **Anonymous blacklisting:** A server can block the IP address of a misbehaving user without knowing the identity of the user or his/her IP address.
- **Privacy:** Honest and misbehaving users both remain anonymous.
- **Backward anonymity:** The blacklisted user's previous activity remains anonymous/unlinkable, and is refused future connections.
- **Blacklist-status awareness:** A user can check whether he/she has been blocked before accessing services at the server.
- **Subjective judging:** Since misbehaving users are blocked without compromising their privacy, servers can provide their own definition of "misbehavior".

II. AN EXAMPLE ANONYMIZING NETWORK-TOR

TOR is the acronym for "The Onion Router". In an onion router system[5], network messages are multiply (triply) encrypted at the source and sent randomly through an IP network of routers (onion servers), where each router removes one layer of encryption — just as you can peel the layers off an onion. The Entry Point server is chosen from a smaller set of onion router servers called Entry Guards and then randomly chosen from this set. Each of the three servers, the one chosen randomly from the Entry Guard group and the other two chosen randomly from available onion router servers worldwide, then use their public key to remove one layer of encryption at a time. When the message arrives at its destination, the message is unencrypted but the receiver has no knowledge of where the message came from or what path it took to get there, only the last server to forward the data on. Not only is the receiver in the dark, but all of the intermediate nodes between the encryption source and the exit node also have no idea of the source, contents, or destination of the packets, making it impossible for anyone inside the onion network to be able to compromise the communication.

Figure 1 shows a schematic of how hidden services work. An installed hidden service advertises for clients by broadcasting its availability (1) using the hidden service protocol through random paths (virtual circuits) to servers and by storing its information and public key in the Tor Directory Server. Those servers accept the role of being an Introduction Point and store a public key for the hidden service (2). Because the path taken between the hidden service's server and the Introduction Points consists of random virtual circuits, there is no way for a client to be able to associate the two systems with one another or to learn the hidden server's IP address.

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

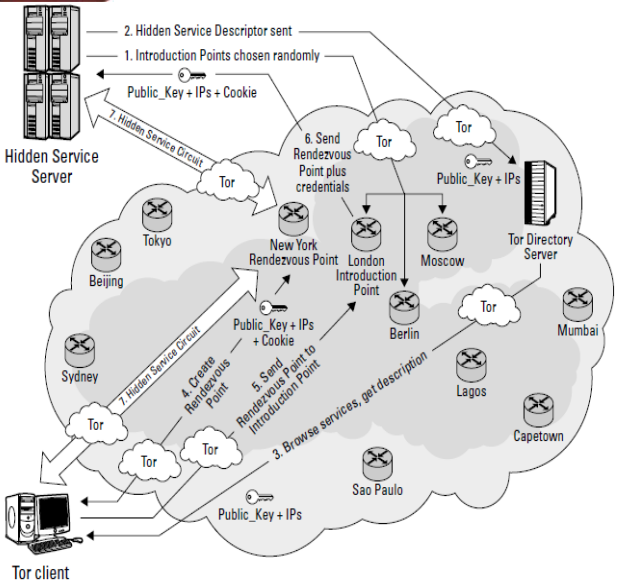


Fig.1. Hidden services on the TOR network

A Tor client learns about hidden services from the Tor Directory Server (3) and creates a Rendezvous Point. Then the Tor client communicates with one of the Introduction Point Servers (4). A Rendezvous Point contains both a public key and a cookie that are used to encrypt/decrypt information as well as supply information that allows the data to be forwarded from the hidden server to the Tor client. Once the Introduction Point transfers the Tor client's information to the Hidden Service Server, the virtual circuit shown as 7 with a large arrow is created. The system separates the Introduction Point from the Rendezvous Point, and by doing so ensures that the Tor client's information remains anonymous.

Unfortunately, some users have abused such networks to deface websites such as Wikipedia. Since servers are unable to block anonymous users, their normal response is to simply block the entire anonymizing network, denying anonymous access to honest and dishonest users alike. To counter only misbehaving users and allow honest users to use anonymizing networks like TOR, a system called Nymble is used.

III. EXISTING SYSTEM

Nymble is a credential system that can be used in conjunction with anonymizing networks such as Tor to selectively block anonymous users while maintaining their privacy. The purpose of the Nymble project is to allow for responsible, anonymous access online. It provides a mechanism for server administrators to block misbehaving users while allowing for honest users to stay anonymous; in fact even the blocked users remain anonymous. The name "Nymble" comes from a play on the word "pseudonym"[6][7] and "nimble". Instead of giving users a simple pseudonym, the Nymble system assigns users "nymbles"; that is, a pseudonym with better anonymity properties.

Nymble is based on two administratively-separate "manager" servers, the Pseudonym Manager (PM) and the Nymble Manager (NM). The PM is responsible for pairing a user's IP address with a pseudonym deterministically generated based on the user's IP address. The NM pairs a user's pseudonym with the target server. As long as the two managers are not colluding, the user's connections remain anonymous to the PM, pseudonymous to the NM (note that the user does not communicate directly with the NM, and connects to the NM through Tor), and anonymous to servers that the user connects to.

A. Connecting to a server

Pseudonym Manager :

The user (in this case, Alice) must first demonstrate control over a resource, that is the Alice's IP-address. To do this Alice must first connect directly with the PM before receiving a pseudonym. The PM has knowledge of existing Tor routers, and thus can ensure that Alice is communicating with it directly. Note that the PM has no knowledge of the user's destination, similar to the entry node in Tor. The PM's sole responsibility it to map IP addresses to pseudonyms (Figure 2).

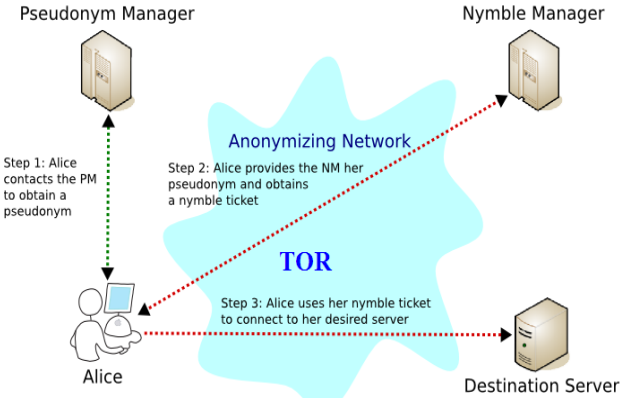


Figure 2: Connecting to a server

Nymble Manager:

Alice then connects to the NM through Tor presenting her pseudonym and her target server. The NM does not know the IP address of the user, but the pseudonym provided by the PM guarantees that some unique IP address maps to the pseudonym. She receives a set of nymble tickets as her credential for the target server. These nymble tickets are unlinkable, and therefore Alice can present these nymble tickets (once each) to gain anonymous access at the target server. The nymble ticket provides cryptographic protection as well as a trap door that can be accessed using a linking token

B. Blacklisting a user

Servers can present a user's nymble ticket to the NM as part of a complaint. The NM extracts a "linking token" from the nymble ticket that will allow the server to link future connections by the blacklisted user. The NM also issues servers with blacklists[8], which users can examine before performing any actions at the server. By checking servers' blacklists, blacklisted users are assured that their privacy is not compromised(Figure 3). We now explain how nymble tickets are bound to certain "time periods" and "linkability windows."

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

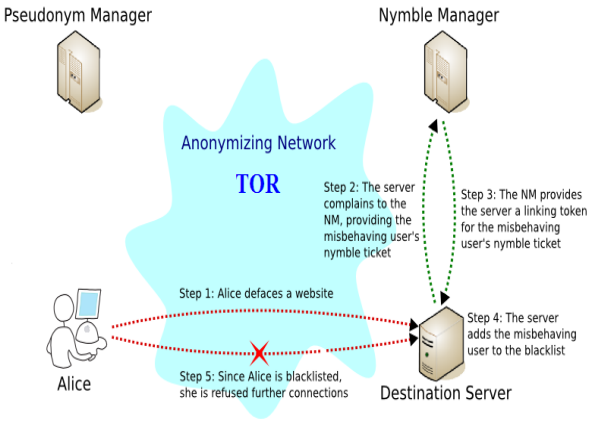


Fig.3. Blacklisting a user

Time in the nymble protocol is divided into linkability windows of some duration (default is 1 day). A linkability window(Figure 4) is then further divided into smaller time periods (default is 5 minutes). We illustrate the concepts in the diagram bellow; the linkability window is represented by the large, transparent rectangle while the time periods are labeled t0, t1, t2, etc.

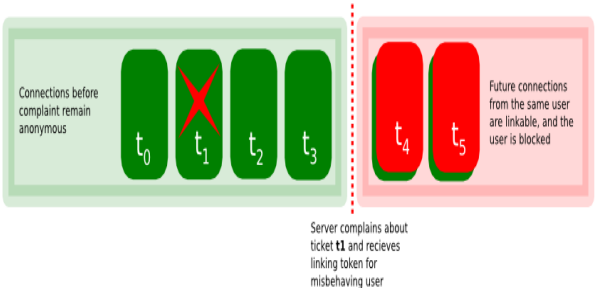


Fig.4. Linkability Window

A user's connections within a time period are tied to a single nymble ticket. If and when a user misbehaves, the server may not realize it for some amount of time and may not report it until a later time period. However, after receiving a linking token the server is able to block all future connections until the next linkability window. This is done for two reasons:

- **Dynamism:** IP-addresses can be reassigned to different, well-behaved users making it undesirable to permanently blacklist IP-addresses.
- **Forgiveness:** It ensures that bad behavior is forgiven after a certain amount of time.

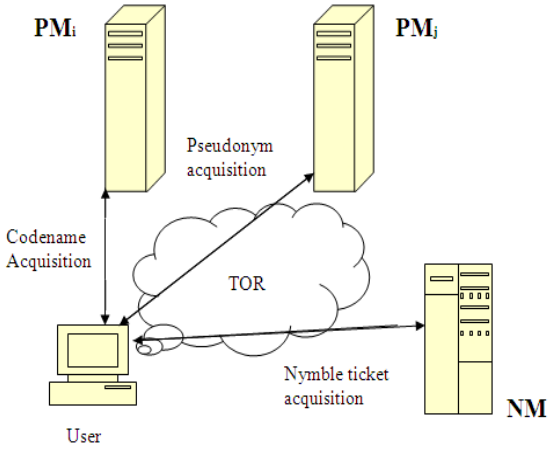
Problems with Nymble System: The main problem in Nymble is security. The security of the system depends on Pseudonym Manager (PM) and Nymble Manager (NM). There is a possibility of the PM and the NM colluding to de-anonymize the user. The other problem is that the Nymble system is not scalable and robust. This is because of only one Nymble Manager in this existing system, which handles the task controlling misbehaving users. If this Nymble manager fails, the misbehaving users cannot be blacklisted and they can continue misbehaving.

IV. PROPOSED SYSTEM

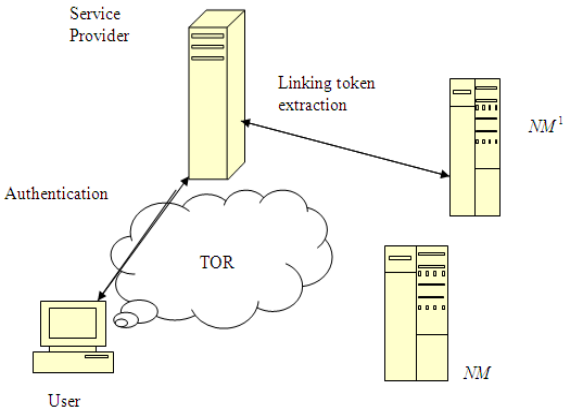
In the proposed scheme, more than one Nymble manager is used. Every NM should be able to do the following: A NM generates a chain of nymble tickets for the user. The integrity of the nymble ticket is verified without knowing issuer's identity. At last extracting linking token from a nymble ticket which was issued by another NM.

The steps in the proposed scheme are as follows:

1. First NM verifies the integrity of the pseudonym and generates nymble tickets for that connection. After that sign it using ring signature method:



2. Now the Nymble manager verifies the integrity of the nymble ticket and then decrypt it to obtain the linking token:



A. Nymble Manager Initialization:

Nymble manager can be initialized using the algorithm 1.

Algorithm 1: NMInitState

Output: {nmState}

1 $signKey_{NMg}$, $verifyKey_{NMg}$:= RSA.KeyGen()
2 encKeyNM := Enc.GetSharedKey()
3 seedKeyNM := MA.GetSharedKey()
4 Extract $verifyKey_{PM1...nn}$ from PMs' certificates
5 Extract verifyKeyNM1..i-1…i+1...m from NMs' certificates
6 keys: = ($signKey_{NMi}$, $verifyKey_{NMi}$,$verifyKey_{NM1...i-1...i+1...m}$, $encKey_{NM}$, $seedKey_{NM}$, $verifyKey_{PM1...n}$)
7 nmEntries := 0

8 *return nmState := (keys,nmEntries)*

NMi is a Nymble Manager which runs the protocol. As a result of the interaction between NMs, shared key *encKeyNM* and *seedKeyNM* for encrypting the linking token and creating the seed respectively will be generated. Moreover, each NM should generate a public key (*verifyKey*) and a private key (*signKey*) in order to sign and verify the integrity of nymble tickets and the blacklist of a server using a ring signature scheme.

B. Pseudonym Manager Initialization:

All PMs will execute PMInitState () to generate all necessary keys for Codename and Pseudonym Registration. Each PM will choose a pair of *signKey* and *verifyKey* as its private and public keys. The *verifyKey* is published to everyone else. Those keys will be then used for the ring signature scheme later. The *pmState* is initiated to contain all the keys. PM can be initialized using algorithm 2:

Algorithms 2: PMInitState

Output: {pmState}

1 signKey, verifyKey := RSA.KeyGen()

2 codenameKey := MA.GetCodenameKey()

3 nymKey := MA.GetPnymKey()

4 Extract $verifyKeyPM_{1...s-1,s+1...n}$ from keys in PMs' certificates

5 keys := ($signKeyPM_s$, $verifyKeyPM_s$, codenameKey, nymKey, $verifyKeyPM_{1...s-1,s+1...n}$)

6 return pmState := (keys)

C. Server Registration:

A Server can ask any Nymble manager to create an initial state for it. Then NM creates an empty blacklist for that server and signs it. Algorithm 3 is used for server registration:

Algorithm 3: NMRegisterServer

Input: {sid,t,w}

Output: {svrState}

1 Extract $verifyKey_{NM1...m}$ from keys in nmState

2 (keys;nmEntries) := nmState

3 nmEntries′ := nmEntries|| (sid; $daisy_L$, t)

4 nmState := (keys, nmEntries′)

5 target := $h_{(L-t+1)}$ ($daisy_L$)

6 blist := 0

7 cert := NMSignBL(sid,t,w, target,blist)

8 return svrState := (sid, verifyKeyNM1…m , blist, cert, 0, 0, 0, t)

D. Acquiring Nymble Ticket

In nymble ticket acquisition protocol, the user sends her pseudonym and the id of server to a randomly selected Nymble Manager in order to obtain a list of nymble tickets for consecutive time periods. The Nymble Manager first runs NMVerifyPseudonym to check whether the pseudonym is valid. Then it runs NMCreateCredential to create the list of nymble tickets(called *cred*) for the user and sign every ticket using a ring signature scheme that allows other NMs and all servers to verify it. Algorithms 4 and 5 are used for acquiring nymble tickets.

Algorithm 4: NMVerifyPseudonym

Input: {pnym,w}

Output: {true or false}

1 Extract $verifyKey_{PM1...n}$ from keys in nmState

2 (nym,σ_{nym}) := pnym

3 return σ_{nym} == RingSig:Verify(nym|| || w, $verifyKey_{PM1...n}$)

Algorithm 5: NMCreateCredential

Input: {pnym,sid,w}

Output: cred

1 Extract $verifyKey_{NM1...m}$, $signKey_{NMg}$, $encKey_{NM}$, $seedKey_{NM}$ from keys in nmState

2 $seed_0$:= f(MA.MAC(pnymsid||sid||w, $seedKey_{NM}$))

3 $nymble_*$:= g($seed_0$)

4 for t from 1 to L do do

5 $seed_t$:= f($seed_{t-1}$)

6 $nymble_t$:= g($seed_t$)

7 ctxtt := Enc:Encrypt($nymble_*$|| seedt,$encKey_{NM}$)

8 ticket′t := sid||t||wn||$ymble_t$||$ctxt_t$

9 σ_{ticket_t} := RingSig.Sign($ticket'_t$, $verifyKey_{NM1...m}$, $signKey_{NMg}$)

10 ticket[t] := (t, nymblet, ctxt, σ_{ticket_t})

11 return cred := ($nymble_*$, tickets)

E. Complaining and Black list updating:

In our system, blacklists should be kept up to date to contain nymble* of recent complaints and to be verified successfully by users in a time period during a linkability window. Each server should update its blacklist by the first connection establishment request in each time period. If there is no new complaints the randomly selected Nymble Manager (*NMc*) should update the daisy and send back the new daisy with current time period index to the server. Then the server have to update the certificate and replace *td* and daisy. In the case that the server wants to complain about a list of misbehaving users, *NMc* runs NMHandleComplaints algorithm after receiving blacklist, certificate and the list of tickets prepared for complaint. In this algorithm, the Nymble Manager should first verify integrity of blacklist by running NMVerifyBL and then verify integrity of all tickets sent by the server by running NMVerifyTicket algorithm. The last step is creating new blacklist entries and new seeds for linking blacklisted users. Last step should return the same number of entities as the size of complaint list; otherwise, the server can learn if two complaints were about the same user or not. Algorithms 6 and 7 are for complaining about misbehaving and verifying blacklist status:

Algorithm 6: UserVerifyBL

Input: {sid, t, w, blist, cert}

Output: {true or false}

1 Extract $verifyKey_{NM1...m}$ from NMs' certificates

2 (t_d, daisy,t_s, σ_{BL}) := cert

3 if t_d is not equal to t v $t_d < t_s$ then

4 return false

5 target := $h_{(td-ts)}$(daisy)

6 return RingSig:Verify(sid|ts||w||target||blist,σ_{BL}, $verifyKey_{NM1...m}$)

Algorithm 7: UserCheckIfBlacklisted

User check is blacklisted….

Input: (sid, blist)

Output: b ∈ {**true**, **false**}

1: Extract $nymble_*$ from cred in usrEntries[sid] in usrState

2: **return** ($nymble_*$ ∈ blist)

V. CONCLUSION

Nymble is a system that allows websites to selectively blacklist users of anonymizing networks such as Tor without knowing the user's IP-address. Users not on the blacklist enjoy anonymity while blacklisted users are not allowed future connections for a duration of time while their previous connections remain unlinkable. Since Nymble allows websites to blacklist anonymious users of their choice, and since users are notified of their blacklist status, Nymble gives websites the power to define their own definition of "misbehavior". But Nymble suffers from security problems, because there is a chance of collusion between PM and NM. Our proposed method avoids this problem and also the method is scalable and robust. Even if a Nymble manager fails, the process continues with the other NM's. This method also reduces the heavy overload on PM and NM.

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second- Generation Onion Router," Proc. Usenix Security Symp., pp. 303- 320, Aug. 2004.
[2] http://www.torproject.org.in/
[3] Patrick P. Tsang, Apu Kapadia, Cory Cornelius, and Sean W. Smith. Nymble: Blocking misbehaving users in anonymizing networks. *IEEE Transactions on Dependable and Secure Computing*, 99(1), 2011.
[4] J.R. Douceur, "The Sybil Attack," Proc. Int'l Workshop on Peer-to- Peer Systems (IPTPS), Springer, pp. 251-260, 2002.
[5] J. Feigenbaum, A. Johnson, and P.F. Syverson, "A Model of Onion Routing with Provable Anonymity," Proc. Conf. Financial Cryptography, Springer, pp. 57-71, 2007.
[6] A. Lysyanskaya, R.L. Rivest, A. Sahai, and S. Wolf, "Pseudonym Systems," Proc. Conf. Selected Areas in Cryptography, Springer, pp. 184-199, 1999.
[7] D. Chaum, "Showing Credentials without Identification Transfeering Signatures between Unconditionally Unlinkable Pseudonyms," Proc. Int'l Conf. Cryptology (AUSCRYPT), Springer, pp. 246-264, 1990.
[8] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81, New York, NY, USA,2007. ACM.

AUTHOR'S PROFILE

M. Vanaja
M.Tech.
Chaitanya Institute of Science and Technology,
Kakinada (AP), India

M. Vamsi Krishna
Head, Dept.of CSE
Chaitanya Institute of Science and Technology,
Kakinada (AP), India

Venkata Ramana
Asst.Professor, Dept.of CSE
Chaitanya Institute of Science and Technology,
Kakinada (AP), India