NYMBLE BLOCKING SYSTEM

Anand Joshi[1], Arshiya Shaikh[2], Aruna Kadam[3] , Vasudha Sahu[4]

Department of Computer Engineering, MAE Alandi (D), Pune
anandjoshi89@gmail.com
arshiya91shaikh@gmail.com
kadam.aruna@gmail.com
rapunzel.vasudha@gmail.com

Abstract

In order to allow users to access Internet services privately, anonymizing networks like Tor uses a series of routers to hide the client's IP address from the server. These networks, however, have been marred by users employing this anonymity for abusive purposes such as defacing popular web sites. Usually, web site administrators rely on IP-address blocking in order to disable access to misbehaving users, but it is impractical if the abuser routes through an anonymizing network. In order to avoid this, administrators bar all known exit nodes of the anonymizing network, thereby denying anonymous access to all the users (whether misbehaving or not). To solve this issue, we introduce Nymble, a system where servers blacklist misbehaving users, thereby blocking users without affecting their anonymity. Nymble is thus agnostic to varied definitions of misbehavior. Servers can block users for any reason, and the privacy of blacklisted users is not affected in any case.

Keywords :

Anonymous blacklisting; Privacy; Revocation

1.INTRODUCTION

IN order to hide a client's IP address anonymizing networks like Tor route traffic through independent nodes in separate administrative domains. Some users, however, have misused such networks—under the cover of anonymity, they have repeatedly defaced popular Web sites such as Wikipedia. As administrators cannot block individual users' IP addresses, they resort to blacklisting the entire anonymizing network. However, such methods though eliminate malicious activity through anonymizing networks but they also deny anonymous access to behaving users. (A case repeatedly observed with Tor.1). In a pseudonymous credential system, users log into Web sites using pseudonyms, which can be blocked if in case a user misbehaves. However, this method may results in pseudonymity for all users, thereby dampening the anonymity provided by the anonymizing network. Anonymous credential systems employ group signatures. On the other hand, basic group signatures allow servers to annul a misbehaving user's anonymity by complaining about it to a group manager. Servers must contact the group manager for every authentication, and thus, this method lacks scalability. Traceable signatures help the group manager, which then release a trapdoor allowing all signatures generated by a particular user to be traced. Even though, using such an approach does not provide the necessary backward unlinkability that we desire, a user's accesses before the complaint always remain anonymous. Backward unlinkability allows for immanent blacklisting, in which servers can blacklist users for whatever reason as the privacy of the blacklisted user is not at risk. In contrast, approaches without backward unlinkability need to pay careful attention to when and

DOI : 10.5121/ijcses.2012.3207

why a user must have all their connections linked, and users must worry about whether their behaviors will be judged fairly. Subjective blacklisting is more suited to Wikipedia like servers, where misbehaviors such as erronous edits to a Webpage, are tough to specify in exact mathematical terms. In some systems, misbehavior can indeed be defined precisely.
These methods hold true for only a few definitions of misbehavior — it is quite arduous to map more complex definitions of misbehavior with related approaches. With dynamic accumulators, a cancelling operation might result in a new accumulator and public parameters for the group, and making it mandatory to update all other existing users' credentials, thus making it impractical. Verifier-local revocation (VLR) overcomes this by requiring the server ("verifier") to perform only local updates during revocation. But VLR calls for heavy computation at the server side that is linear in the size of the blacklist. In contrast, our scheme takes the server about one millisecond per authentication, which is way faster than VLR. These low overheads help servers to use a solution when compared against the potential benefits of anonymous publishing

1.1 Nymble

To solve the given problem, we introduce a system called Nymble, which possesses the following properties: anonymous authentication, backward unlinkability, subjective blacklisting, fast authentication speeds, rate-limited anonymous connections, revocation auditability (where users can verify whether they have been blacklisted), and it also deals with the Sybil attack so as to make its implementation practical. In Nymble, users acquire a set of nymbles, a unique type of pseudonym, in order to connect to Web Servers. Lacking any other information, these nymbles are logically hard to link, and hence, using the collection of nymbles simulates unidentified access to services. Web sites, nevertheless, can block users by obtaining a seed for a specific nymble, and thus allowing them to establish a connection with future nymbles from the user — and those prior to the complaint remain unlinkable and untraceable.
Servers can thus block anonymous users without gaining access to their IP addresses while allowing legitimate users to connect anonymously. Our system let the users know about their blacklisted status before they are introduced to a nymble, and are disconnected immediately in case they are blacklisted. A large number of anonymizing networks can rely on the same Nymble system, and blacklisting anonymous users regardless of their anonymizing network.

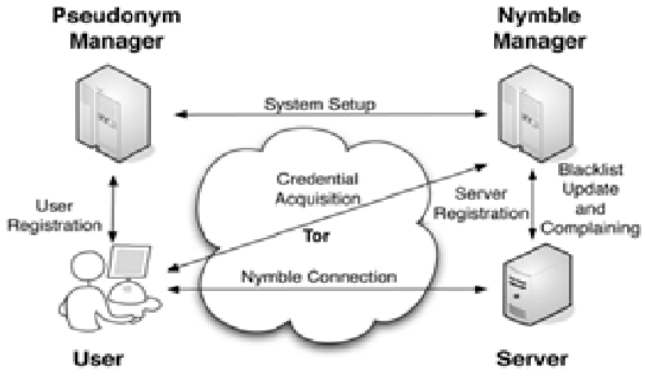

Fig 1. The Nymble system architecture showing the various modes of interaction. Note that users interact with the NM and servers through the anonymizing network.

2 AN OVERVIEW TO NYMBLE

We now introduce a high-level overview of the Nymble architecture, and divide the entire protocol description and security analysis to the respective sub-sections.

2.1 Resource-based blocking

To keep a tab on the total number of identities that a user can obtain (popularly known as the Sybil attack), the Nymble system attaches nymbles to resources which are difficult enough to obtain in great numbers. For example, here we have used IP addresses as the resource, but our scheme generalizes to other resources as well such as email addresses or identity certificates. The issues related with resource-based blocking is discussed further in Section 8, where we have suggested other alternatives for resources. The Sybil attack problem is faced by any credential system and we suggest some promising approaches based on resource-based blocking since we aim to create a real-world deployment.

2.2 The Pseudonym manager

The user initially must connect to Pseudonym Manager (PM) and establish control over a resource; so as to block the IP-address, the user ought to connect to the Pseudonym Manager directly, as shown in Fig. 1. We presume that PM has knowledge of Tor routers and can ensure that users are communicating with it directly. Pseudonyms are chosen based on the controlled resource, making sure that the very pseudonym is always issued for the same resource. The user does not disclose what server he wants to connect to, and the PM's duties are restricted to mapping IP addresses (or other resources) to pseudonyms. The user connects to the PM only once per linkability window (e.g., once a day).

2.3 The Nymble Manager

Post gaining a pseudonym from the PM, the user connects to the Nymble Manager via the anonymizing network, and then request for nymbles to obtain access to a particular server. A user's requests to the NM are therefore pseudonymous, and nymbles are generated using the user's pseudonym and the server's identity. Nymbles are thus specific to a particular user-server pair. As long as the PM and the NM do not collude, the NM knows only the pseudonym-server pair, and the PM knows only the user identity-pseudonym pair. In order to provide the required cryptographic protection and security properties, nymbles are encapsulated within nymble tickets. Servers pack seeds into linking tokens, and therefore, we will speak of linking tokens being used to link future nymble tickets.

2.4 Time

Nymble tickets are linked with specific time periods. Time is divided into linkability windows of duration W, each of which is split into L time periods of duration T (i.e., W ¼L _ T) as shown in Fig.2. Time periods and linkability windows are chronologically referred to as t1; t2; . . . ; tL and w1; w2; . . . , respectively. While a single user's access within a time period is allotted a single nymble ticket, different nymble tickets across time periods provides the user with anonymity between different time periods. Smaller time periods provide users with higher rates of anonymous authentication, while longer time periods allow servers to rate-limit the number of misbehaviors from a particular user before he is blocked. The linkability window allows for dynamism since resources such as IP addresses can get reassigned and it is not desirable to blacklist such resources indefinitely, and it ensures forgiving a misbehaving user

after a certain window linkability period. All entities are time synchronized, and we can thus calculate the current linkability window and time period.

2.5 Blacklisting a user

In case of a misbehavior, the server may link any future connection from this user within the same linkability window. Consider Fig. 2: A user misbehaves at a server during time period t_ within linkability window w_. The server then finds this misbehavior and reports it to the NM in time period tc (t_ < tc _ tL) of the same linkability window. In the complaint, the server presents the nymble ticket of the misbehaving user and obtains the corresponding seed from the NM. The server is then able to link future connections by the user in time periods tc; tc þ 1; . . . ; tL of the same linkability window w_ to the complaint. Therefore, once the server has complained about a user, that user is blacklisted for that particular linkability window. Even though misbehaving users can be blocked for the future too, the past connections anyhow remain unlinkable, providing subjective blacklisting and backward unlinkability.

2.6 Notifying the user of blacklist status

Users using anonymizing networks want their connections to be anonymous. When a server obtains a seed for that user, it can still link the user's subsequent connections. It is very important that users be notified of being blacklisted before presenting a nymble ticket to a server. The user can thus download the server's blacklist and verify its status. When blacklisted, the user immediately gets discontinued.

As the blacklist is cryptographically signed by the NM, the blacklist's credibility is easily verified as to if the blacklist was updated in the same time period. Otherwise, the NM provides servers with "daisies" every time period so that users are able to verify the freshness of the blacklist. As discussed further, these daisies are elements of a hash chain, providing a lightweight alternative to digital signatures. Thus, we ensure that race conditions are not possible in verifying a blacklist's novelty. A user is guaranteed that he or she will not be linked if the user verifies the integrity and freshness of the blacklist before sending his or her nymble ticket.

3 SECURITY MODEL

Nymble aims for four security goals. We provide informaldefinitions here; a detailed formalism can be found in our technical report , which explains how these goals must also resist coalition attacks.

3.1 Goals and threats

An entity can be termed as honest when its operations abide by the system's specification. An honest entity attempts to infer knowledge from its own information (e.g., its secrets, state, and protocol communications). An honest entity becomes corrupt when it is compromised by an attacker, and hence, reveals its information at the time of compromise.

Blacklistability assures that any legitimate server can surely block misbehaving users. Also, if an honest server complains about a misbehaving user in the present linkability window, it will be successful and the user will not be able to connect.
Rate-limiting assures that any legitimate server that no user can connect to it more than once within any single time period.
Nonframeability guarantees that any legitimate user can connect through nymble to that server. This keeps an attacker from framing a legitimate user, e.g., by getting the user blocked for someone else's misbehavior. Here we assume each user has a single unique identity. When IP

addresses are used, a user can be "framed" as an honest user who later obtains the same IP address. Nonframeability holds true only against attackers with different IP addresses. A user is considered legitimate by a server only if he has not been blacklisted, and has not exceeded the rate limit. Honest servers are able to distingish between honest and dishonest users.

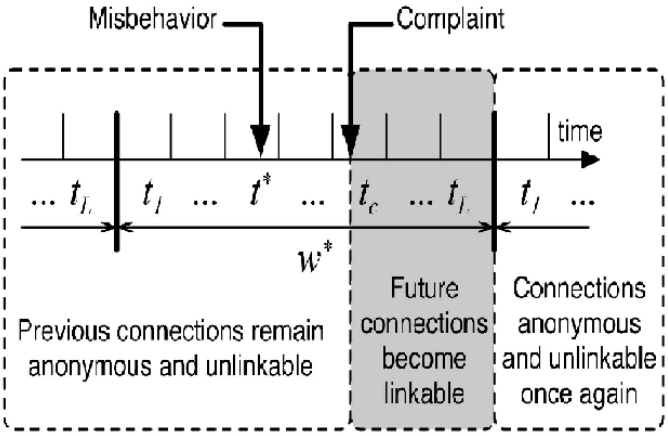

Fig 2. The life cycle of a misbehaving user. If the server complains in time period t_c about a user's connection in t^*.the user becomes linkable starting in t_c. The complain in t_c. can include nimble tickets from onlu t_{c-1} and earlier

Anonymity protects the anonymity of honest users, regardless of their legitimacy according to the (possibly corrupt) server; the server cannot learn any more information beyond whether the user behind (an attempt to make) a nymble connection is legitimate or illegitimate.

4 PREREQUISITES

4.1 Notation

The notation a€ $_R$ S represents an element drawn uniformly at random from a nonempty set S.№ is the set of nonnegative integers, and N is the set №\{0}. S[i] is the ith element of list s||t is the concatenation of (the unambiguous encoding of) List s and t. The empty list is denoted by Ø. Lists of tuples are sometimes counted as dictionaries. For example, if L is the list ((alice, 1234), (Bob,5678)), then L[Bob] denotes the tuple (Bob,5678). If A is an (possibly probabilistic) algorithm, then A(x) denotes the output when A is executed given the input x.a :=b means that b is assigned to a.

4.2 Data structures and Modules

Nymble uses several important data structures, and we divide them into the following 4 modules:

4.2.1 Module 1: Generation of pseudonym

The PM issues pseudonyms to users. A pseudonym pnym

has two components nym and mac: nym is a pseudorandom

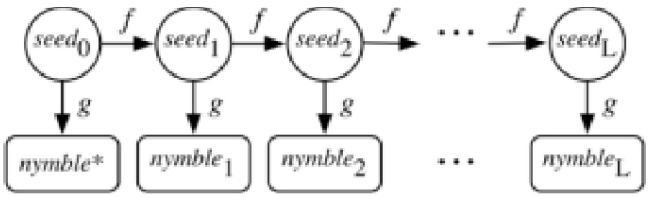

Fig 4. Evolution of seeds and nymbles. Given $seeds_{i,}$, it is easy to Compute $nymble_{i,}$ $nymble_{i+1}$.....$nymble_{L,}$ but not $nymble^{*,}$ $nymble_{i}$....., $nymble_{i-1.}$

mapping of the user's identity (e.g., IP address),7 the linkability window w for which the pseudonym is valid, and the PM's secret key nymKeyP ; mac is a MAC that the NM uses to verify the integrity of the pseudonym.

Algorithms 1 and 2 describe the procedures of creating and verifying pseudonyms.

Algorithm 1. PMCreatePseudonym
Input: (uid,w) € H * №
Persistent state: pmState € S_p
Output: pnym € P
1: Extract $nymKey_P$; $macKey_{NP}$ from pmState
2: nym :¼ MA:Mac(uid||w, $nymKey_P$)
3: mac :¼ MA:Mac(nym||w, $macKey_{NP}$)
4: return pnym :¼ (nym, mac)

Algorithm 2. NMVerifyPseudonym
Input: (pnym;w) € P * №
Persistent state: nmState 2 S_N
Output: b € (true; false)
1: Extract $macKey_{NP}$ from nmState
2: (nym; mac) := pnym
3: return mac = ? MA:Mac(nym||w,$macKey_{NP}$)

The NM executes NMInitState and initializes nmState in order to generate the algorithm's output. The NM extracts $macKey_{NP}$ from nmState and sends it to the PM over a type-Auth channel. $macKey_{NP}$ is a shared anonymously between the NM and the PM, so that the NM can verify the authenticity of pseudonyms issued by the PM. (Refer algorithm 3)

Algorithm 3. NMInitState

Output: nmState € SN

1: macKeyNP := Mac:KeyGen()

2: macKeyN := Mac:KeyGen()

3: seedKeyN := Mac:KeyGen()

4: (encKeyN; decKeyN) := Enc:KeyGen()

5: (signKeyN; verKeyN) := Sig:KeyGen()

6: keys :=(macKeyNP ; macKeyN; seedKeyN, encKeyN; decKeyN; signKeyN; verKeyN)

8: nmEntries := ¢;

9: return nmState :=(keys;nmEntries)

4.2.2 Module 2: Generation of nymble

Nymble, a pseudorandom number, plays the role of an identifier for a particular time period. Nymbles (presented by a user) across periods are unlinkable unless a server has blacklisted that user. Nymbles are presented as part of a nymble ticket, as described next. As shown in Fig. 4, seeds evolve throughout a linkability window using a seed-evolution function f; the seed for the next time period (seednext) is computed from the seed for the current time period (seedcur) as seednext ¼ f(seedcur):

The nymble (nymblet) for a time period t is evaluated by applying the nymble evaluation function g to its corresponding seed (seedt), i.e., nymblet ¼ g(seedt): The NM sets seed0 to a pseudorandom mapping of the user's pseudonym pnym, the (encoded) identity sid of the server (e.g., domain name), the linkability window w for which the seed is valid, and the NM's secret key seedKeyN. User-server-window combinations are therefore specifically linked to seeds. As a consequence, a seed is useful only for a particular server to link a particular user during a particular linkability window.

In our Nymble construction, f and g are two distinct cryptographic hash functions. Hence, it is easy to compute future nymbles starting from a particular seed by applying f and g appropriately, but infeasible to compute nymbles otherwise. Without a seed, the sequence of nymbles appears unlinkable, and honest users can enjoy anonymity. Even when a seed for a particular time period is obtained, all the nymbles prior to that time period remain unlinkable.

A credential contains all the nymble tickets for a particular linkability window that a user can present to a particular server. Algorithm 4 has a procedure that generates a credential when requested: A ticket contains a server specific nymble, linkability window and time period. ctxt is scrambled data that NM uses during a nymble ticket complaint. In particular, ctxt contains the first nymble (nymble_) in the user's sequence of nymbles, and the seed used to generate that nymble. Upon a complaint, the NM extracts the user's seed and issues it to the server by evolving the seed, and nymble helps the NM to recognize whether the user has already been blacklisted.

Algorithm 4. NMCreateCredential

Input: (pnym; sid;w) € P * H* №

Persistent state: nmState € S_N

Output: cred € D

1: Extract $macKey_{NS}$; $macKey_N$; $seedKey_N$; $encKey_N$ from

keys in nmState

2: seed0 := f(Mac(pnym||sidk||w,$seedKey_N$)

3: nymble* := ($seed_0$)

4: for t from 1 to L do

5: $seed_t$:= f($seed_t$_1)

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

7: $ctxt_t$:= Enc:Encrypt(nymble*||$seed_t$; $encKey_N$)

8: $ticket_t$:= sid||t||w$knymble_t$||$ctxt_t$

9: macN,t := MA:Mac($ticket_t$; $macKey_N$)

10: macNS;t := MA:Mac($ticket_0$||macN,t; $macKey_{NS}$)

11: tickets[t]:= (t, $nymble_t$, $ctxt_t$, macN,t; macNS,t)

12: return cred := (nimble*, tickets)

The MACs macN and macNS are used by the NM and the server, respectively, to verify the integrity of the nymble ticket, as described in Algorithms 5 and 6(Under next module). As will be explained later, the NM will need to verify the ticket's integrity upon a complaint from the server.

Algorithm 5. NMVerifyTicket

Input: (sid, t, w,ticket) € H * №^{2} * T

Persistent state: svrState

Output: b € (true; false)

1: Extract $macKey_N$ from keys in nmState

2: (. , nymble, ctxt, mac_N, mac_{NS}) := ticket

3: content := sid||t||w||nymble||ctxt

4: return macN =? MA:Mac(content, $macKey_N$)

4.3.3 Connection to website using nymble and seed

Here in this module, the algorithm performs the task of checking if the linkability of the ticket (using Algorithm no. 6) . If the nymble is linked to the server then we can conclude that the user has misbehaved and thus the status of the user is updated.

Algorithm 5. ServerVerifyTicket

Input: (t, w, ticket) € №^{2} * T

Persistent state: svrState

Output: b € (true; false)

1: Extract sid; $macKey_{NS}$ from svrState

2: (. , nymble, ctxt, macN; macNS) := ticket

3: content := sid||t||w||nimble||ctxt||macN

4: return macNS = ? MA:Mac(content, $macKey_{NS}$).

Algorithm 6. ServerLinkTicket

Input: ticket € T

Persistent state: svrState € SS

Output: b € (true; false)

1: Extract lnkng-tokens from svrState

2: (.; nymble; …) := ticket

3: for all i = 1 to | lnkng-tokens | do

4: if (.; nymble) = | lnkng-tokens[i] then

5: return true

6: return false

Algorithm 7. ServerUpdateState

Persistent state: svrState 2 S_s

1: Extract lnkng-tokens from svrState

2: for all i = 1 to | lnkng-tokens | do

3: (seed; nymble) := lnkng-tokens[i]

4: seed' := f(seed); nymble' := g(seed')

5: tokens'[i] := (seed'; nymble')

6: Replace lnkng-tokens in svrState with tokens'

7: Replace seen-tickets in svrState with ¢

4.3.4 Blacklisting Misbehaved User

A server's blacklist is a list of nymble s corresponding to all the nymbles that the server has complained about. Users can quickly check their blacklisting status at a server by checking to see whether their nymble appears in the server's blacklist (see Algorithm 8).

Algorithm 8. UserCheckIfBlacklisted

Input: (sid; blist) € H*Bn, n; l € N_0

Persistent state: usrState € S_u

Output: b € (true; false)

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

2: return (nymble*€?blist)

Algorithms 9 describe how users and the NM can verify the integrity and freshness of blacklists.

Algorithm 9. NMVerifyBL

Input: (sid; t; w; blist; cert) € H*N2 *Bn * C, n € N0

Persistent state: nmState 2 SN

Output: b € (true; false)

1-6: Same as lines 1-6 in VerifyBL

7: Extract macKeyN from keys in nmState

8: return mac = ? MA:Mac(content; macKeyN)

NMComputeBLUpdate (see Algorithm10) creates new entries to be appended to the server's blacklist. Each entry is either the actual nymble of the user being omplained about if the user has not been blacklisted already, or a random nymble otherwise. This way, the server cannot learn if two complaints are about the same user, and thus, cannot link the Nymble connections to the same user.

Algorithm 10. NMComputeBLUpdate

Input: (sid; t; w; blist; cmplnt-tickets) € H*N2 *Bn * T m

Persistent state: nmState € SN

Output: (blist'; cert') € Bm * C

1: (keys;nmEntries) := nmState

2(. ; macKeyN; seedKeyN; encKeyN; _; signKeyN; .) := keys

3: for i = 1 to m do

4: (.; .; ctxt; .; .) := cmplnt-tickets[i]

5: nymble*||seed := Decrypt(ctxt; decKeyN)

6: if nymble_ € blist then

7: blist'[i] € R H

8: else

9: blist'[i] := nymble*

10: daisyL € R H

11: target' := h(L-t+1)(daisy'L)

12: cert' := NMSignBL(sid; t; w; target'; blistkblist')

13: Replace daisyL and tlastUpd in nmEntries[sid] in nmState with daisy0 L and by t,

respectively

14: return (blist'; cert')

5. ADDITIONAL FEATURES

5.1 Enterprisal E-mail Server

Enterprises these days are facing a unique problem i.e. loss of confidential information. Any firm's one of the most valued asset is its data, which it somehow plans to keep out of bounds for any outsider. Thereby, most firms restrict the use of data storage devices like pen drives or recordable disks in the office premises, in order to put a check on the loss of secret company information. However, it is found that data can be e-mailed to servers which can be accessed even from outside the company bounds. That way, any ominous person can e-mail the documents to someone who might use them for ill purposes.

Hereby, we present a system for an enterprise e-mail server. We will design an e-mail server (or rent an e-mail domain i.e. name@company.com from Gmail or any other domain selling websites) which will forbid the loss of sensitive information. The e-mail server apart from

searching all the communicated e-mails for any abusive content, also restricts the transfer of mails to outbound servers, in order to prevent leak of secret information.

5.2 User based content hosting websites

User based content hosting websites such as Wikipedia etc. are currently unable to restrict a user from making abusive comments. These websites despite having a large user base are often marred by abusive contents posted on their website by anonymous users. They, given the large number of comments posted per second, do not have a server side mechanism and usually rely on existing users in order to post a content as abusive. As inefficient as that may be, it is the only reliable way currently available to handle the issue.
Here we present Nymble, a system to detect an abusive post by itself, thereby eliminating the need to rely on users for the same. The system, detects any abusive behavior from a user and if found guilty, blocks him from accessing the website for a stipulated amount of time. However, after the time is over, the user is forgiven, and he can access the website again until he misbehaves again.

6 SECURITY ANALYSIS

Our Nymble construction has Blacklistability, Rate-limiting, Nonframeability, and Anonymity provided that the trust assumptions in Section 3.2 hold true, and the cryptographic primitives used are secure.

6.1 Blacklistability

An honest PM and NM will issue a coalition of c unique users at most c valid credentials for a given server. Due to the security of HMAC, only Nymble Manager can issue valid tickets, and for any given time period, the coalition has at most c valid tickets, thus making at most c connections in any time period irrespective of server's blacklisting. It is sufficient to show that if each of the c users has been blacklisted in some previous time period, the coalition cannot authenticate in the time period k_. Assume the contrary that connection establishment k_ using one of the coalition members' ticket_ was successful even though the user was blacklisted in a previous time period k0. Since connection establishments k0 and k_ were successful, the corresponding tickets ticket0 and ticket_ must be valid. Assuming the security of digital signatures and HMAC, an honest server can always contact an honest NM with a valid ticket and the NM will successfully terminate ehost running Ubuntu.

6.2 Nonframeability

Assume the contrary that the adversary successfully framed honest user i_ with respect to an honest server in time period t_, and thus, user i_ was unable to connect in time period t_ using ticket_ even though none of his tickets were previously blacklisted. Because of the security of HMAC, and since the PM and NM are honest, the adversary cannot forge tickets for user i_, and the server cannot already have seen ticket_; it must be that ticket_ was linked to an entry in the linking list. Thus, there exists an entry (seed_; nymble_) in the server's linking list, such that the nymble in ticket_ equals nymble_. The server must have obtained this entry in a successful blacklist update for some valid ticketb, implying the NM had created this ticket for some user ~i.
If ~i 6¼ i_, then user ~i's seed is different from user i_'s seed as long as the PM is legitimate, and yet the two seed's evolve to the same seed_, which belies the collision resistance attribute

of the evolution function. Thus, we have ~i ¼ i_. But, as already argued, the adversary cannot forge i_'s ticketb, and it must be the case that i_'s ticketb was blacklisted before t_, which contradicts our assumption that i_ was a legitimate user in time t_.

6.3 Anonymity

We show that an adversary learns only that some legitimate user connected or that some illegitimate user's connection failed, i.e., there are two anonymity sets of legitimate and illegitimate users. Distinguishing between two illegitimate users. We argue that any two chosen illegitimate users out of the control of the adversary will react indistinguishably. Since all honest users execute the Nymble connection Establishment protocol in exactly the same manner up until the end of the Blacklist validation stage , it suffices to show that every illegitimate user will evaluate safe to false, and hence, terminate the protocol with failure at the end of the Privacy check stage .

For an illegitimate user (attempting a new connection) who has already disclosed a ticket during a connection establishment earlier in the same time period, ticket Disclosed for the server will have been set to true and safe is evaluated to false during establishment k_.

An illegitimate user who has not disclosed a ticket during the same time period must already be blacklisted.

Thus, the server complained about some previous ticket_ of the user. Since the NM is honest, the user's nymble appears in some previous blacklist of the server. Since a Nymble Manager never erases entries from a blacklist, it will be visible in all the subsequent blacklists, and safe is calculated to be false for the current blacklist. Servers cannot forge blacklists or present blacklists for earlier time periods (as, otherwise, the digital signature would be forgeable, or the hash in the daisy chain could be inverted).

Distinguishing between two legitimate users. The authenticity of the channel implies that a legitimate user knows the correct identity of the server, and thus, Boolean ticket Disclosed for the server remains false. Furthermore, User Check If Blacklisted returns false (assuming the security of digital signatures) and safe is evaluated to true for the legitimate user. Now, in the ticket presented by the user, only nymble and ctxt are functions of the user's identity. Since the adversary does not know the decryption key, the CCA2 security of the encryption implies that ctxt reveals no information about the user's identity to the adversary. Finally, since the server has not obtained any seeds for the user, under the Random Oracle model, the nymble presented by the user is indistinguishable from random and cannot be linked with other nymbles presented by the user.

Furthermore, if and when the server complains about a user's tickets in the future, the NM ensures that only one real seed is issued (subsequent seeds corresponding to the same user are random values), and thus, the server cannot distinguish between legitimate users for a particular time period by issuing complaints in a future time period.

6.4 Across Multiple Linkability Windows

With multiple linkability windows, our Nymble construction still has Accountability and Nonframeability because each ticket is valid for and only for a specific linkability window; it still has Anonymity because pseudonyms are an output of a collision-resistant function that takes the linkability window as input.

7 DISCUSSION

IP-address blocking. Using IP addresses as a resource for limiting the Sybil attack, the current implementation uses IP-address blocking used by Internet services. In either case, there are some predefined limitations of using IP addresses as the rare resource. If a user can obtain

multiple addresses, he can get through both nymble-based and regular IP-address blocking. Subnetbased blocking eases this problem, and while it is possible to modify our system to support subnet-based blocking, new security problems might surface.

Other resources. Users of anonymizing networks would be reluctant to use resources that directly reveal their identity . Email addresses could provide more privacy, but provide weak blacklistability guarantees because users can easily create new email addresses. Other possible resources include client puzzles and e-cash, where users are required to perform a certain amount of computation or pay money to acquire a credential. These approaches would limit the number of credentials obtained by a single individual by raising the cost of acquiring credentials.

Server-specific linkability windows. An enhancement would be to provide support to vary T and L for different servers. As described, our system does not support varying linkability windows, but does support varying time periods. This is due to the fact that the PM is unaware of the server that the user intends to connect to, yet it must issue pnyms bound to a linkability window. We do note that the use of resources such as client puzzles or e-cash would eliminate the need for a PM, and users could obtain Nymbles directly from the NM. In that case, server-specific linkability windows could be used.

Side-channel attacks. While our current implementation does not fully protect against side-channel attacks, we mitigate the risks. We have implemented various algorithms in a way that their execution time leaks little information that cannot already be inferred from the algorithm's output. Also, since a confidential channel does not hide the size of the communication, we have constructed the protocols so that each kind of protocol message is of the same size regardless of the identity or current legitimacy of the user.

8 CONCLUSION

The existing sytem, however efficient as it may appear, largely depends on human users to classify a post into an act of misbehavior. The misbehaving post, even when flagged as abusive, is usually not immediately removed from the website. Anonymizing networks as Tor, thus far, has been completely blocked by several web services because of users who abuse their anonymity.

Thereby in our paper titled "Nymble Blocking System", we introduce a system which algorithmically flags a post as an act of misbehavior, and thereby eradicating the necessity to depend on the existing users for the same. Our system allows websites to selectively block users of anonymizing networks. Using it, websites can blacklist users without hindering their anonymity.

REFERENCE

[1] C. Cornelius, A. Kapadia, P.P. Tsang, and S.W. Smith, "Nymble: Blocking Misbehaving Users in Anonymizing Networks," Technical Report TR2008-637, Dartmouth College, Computer Science, Dec. 2008.

[2] G. Ateniese, D.X. Song, and G. Tsudik, "Quasi-Efficient Revocation in Group Signatures," Proc. Conf. Financial Cryptography, Springer, pp. 183-197, 2002.

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

[4] M. Bellare, H. Shi, and C. Zhang, "Foundations of Group Signatures: The Case of Dynamic Groups," Proc. Cryptographer's Track at RSA Conf. (CT-RSA), Springer, pp. 136-153, 2005.

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

[6] 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.

[7] 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.

[8] J. Camenisch and A. Lysyanskaya, "Signature Schemes and Anonymous Credentials from Bilinear Maps," Proc. Ann. Int'l Cryptology Conf. (CRYPTO), Springer, pp. 56-72, 2004.

[9] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A Practical and Provably Secure Coalition-Resistant Group Signature, Scheme," Proc. Ann. Int'l Cryptology Conf. (CRYPTO), Springer, pp. 255-270, 2000.

[10] 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.

[11] S. Goldwasser, S. Micali, and R.L. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM J. Computing, vol. 17, no. 2, pp. 281-308, 1988.

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

[13] P.P. Tsang, M.H. Au, A. Kapadia, and S.W. Smith, "Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 72-81, 2007.

[14] P.C. Johnson, A. Kapadia, P.P. Tsang, and S.W. Smith, "Nymble: Anonymous IP-Address Blocking," Proc. Conf. Privacy Enhancing Technologies, Springer, pp. 113-133, 2007.

Biography

Anand Joshi is pursuing his computer engineering degree from Pune University. Having a strong inclination towards ASP.NET coding, he did his initial schooling from Amravati, Maharashtra. An excellent communicator and a fine writer, he plans to pursue his career as a software developer.

Arshiya Shaikh is currently pursuing her graduation course as a computer engineer from MAE, Alandi in Pune University. Originally hailing from Junnar, a small village in countryside Maharashtra, she is the first girl from her school to study engineering. A devoted learner, she intends to pursue teaching as a profession and pay back her due back to the society.

Aruna Kadam is currently pursuing a M.E. from University Of Pune. She did her graduation from Shivaji University, Satara. She has worked as a lecturer in Mozhe College of Engineering, Pune and currently works at Maharashtra Academy Of Engineering, Pune. She plans to continue her research in the field of computer science

Vasudha Sahu, the quintessential project leader of our project group is pursuing her B.E. degree in Computer Engineering from Pune University. She has a strong liking in the field of software development and has excellent skills in the field of programming languages like C and C++. She wants to pursue her dream to be a software developer in a major IT firm.