ISSN No. 0976-5697

Volume 4, No. 6, May 2013 (Special Issue)

International Journal of Advanced Research in Computer Science

CASE STUDY AND REPORT

Available Online at www.ijarcs.info

Nymble: Blocking Misbehaving Users inAnonymizing Networks

Dipali V. Yelane
Department of information technology
Jawaharlal darda institute of engineering & Technology, yavatmal
yelaned@ymail.com

Navita Nagdive
Department of information technology
Jawaharlal darda institute of engineering & Technology, yavatmal
navita441992@gmail.com

Prof. Dr. R. M. Tugnayat
Department of information technology
Jawaharlal darda institute of engineering & Technology, yavatmal

Abstract: Anonymizing networks such as Tor allow users to access Internet services privately by using a series of routers to hide the client's IP address from the server. The success of such networks, however, has been limited by users employing this anonymity for abusive purposes such as defacing popular websites. Website administrators routinely rely on IP-address blocking for disabling access to misbehaving users, but blocking IP addresses is not practical if the abuser routes through an anonymizing network.

As a result, administrators block *all* known exit nodes of anonymizing networks, denying anonymous access to misbehaving and behaving users alike. To address this problem, we present Nymble, a system in which servers can "blacklist" misbehaving users, thereby blocking users without compromising their anonymity.

Our system is thus agnostic to different servers' definitions of misbehavior — servers can blacklist users for whatever reason, and the privacy of blacklisted users is maintained.

I. INTRODUCTION

Anonymizing networks such as Tor 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 websites such as Wikipedia. Since web-site administrators cannot blacklist individual malicioususers' IP addresses, they blacklist theentire anonymizing network. Such measures eliminate malicious activity through anonymizing networks at the cost of denying anonymous access to behaving users. In other words, a few "bad apples" can spoil the fun for all. (This has happened repeatedly with Tor.[1])

There are several solutions to this problem, each providing some degree of accountability. Inpseudonymous credential systems, users log into websites using pseudonyms, which can be added to a blacklist if a user misbehaves. Unfortunately, this approach results inpseudonymity for all users, and weakens the anonymity provided by the anonymizing network.Anonymous credential systemsemploy group signatures. Basic group signatures allow servers to revoke a misbehaving user's anonymity bycomplaining to a group manager. Servers must query the group manager for every authentication, and thus lacksscalability.Traceable signatures allow the group manager to release a trapdoor that allowsall signatures generated by a particular user to be traced; such an approach does not provide the backward unlinkability that.

We desire, where a user's accesses before the complaint remain anonymous. Backward unlinkability allows for what we call subjective blacklisting, where servers can blacklist users for whatever reason since the privacy ofthe blacklisted user is not at risk.In contrast, approaches without backward unlinkability need to pay careful attention to when and why a user must have all their connections linked, and users must worry about whether their behaviors will be judged fairly.

Subjective blacklisting is also better suited to serverssuch as Wikipedia, where misbehaviors such as questionable edits to a webpage, are hard to define in mathematical terms. In some systems, misbehavior can indeedbe defined precisely. For instance, double-spending ofan "e-coin" is considered a misbehavior in anonymous e-cash systems, following which the offending user is deanonymized. Unfortunately, such systemswork for only narrow definitions of misbehavior — it is difficult to map more complex notions of misbehavior onto "double spending" or related approaches.

II. OVERVIEW TO NYMBLE

A high-level overview of the Nymble system, and defer the entire protocol description and security analysis to subsequent sections.

A. Resource-based blocking:

To limit the number of identities a user can obtain (calledtheSybil attack[19]), the Nymble system bindsnymbles to resources that are sufficiently difficult to obtain in great numbers. For example, we have used IP addressesas the resource in our implementation, but our schemegeneralizes to other resources such as email addresses,identity certificates, and trusted hardware. We addressthe practical issues related with resource-based blockingin Section8, and suggest other alternatives for resources.

B. The Pseudonym Manager:

The user must first contact thePseudonym Manager

CONFERENCE PAPER
"A National Level Conference on Recent Trends in Information Technology and Technical Symposium" On 09th March 2013
Organized by
Dept. of IT, Jawaharlal Darda Inst. Of Eng. & Tech., Yavatmal (MS), India

(PM) and demonstrate control over a resource; for IP-addressblocking, the user must connect to the PM directly (i.e., not through a known anonymizing network), as shown in Figure 1. We assume the PM has knowledge aboutTor routers, for example, and can ensure that users are communicating with it directly.[6] Pseudonyms are de-terministically chosen based on the controlled resource,ensuring that the same pseudonym is always issued for the same resource.Note that the user does not disclose what server he or she intends to connect to, and the PM's duties are limited to mapping IP addresses (or other resources) topseudonyms. As we will explain, the user contacts the PM only once per linkability window (e.g., once a day).

C. The Nymble Manager:

After obtaining a pseudonym from the PM, the user connects to the *Nymble Manager (NM)* through the anonymizing network, and requests nymbles for access to a particular server (such as Wikipedia). A user's requests to the NM are therefore pseudonymous, andnymbles are generated using the user's pseudonym andthe server's identity. These nymbles are thus specific toa particular user-server pair. Nevertheless, as long as the PM and the NM do not collude, the Nymble system cannot identify which user is connecting to what server;the NM knows only the pseudonym-server pair, and the PM knows only the user identity-pseudonym pair.

D. Time:

Nymble tickets are bound to specific time periods. As illustrated in Figure 2, time is divided into linkability windows of duration W, each of which is split into L time periods of duration T (i.e., W = L!T). We will referto time periods and linkability windows chronologicallyas t1, t2,.. . , tL and w1, w2,.. . respectively. While a user's access within a time period is tied to a single nymble ticket, the use of different nymble tickets across time periods grants the user anonymity between time periods. Smaller time periods provide users with higher rates of anonymous authentication, while longer time periodsallow servers to rate-limit the number of misbehaviorsfrom a particular user before he or she is blocked. Forexample, T could be set to 5 minutes, and W to 1 day (and thus L = 288). The linkability window allows for dynamism since resources such as IP addresses can get re-assigned and it is undesirable to blacklist such resources indefinitely, and it ensures forgiveness of misbehavior after a certain period of time. We assume all entities are time synchronized (for example, withtime.nist.gov via the Network Time Protocol (NTP)), and can thus calculate the current linkability window and time period.

Fig. 2. The life cycle of a misbehaving user. If the servercomplains in time period tc about a user's connection int!, the user becomes linkable starting in tc. The complaintin tc can include nymble tickets from only$_{tc-1}$ and earlier.

E. Blacklisting a user:

If a user misbehaves, the server may link any future connection from this user within the current linkability window (e.g., the same day). Consider Figure2 as anexample: A user connects and misbehaves at a server during time period t! within linkability window w!. The server later detects this misbehavior and complains to the NM in time period tc (t! < tc " tL) of the same linkability window w!. As part of 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 + 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 the rest of the day, for example (the linkability window). Note that the user's connections in t1,t2,... ,t$^{!}$,t$^{!}$ + 1,... tc remain unlinkable (i.e., including those since the misbehavior and until the timeof complaint). Even though misbehaving users can be blocked from making connections in the future, the users' past connections remain unlinkable, thus providing backward unlinkability and subjective blacklisting.

F. Notifying the user of blacklist status:

Users who make use of anonymizing networks expecttheir connections to be anonymous. If a server obtains a seed for that user, however, it can link that user's subsequent connections. It is of utmost importance, then,that users be notified of their blacklist status before theypresent a nymble ticket to a server. In our system, theuser can download the server's blacklist and verify herstatus. If blacklisted, the user disconnects immediately.

Since the blacklist is cryptographically signed by the NM, the authenticity of the blacklist is easily verified if the blacklist was updated in the current time period (only one update to the blacklist per time period is allowed). If the blacklist has not been updated in the cur-rent time period, the NM provides servers with "daisies" every time period so that users can verify the freshnessof the blacklist ("blacklist from time period $_{told}$ is fresh as of time period $_{tnow}$"). As discussed in Section 4.3.4, these daisies are elements of a hash chain, and provide a lightweight alternative to digital signatures. Using digital signatures and daisies, we thus ensure that race conditions are not possible in verifying the freshness of a blacklist. 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.

III. SECURITY MODEL

A. Goals and threats:

An entity is honest when its operations abide by the system's specification. An honest entity can be curious: it attempts to infer knowledge from its own information (e.g., its secrets, state, and protocol communications). An honest entity becomes corrupt when it is compromisedby an attacker, and hence reveals its information at the time of compromise, and operates under the attacker's full control, possibly deviating from the specification.

B. Blacklistability:

Assures that any honest server can indeed block misbehaving users. Specifically, if an honest server complains about a user that misbehaved in the current linkability window, the complaint will be successful and the user will not be able to "nymble-connect," i.e., establish a Nymble-authenticated connection, to the server successfully in subsequent time periods (following the time of complaint) of that linkability window.

CONFERENCE PAPER
"A National Level Conference on Recent Trends in Information Technology and Technical Symposium" On 09th March 2013
Organized by
Dept. of IT, Jawaharlal Darda Inst. Of Eng. & Tech., Yavatmal (MS), India

C. Rate-limiting:

Assures any honest server that no user can successfully nymble-connect to it more than once within any single time period.

D. Non-f rameability:

Guarantees that any honest user who is legitimate according to an honest server can nymble-connect to that server. This prevents an attacker from framing a legitimate honest user, e.g., by getting the user blacklisted for someone else's misbehavior. Thisproperty assumes each user has a single unique identity.When IP addresses are used as the identity, it is possible for a user to "frame" anhonest user who later obtains the same IP address. Non-frameability holds true only against attackers with different identities (IP addresses).

A user is legitimate according to a server if she has not been blacklisted by the server, and has not exceeded the rate limit of establishing Nymble-connections. Honest servers must be able to differentiate between legitimate and illegitimate users.

Table: 1

Who	Whom	How	What	**Algorithm 1** PMCreatePseudonym
Servers	PM & NM	honest	*Blacklistability*	**Input:** (uid,w)# H × N
			&Rate-limiting	**Persistent state:** pmState # SP
Users	PM & NM	honest	*Non-frameability*	**Output:** pnym # P
Users	PM	honest	*Anonymity*	1: Extract $nymKey_P$, $macKey_{NP}$ from pmSt
Users	NM	honest &	*Anonymity*	2: nym := MA. Mac(uid||w , nymKeyP)
		not curious		3: mac := MA.Mac(nym||w,mac Key_{NP})
Users	PM or NM	honest	*Non-identification*	4: **return** pnym := (nym, mac)

E. 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.

F. Trust assumptions:

We allow the servers and the users to be corrupt and controlled by an attacker. Not trusting these entities is important because encountering a corrupt server and/oruser is a realistic threat. Nymble must still attain its goals under such circumstances. With regard to the PM and NM, Nymble makes several assumptions on *who* trusts *whom* to be*how* for*what* guarantee. We summarize thesetrust assumptions as a matrix in Figure3. Should a trust assumption become invalid, Nymble will not be able to provide the corresponding guarantee.

For example, a corrupt PM or NM can violate *Black-listability* by issuing different pseudonyms or credentials to blacklisted users. A dishonest PM (resp. NM) can *frame* a user by issuing her the pseudonym (resp. credential) of another user who has already been blacklisted. To undermine the *Anonymity* of a user, a dishonest PM (resp. NM) can first impersonate the user by cloning her pseudonym (resp. credential) and then attempt to authenticate to a server—a successful attempt reveals that the user has already made a connection to the server during the time period. Moreover, by studying thecomplaint log, a curious NM can deduce that a user has connected more than once if she has been complained about two or more times. As already described in Section2.3, the user must trust that at least the NM or PM is honest to keep the user and server identity pair private.

G. Cryptographic primitives:

Nymble uses the following building blocks :

a. Secure cryptographic hash functions. These are one-way and collision-resistant functions that resemble random oracles [5]. Denote the range of the hash functions by H.

b. Secure message authentication (MA) [3]. These consist of the key generation (MA.KeyGen), and the message authentication code (MAC) computation (MA.Mac) algorithms. Denote the domain of MACs by M .

c. Secure symmetric-key encryption (Enc) [4]. These consist of the key generation (Enc.KeyGen), encryption (Enc.Encrypt), and decryption (Enc.Decrypt) algorithms. Denote the domain of ciphertexts by Γ .

d. Secure digital signatures (Sig) [22]. These consist of the key generation (Sig.KeyGen), signing (Sig.Sign), and verification (Sig.Verify) algorithms. Denote the domain of signatures by Σ .

H. Pseudonyms:

The PM issues pseudonyms to users. A pseudonym pnym has two components nym and mac: nym is a pseudo-random 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 $nymKey_P$;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.

I. Seeds and nymbles

Anymble is a pseudo-random number, which serves as an identifier for a particular time period. Nymbles (presented by a user) across periods are unlinkable unless aserver has blacklisted that user. Nymbles are presented as part of a nymble ticket, as described next.As shown in Figure 4, seeds evolve throughout a linkability window using a seed-evolution function f; the seed for the next time period ($seed_{next}$) is computed from the seed for the current time period ($seed_{cur}$) asseed n e x t = f(seedc u r).

The nymble ($nymble_t$) for a time period t is evaluated

byapplying the nymble-evaluation function g to its corresponding seed (seedt), i.e.,$nymble_t = g(seed_t)$.

J. Nymble tickets and credentials:

A credential contains all the nymble tickets for a particular linkability window that a user can present to a particular server. Algorithm 3 describes the following procedure of generating a credential upon request. A ticket contains a nymble specific to a server, time period, and linkability window. ctxt is encrypted data that the NM can use during a complaint involving the nymble ticket. 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.

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.

K. Blacklists:

A server's blacklist is a list of nymbles 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.

L. Blacklist integrity:

It is important for users to be able to check the integrity and freshness of blacklists, because otherwise servers could omit entries or present older blacklists and link users without their knowledge. The NM signs the blacklist, along with the server identity sid, the current time period t, current linkability window w, and target (used for freshness, explained soon), using its signing key $signKey_N$. As will be explained later, during a complaint procedure, the NM needs to update the server's blacklist, and thus needs to check the integrity of the blacklist presentedby the server. To make this operation more efficient, theNM also generates a MAC using its secret key $macKey_N$ (line 3). At the end of the signing procedure, the NM returns a blacklist certificate (line 6), which contains the time period for which the certificate was issued, a*daisy* (used for freshness, explained soon), mac and sig.Algorithms8 and9 describe how users and the NM can verify the integrity and freshness of blacklists.

M. Blacklist freshness :

If the NM has signed the blacklist for the current time period, users can simply verify the digital signature in the certificate to infer that the blacklist is both valid (not tampered with) and fresh (since the current time period matches the time period in the blacklist certificate). To prove the freshness of blacklists every time period, however, the servers would need to get the blacklists digitally signed every time period, thus imposing a high load on the NM. To speedup this process, we use a hash chain [20], [29] to certify that "blacklist from time period t is still fresh." For each complaint, the NM generates a new random seed daisyL for a hash chain corresponding to time period L. It then computes ^{daisy}L-1,daisyL-2, . $daisy_t$up to current time period t by successively hashing theprevious daisy to generate the next with a cryptographic hash function h.

N. Complaints and linking tokens:

A server complains to the NM about a misbehaving userby submitting the user's nymble ticket that was used inthe offending connection. The NM returns a seed, fromwhich the server creates alinking token, which contains the seed and the corresponding nymble.

Each server maintains a list of linking tokens in a linking-list, and updates each token on the list every timeperiod. When a user presents a nymble ticket, the serverchecks the nymble within the ticket against the nymblesin the linking-list entries. A match indicates that the user has been blacklisted.

O. Communication channels:

Nymble utilizes three types of communication channels, namely type-Basic, -Auth and -Anon (Figure 6).

We assume that a public-key infrastructure (PKI) suchas X.509 is in place, and that the NM, the PM and all theservers in Nymble have obtained a PKI credential froma well-established and trustworthy CA. (We stress thatthe users in Nymble, however, need not possess a PKIcredential.) These entities can thus realize type-Basic andtype-Auth channels to one another by setting up a TLS[8] connection using their PKI credentials.

All users can realize type-Basic channels to the NM, thePM and any server, again by setting up a TLS connection. Additionally, by setting up a TLS connection over the Tor anonymizing network,[9] users can realize a type-Anon channel to the NM and any server.

IV. NYMBLE CONSTRUCTION

A. System setup:

During setup, the NM and the PM interact as follows.

a. The NM executes NMInitState() and initializes its state nmState to the algorithm's output.
b. The NM extracts $macKey_{NP}$ from nmState and sends it to the PM over a type-*Auth* channel. $macKey_{NP}$ is a shared secret between the NM andthe PM, so that the NM can verify the authenticity of pseudonyms issued by the PM.
c. The PM generates nymKeyP by running Mac.KeyGen() and initializes its state pmState to the pair ($nymKey_P$, $macKey_{NP}$).
d. The NM publishes $verKey_N$ in nmState in a way that the users in Nymble can obtain it and verify its integrity at any time (e.g., during registration).

B. Server registration:

To participate in the Nymble system, a server with identity sid initiates a type-*Auth* channel to the NM, andregisters with the NM according to the*Server Registration* protocol below. Each server may register at most once in any linkability window.

a. The NM makes sure that the server has not already registered: If (sid,·,·)# nmEntries in its nmState, it terminates with failure; it proceeds otherwise.
b. The NM reads the current time period and linkability window as $_{tnow}$ and $_{wnow}$ respectively, and then obtains a svrState by running $NMRegisterServer_{nm}St_at_e$(sid, ^tnow, wnow) .
c. The NM appends svrState to its nmState, sends it

to the *Server*, and terminates with success.

d. The server, on receiving svrState, records it as itsstate, and terminates with success.

C. User registration:

A user with identity uid must register with the PM once each linkability window. To do so, the user initiates a type-*Basic* channel to the PM, followed by the *User Registration* protocol described below.

a. The PM checks if the user is allowed to register. In our current implementation the PM infers the registering user's IP address from the communication channel, and makes sure that the IP address does not belong to a known Tor exit node. If this is not the case, the PM terminates with failure.

b. Otherwise, the PM reads the current linkability window as $_{wnow,}$ and runs pnym:= PMCreate $Pseudonym_{pm}S_{tate}$(uid, wnow).

c. The PM then gives pnym to the user, and terminates with success. The user, on receiving pnym, sets her state usrSt ate to (pnym, $) , and terminates with success.

D. Credential acquisition:

To establish a Nymble-connection to a server, a user must provide a valid ticket, which is acquired as part of a credential from the NM. To acquire a credential for server sid during the current linkability window, a registered user initiates a type-*Anon* channel to the NM, followed by the *Credential Acquisition* protocol below.

a. The user extracts pnym from usrState and sends the pair (pnym, sid) to the NM.

b. The NM reads the current linkability window as wnow.It makes sure the user's pnym is valid: If $NMVerifyPseudonym_{nm}S_{tate}$(pnym, wnow) returns **false,** the NM terminates with failure; it proceeds otherwise.

c. The NM runs NMCreateCredentialn m S t a t e (pnym, sid, wnow),which returns a credential cred. The NM sends cred to the user and terminates with success.

d. The user, on receiving cred, creates usrEntry : = (sid, cred,**false),** appends it to its state usrState, and terminates with success.

E. Service provision and access logging:

If both the user and the server terminate with success inthe*Nymble-connection Establishment* described above, theserver may start serving the user over the same channel.The server records ticket and logs the access during the session for a potential complaint in the future.

F. Auditing and filing for complaints:

If at some later time the server desires to blacklist the user behind a Nymble-connection, during the establishment of which the server collected ticket from the user, the server files a complaint by appending ticket to cmplnt-tickets in its svrState.

G. Blacklist update:

Servers update their blacklists for the current time periodfor two purposes. First, as mentioned earlier, the serverneeds to provide the user with its blacklist (and blacklist certificate) for the current time period during a Nymble connection establishment. Second, the server needs to beable to blacklist the misbehaving users by processing the newly filed complaints (since last update).

The procedure for updating blacklists (and their certificates) differs depending on whether complaints are involved. When there is no complaint (i.e., the server's cmplnt-tickets is empty), blacklists stay unchanged; the certificates need only a "light refreshment." When there are complaints, on the other hand, new entries are added to the blacklists and certificates need to be regenerated. Since these updates are certified for integrity and freshness at the granularity of time periods, multiple updates within a single time period are disallowed (otherwise servers could send users stale blacklists).

V. SECURITYANALYSIS

An honest PM and NM will issue a coalition of c unique users at most c valid credentials for a given server. Because of the security of HMAC, only the NM can issue valid tickets, and for any time period the coalition has at most c valid tickets, and can thus make at most c connections to the server in any time period regardless of the server's blacklisting. It suffices to show that if each of the c users has been blacklisted in some previous time period of the current linkability window, the coalition cannot authenticate in the current 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 previoustime period k#. Since connection establishments k# andk! were successful, the corresponding tickets $ticket^{\#}$ 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 during the blacklist update. Since the server blacklisted the valid $ticket^{\#}$ and updates its linking list honestly, the Server Link Ticket will return fail on input $ticket^{!}$, and thus the connection k! must fail, which is a contradiction.

A. Non-Frameability:

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 =& i!, then user ˜i's seed0 is different from user i!'sseed0 so long as the PM is honest, and yet the two seed0'sevolve to the same $seed^{!}$, which contradicts the collision-resistance property of the evolution function. Thus we have ˜i = 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.

B. Anonymity:

Distinguishing between two illegitimate users We argue

CONFERENCE PAPER
"A National Level Conference on Recent Trends in Information Technology and Technical Symposium" On 09th March 2013
Organized by
Dept. of IT, Jawaharlal Darda Inst. Of Eng. & Tech., Yavatmal (MS), India

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 untilthe end of the *Blacklist validation* stage , itsuffices 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 anhonest NM never deletes entries from a blacklist, it will appear in all subsequent blacklists, and safe is evaluated to **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).

VI. CONCLUSIONS

Nymble, which can be used to add a layer of accountability to any publicly known anonymizing network. Servers can blacklist misbehaving users while maintaining their privacy, and we show how these properties can be attained in a way that is practical, efficient, and sensitive to needs of both users and services will increase the mainstream acceptance of anonymizing networks such as Tor, which has thus far been completely blocked by several services because of users who abuse their anonymity.

VII. REFERENCES

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. InCRYPTO, LNCS 1880, pages 255–270. Springer, 2000.

[2] G. Ateniese, D. X. Song, and G. Tsudik. Quasi-Efficient Revocation in Group Signatures. InFinancial Cryptography, LNCS 2357, pages 183–197. Springer, 2002.

[3] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. InCRYPTO, LNCS 1109, pages 1–15. Springer, 1996.

[4] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In FOCS, pages 394–403, 1997.

[5] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In Proceedings of the 1st ACM conference on Computer and communications security, pages 62–73. ACM Press, 1993.

[6] M. Bellare, H. Shi, and C. Zhang. Foundations of Group Sig-natures: The Case of Dynamic Groups. In CT-RSA, LNCS 3376, pages 136–153. Springer, 2005.

[7] D. Boneh and H. Shacham. Group Signatures with Verifier-Local Revocation. In ACM Conference on Computer and Communications Security, pages 168–177. ACM, 2004.

[8] S. Brands. Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). In CRYPTO, LNCS 773, pages 302–318. Springer, 1993.

[9] E. Bresson and J. Stern. Efficient Revocation in Group Signatures. In Public Key Cryptography, LNCS 1992, pages 190–206. Springer, 2001.

[10] J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In EUROCRYPT, LNCS 2045, pages 93–118. Springer, 2001.

CONFERENCE PAPER
"A National Level Conference on Recent Trends in Information Technology and Technical Symposium" On 09th March 2013
Organized by
Dept. of IT, Jawaharlal Darda Inst. Of Eng. & Tech., Yavatmal (MS), India