Extending Nymble-like Systems

Ryan Henry and Ian Goldberg
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1
{rhenry,iang}@cs.uwaterloo.ca

***Abstract*—We present several extensions to the Nymble framework for anonymous blacklisting systems. First, we show how to distribute the Verinym Issuer as a threshold entity. This provides liveness against a threshold Byzantine adversary and protects against denial-of-service attacks. Second, we describe how to revoke a user for a period spanning multiple linkability windows. This gives service providers more flexibility in deciding how long to block individual users. We also point out how our solution enables efficient blacklist transferability among service providers. Third, we augment the Verinym Acquisition Protocol for Tor-aware systems (that utilize IP addresses as a unique identifier) to handle two additional cases: 1) the operator of a Tor exit node wishes to access services protected by the system, and 2) a user's access to the Verinym Issuer (and the Tor network) is blocked by a firewall. Finally, we revisit the objective blacklisting mechanism used in Jack, and generalize this idea to enable objective blacklisting in other Nymble-like systems. We illustrate the approach by showing how to implement it in Nymble and Nymbler.**

***Keywords*-privacy enhancing technologies; anonymity; authentication; anonymous blacklisting; privacy-enhanced revocation.**

I. Introduction

In [21], [35], Tsang *et al.* proposed Nymble as a solution to the problem of allowing service providers on the Internet—such as websites, IRC networks or mail servers—to revoke access from individual misbehaving users of anonymous communications networks. Nymble uses a novel construction to build mutually unlinkable (and efficiently verifiable) authentication tokens for users of anonymous communications networks, while empowering service providers with access revocation capabilities comparable to what they have with nonanonymous users. In particular, the scheme implements a privacy-preserving analog of IP address banning for users of anonymous communications networks. Under some assumptions regarding noncollusion of certain third parties, their approach is provably secure (in the random oracle model); i.e., privacy and availability for honest users are not adversely affected, and blacklisted users remain anonymous. The construction used in Nymble results in an extremely lightweight solution for all parties involved (most notably, for the service provider). It does this, however, by placing a lot of trust in third parties. Since Nymble was first proposed in 2006, several schemes have appeared in the literature to solve the same problem, or one of several closely related problems. (For some examples, see [6], [19], [20], [22], [23], [32]–[34].) Three of these schemes operate within the same general framework as Nymble; they change only low-level details to weaken trust assumptions and to provide stronger privacy guarantees and some new functionality. Moreover, further incarnations of the idea seem likely to emerge in the future [18]. In this paper, we present several extensions to the abstract Nymble framework that can be used to improve the security, liveness and functionality of Nymble-like systems; our extensions solve several open problems identified in the future work sections of [19]–[22].

We first review how the Nymble framework works. Nymble makes use of two trusted third parties (TTPs) called the **Pseudonym Manager** (PM) and the **Nymble Manager** (NM). Together, the PM and the NM issue a **user** (U) with a set of mutually unlinkable, use-once authentication tokens (called **nymbles**). This enables U to access the services offered by a **Service Provider** (SP), while preserving the ability of the SP to block U in the event that U misbehaves. Nymble divides time into fixed intervals called **linkability windows**, which it further subdivides into smaller intervals called **time periods**. When U wishes to use the system, she first connects directly (i.e., not through an anonymous communications network) to the PM; this proves to the PM that U is in possession of a particular IP address. The PM then issues U with a pseudonym (called a **Nym**), which is computed by applying a one-way function (an HMAC with a secret key) to U's IP address. When U wishes to authenticate with some SP, she connects anonymously (over Tor, for example) to the NM and presents a copy of her pseudonym and the **canonical name** of the SP. Based on these two values (Nym and canonical name), the NM computes and issues to U a set of nymbles. Each nymble is valid for a particular place (an SP) and time (a time period within the current linkability window). To be sure, nymbles are not entirely unlinkable; instead, the nymble construction places a trapdoor within each nymble that allows the NM to, given a nymble, compute all *subsequent* nymbles in the same linkability window. If U somehow abuses the services offered by the SP, then the SP can transmit the nymble used by U during that session to the NM. The NM will then use knowledge of the trapdoor to compute all of U's subsequent nymbles for the remainder of the current linkability window. For each remaining time period, the NM will then place the corresponding nymble on a list called the **linking list** (there is a different linking list for each time period) and U's **SP-specific pseudonym** (i.e., her last nymble of the linkability window) on the SP's **blacklist**. By consulting the

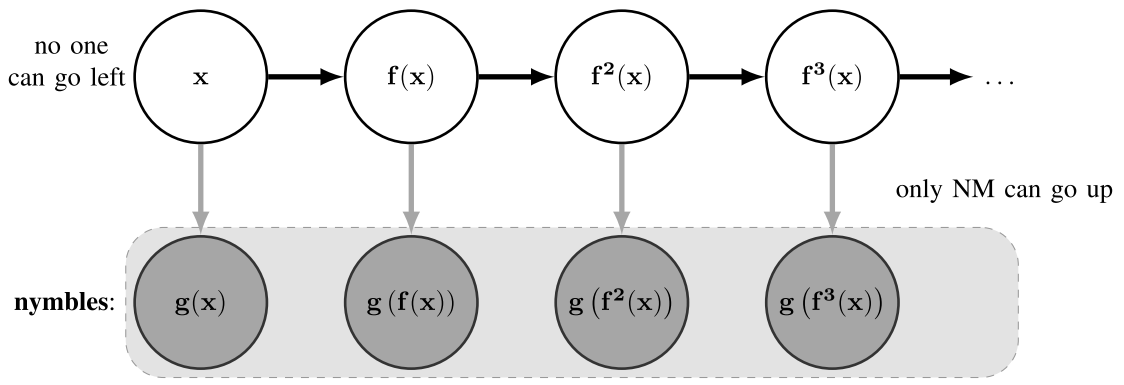


Figure 1. **The nymble construction procedure:** In the original Nymble [21], **black** arrows (i.e., $f(\cdot)$) are implemented with an HMAC and **grey** arrows (i.e., $g(\cdot)$) are implemented with symmetric key encryption.

blacklist, U can easily check her revocation status before she attempts to authenticate. Similarly, by consulting the current linking list and denying access to any user that attempts to authenticate with a nymble on it, the SP can prevent U from further abusing its services for the remainder of the current linkability window. Figure 1 illustrates the basic nymble construction.

In [18], we observed that the pseudonym issued by the PM (or the *Credential Manager* (CM), as it was called in the later Nymble-like systems Nymbler [19] and Jack [22]) in existing schemes is really a *verinym*.[1] This observation highlights the largest security risk in the original Nymble: a malicious PM and NM can easily collude to learn which users are connecting to which SPs. If an SP also colludes, they can also determine what each user is doing when she interacts with that SP. To ensure that the verinym and nymble constructions in future Nymble-like systems are not susceptible to this sort of attack, we proposed a set of two new security properties called the **ZK-verinym property** and the **ZK-pseudonym property**.[2] In our formalization, an entity called the **Verinym Issuer** (VI) replaces the PM; the VI can satisfy the new properties by using a technique first used by Henry *et al.* in [19]. First, a distributed VI issues to U an anonymous credential that encodes U's verinym. To access an SP's services, U computes her own nymbles and uses zero-knowledge proofs (ZKPs) to convince the NM of their correctness. (U reveals only a commitment to her verinym.)

We further point out that the NM in existing schemes fills two distinct roles. For this reason, we propose to replace the NM with two separate entities: the **Nymble Issuer** (NI), who issues nymbles to users, and the **Pseudonym Extractor** (PE) who uses the trapdoor function to revoke misbehaving users. Figure 2 illustrates the various parties involved in the Nymble framework and the interactions among them.

We refer the reader to [18] for a more thorough description of the Nymble framework and Nymble-like systems.

[1]A **verinym** is any piece of identifying information that can single you out of a crowd of potential candidates [16].

[2]The *ZK-verinym property* says that no party other than U (including the PM and NM) learns the verinym associated with U. The *ZK-pseudonym property* says that no party other than the NM is capable of recovering U's pseudonym for a given SP from one of U's nymbles. We refer the reader to [18] for formal definitions of these properties.

A. Security requirements

A secure Nymble-like system must satisfy the following security requirements [18]:

Correctness:
An honest SP will accept any well-formed nymble from an unrevoked user.

Misauthentication resistance:
An honest SP will only accept nymbles that are output by a correct execution of the system's protocols; i.e., it should be infeasible to
- forge a verinym without the VI's secret key, or
- forge a nymble without the NI's secret key.

Backward anonymity:
It is infeasible to associate a user's nymbles with her real identity, even if this user is, or later becomes, revoked.

Unlinkability:
It is infeasible for anyone but the PE to determine if two or more distinct nymbles come from the same user or from different users. If these nymbles come from different SPs or linkability windows, then even the PE should not be able to determine if they come from the same user or from different users.

Revocability:
With the assistance of the PE, an SP can blacklist any user in such a way that no coalition of blacklisted users can later authenticate.

Revocation auditability:
A user can check her revocation status prior to revealing any nymbles to an SP.

Non-frameability:
No coalition of third parties can convince an honest SP to blacklist a user for any action that user is not responsible for.

We refer the reader to [18] for more rigorous formal definitions of these security notions.

B. Performance requirements

In addition to the security requirements we have already mentioned, a *useful* Nymble-like system must also satisfy the following performance requirements [18]:

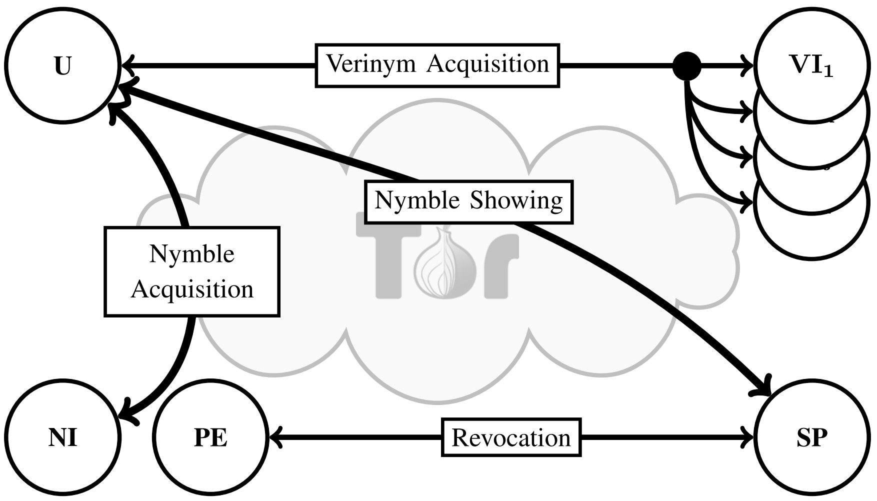


Figure 2. **Nymble framework architecture:** This figure illustrates the various parties involved in the Nymble framework, and the interactions among them.

Verifier efficiency:
The value of the system to an SP must be higher than its cost. Many SPs place little value in the input of anonymous users, so the cost of supporting them must be extremely low. This includes storage, bandwidth and computation, as well as hardware costs.

User efficiency:
The system should be available to all users, and it should avoid adding noticeable latency to the user's interactions with an SP.

Again, we refer the reader to [18] for more rigorous formal definitions of these performance notions.

C. Our contributions

The remainder of this paper proposes some useful extensions to the Nymble framework. These extensions improve the liveness, security and functionality of Nymble-like schemes built from the extended framework; they solve several open problems identified in the future work sections of existing literature on anonymous blacklisting systems [19]–[22].

Our first contribution is to describe how to build a fully distributed threshold VI; our construction satisfies the ZK-verinym property and is ideally suited to a nymble format that satisfies the ZK-pseudonym property. This modification provides security and liveness against a threshold Byzantine adversary.

The second contribution amends the *Nymble Acquisition Protocol* of systems satisfying the ZK-verinym and ZK-pseudonym properties and base verinyms deterministically on a unique resource (e.g., Nymbler [19]) to support revocation of a user for a duration that spans multiple linkability windows. This gives service providers flexibility in deciding how long to block an individual misbehaving user. We point out how our solution enables blacklist transferability among service providers and outline how this construction is efficiently implemented using a generic nymble format.

Next, we augment the *Verinym Acquisition Protocol* for Tor-aware Nymble-like systems (that utilize IP addresses as a unique identifier) to handle two additional cases: 1) when the operator of a Tor exit node wishes to access services protected by the system, and 2) when a user's access to the Verinym Issuer (and the Tor network) is blocked by a firewall. This latter solution leverages Tor's existing bridge infrastructure [10], and a similar idea of using a distributed collection of simple entities, called Identity Verifiers, that assist users by verifying their IP addresses.

Finally, we revisit the objective blacklisting mechanism used by Lin and Hopper in Jack [22]. We generalize their approach to enable similar objective blacklisting capabilities in other Nymble-like system. We illustrate the approach by showing how to implement it in Nymble [21] and Nymbler [19].

II. Distributed (t, s)-threshold Verinym Issuer

The key idea in our distributed VI is to have the VIs use a distributed "unique" (t, s)-threshold signature scheme to compute U's verinym. The security of this approach relies on two properties of the underlying threshold signatures: *unforgeability* and *uniqueness*.[3] All secure signature schemes provide unforgeability [17]; uniqueness, on the other hand, is a property that is only possessed by certain signature schemes.

Definition 1 (Unique Signature Scheme [24]). *A signature scheme is called* ***unique*** *if, for every (possibly maliciously chosen) public key pk and every message msg, there exists at most one signature σ such that $\mathtt{Ver}_{pk}(msg, \sigma) = \mathtt{true}$.*

For completeness, we formally define a (t, s)-threshold signature scheme before discussing our approach in detail.

[3]The uniqueness property is only required if verinyms are computed deterministically from a user's unique resource, as in Nymble [21] and Nymbler [19]. Systems such as Jack [22] and BNymble [23] in which verinyms are based on user-chosen randomness do not require such a property.

Definition 2 ((t,s)-threshold Signature Scheme)**.** *A* $\boldsymbol{(t,s)}$***-threshold signature scheme*** *is a digital signature scheme with s signers and the property that any subset of at least t signers can cooperate to sign a message msg. Conversely, any subset of fewer than t signers should not be able to compute any nontrivial information about a valid signature on msg.*

A ***unique*** $\boldsymbol{(t,s)}$***-threshold signature scheme*** is just a (t,s)-threshold signature scheme with the uniqueness property.

We use the non-interactive threshold RSA signature scheme of Damgård and Koprowski [9] for a concrete realization of this idea. Other choices of unique threshold signature scheme may also work well. We choose Damgård and Koprowski's threshold RSA signatures because: 1) proving knowledge of an RSA signature in zero-knowledge is easy, and 2) the scheme does not require a trusted dealer who knows the factorization of the RSA modulus. This latter point is particularly useful because some schemes (e.g., Nymbler [19]) already make use of an RSA modulus with unknown factorization (and leave details of its generation up to the implementer). Damgård and Koprowski's scheme makes use of a slightly modified form of the Robust Efficient Distributed RSA-Key Generation protocol of Frankel *et al.* [15]. In our case we will require that the public key n is chosen such that $N = 4n+1$ is a prime;[4] this can be accomplished by repeatedly executing the protocol of [15] until a suitable n has been found. The prime number theorem [25, Fact 2.95] tells us that, for example, the protocol will have to be executed an average of $1536 \cdot \ln 2 \approx 1064$ times, for a 1536-bit modulus. Note, however, that key generation occurs infrequently, since we have the distributed VI generate a single n to use for a substantial number of future linkability windows.

The use of signatures in our application presents an interesting challenge; for security purposes, U must be able to prove in zero-knowledge that one committed value is a signature on a second committed value. This means that the VI cannot just sign a hash of the message as is usually done to ensure security and integrity of RSA signatures. Instead, we use a modified version of Rabin's function: $H(z,\xi) = \big(z^2 + (z \bmod \xi)\big) \bmod n$ in place of a hash. We choose this function to prevent U from exploiting the homomorphic properties of RSA encryption. Full details of our approach follow.

As in [18], we subdivide each linkability window into smaller intervals called **verinym validity periods** (VVPs). (The number of time periods in a linkability window will be a multiple of the number of VVPs.) Thus, when the VI issues the user a verinym, this verinym is only valid until some future VVP; the NI will not issue any nymble to a user for any time period in a VVP after her verinym expires.

[4]The protocol of [15] with the modification suggested in [9, §5] produces an RSA modulus $n = p_n q_n$ such that $\gcd\left((p_n-1)/2, s!\right) = \gcd\left((q_n-1)/2, s!\right) = 1$; thus, it is easy to see that $2n+1$ is always divisible by 3. $3n+1$ is even, so we use $N = 4n+1$.

A. Threshold signature acquisition

Let $\mathbf{VI} = \{\mathrm{VI}_1, \mathrm{VI}_2, \ldots, \mathrm{VI}_s\}$ be a set of s VIs. Let ℓ_{hash} be the bit length of the output of some secure cryptographic hash function $hash$ (say, $\ell_{\text{hash}} = 256$). U's unique identifier is z and $y = hash(z)$. There are $K_{\max}$ VVPs per linkability window and the current VVP is K_{cur}. A verinym is valid for at most K_{lim} VVPs but will expire earlier if $K_{\text{cur}} + K_{\text{lim}} \geq K_{\max}$.

The s VIs initialize the system by first jointly computing an RSA modulus $n = p_n q_n$, such that p_n, q_n, and $N = 4n+1$ are all prime, by using the 'Distributed Computation of N' protocol from [15, §10] with the modification suggested in [9, §5].

Choosing a set of public keys: Suppose the public modulus n will be retired after $L_{\max}$ linkability windows. The VIs agree on a prime $\eta > s$ such that $\lceil \log_2(\eta) \rceil + hamming\_weight(\eta)$ is the smallest possible. (If $s < 17$ then $\eta = 17$ is a good choice.) They also agree on a product E of $\eta^{K_{\max}-1}$ and $L_{\max}$ distinct primes, $E = \eta^{K_{\max}-1} \cdot \prod_{L=0}^{L_{\max}-1} e_L$, such that $e_L > s$ for all $0 \leq L < L_{\max}$ and $\lceil \log_2(E) \rceil < \lceil \log_2(n) \rceil - 2$ (so that $E < \varphi(n)$). As with η, each e_L should be chosen with $\lceil \log_2(e_L) \rceil + hamming\_weight(e_L)$ as low as possible. The VIs also choose ξ, a publicly known (random) ℓ_{hash}-bit integer.

Generating the set of private keys: Once E has been chosen, the VIs use the 'Robust Distributed Generation of Public and Private Keys' protocol of [15, §9] to find the private key exponent $D = E^{-1} \bmod \varphi(n)$. After executing the protocol, the public key for linkability window L^* is then $(n, e_{L^*}, v, \eta, K_{\max})$, where v is a verification value; the private key is $d_{L^*} = D \cdot \prod_{L \neq L^*} e_L = D \cdot E/(e_{L^*} \cdot \eta^{K_{\max}-1})$. Each $\mathrm{VI}_i \in \mathbf{VI}$ has a share s_i of D, and a public verification key $v_i = v^{s_i (s!)^2} \bmod n$. For the remainder of this paper, we shall assume that we are working in a fixed linkability window L_{cur}, whose corresponding public and private keys are simply denoted by e and d, respectively, instead of $e_{L_{\text{cur}}}$ and $d_{L_{\text{cur}}}$.

Deriving a public-private key pair: In the first VVP, the public key is e and the private key is d; thus, VI_i uses the share $s_i \cdot (E/e)$ to compute signatures. In the next VVP, the public key is $e \cdot \eta$ and the private key shares are $s_i \cdot (E/(e \cdot \eta))$. In general, in VVP K, $0 \leq K < K_{\max}$, the public key is $e \cdot \eta^K$ and the private key shares are $s_i \cdot (E/(e \cdot \eta^K))$. Note that, if U obtains a signature that is valid in VVP K^*, then it can easily be **backdated**; that is, turned into a signature that is valid in any *earlier* VVP K' of the same linkability window by raising it to the power $\eta^{K^*-K'}$ and reducing modulo n: if $(x)^{e \cdot \eta^{K^*}} \equiv Y \bmod n$ then $(x^{\eta^{K^*-K'}})^{e \cdot \eta^{K'}} \equiv Y \bmod n$. On the other hand, U is unable to produce a signature for a *later* VVP since this requires the computation of $\eta^{-1} \bmod \varphi(n)$.

We model the **Verinym Acquisition Protocol** after the signing protocol from [9, §3]. However, we make two significant changes in our version: 1) the VIs produce a threshold signature on the value $Y = \big(y^2 + (y \bmod \xi)\big) \bmod n$ instead of on y directly, and 2) the VIs modify their key shares as needed to produce the correct signature for a particular linkability window and VVP. (The verification equations that U uses to check the correctness of each share also change slightly as a result.) The protocol is initiated by U and executed with at least t different VIs.

Because U runs the Verinym Acquisition Protocol with at least t different VIs, it is possible for two or more VIs to compute their shares for different expiration VVPs. (This might happen because the protocols execution spans the boundary between two VVPs, or because two different VIs have different policies about K_{lim}.) Of course, U could request a verinym with a particular expiration VVP from each VI; however, we instead have each VI issue a share for the latest VVP that their policy will allow. Once U has obtained all verinym shares, she then backdates each share (to the latest VVP in which all shares are valid) before constructing her verinym.

U does the following:

1. U chooses a random size-t subset of VIs, say $S = \{\text{VI}_{i_1}, \ldots, \text{VI}_{i_t}\} \subseteq_R \mathbf{VI}$.
2. U connects (directly, if z is U's IP address; anonymously, otherwise) to each $\text{VI}_{i_j} \in S$ and requests a verinym. (If z is U's IP address, then this step proves to VI_{i_j} that U possesses z; for other unique identifiers, U issues some form of proof of possession to VI_{i_j}.)

VI_{i_j} does the following:

3. VI_{i_j} receives the request for a verinym from U, with identifier z. It notes the current VVP K_{cur} and computes $y = hash(z)$, $Y = (y^2 + (y \bmod \xi)) \bmod n$ and $K_{i_j} = \min\{K_{\max} - 1, K_{\text{cur}} + K_{\text{lim}}\}$.
4. VI_{i_j} computes the signature share
$$X_{i_j} = Y^{2(s!)^2 \cdot s_{i_j} \cdot (E/(e \cdot \eta^{K_{i_j}}))} \bmod n.$$
5. VI_{i_j} computes a proof of correctness for X_{i_j} by choosing $r \in_R \{0, \ldots, 4\kappa_1 + (12s + 4)\lg s\}$, where κ_1 is a security parameter (see [9], [14], [31]), and computing
$$c_{i_j} = hash(v,\ \tilde{y},\ v_{i_j},\ X_{i_j}^2,\ v^{r(s!)^2},\ \tilde{y}^r),$$
and $g_{i_j} = s_{i_j} \cdot (E/(e \cdot \eta^{K_{i_j}})) \cdot c_{i_j} + r$, where $\tilde{y} = Y^{4(s!)^2}$. The proof is (g_{i_j}, c_{i_j}).
6. VI_{i_j} sends $(X_{i_j}, g_{i_j}, c_{i_j}, K_{i_j})$ to U.

U does the following:

7. U receives $(X_{i_j}, g_{i_j}, c_{i_j}, K_{i_j})$ from each VI_{i_j}.
8. U verifies each share by checking
$$c_{i_j} \stackrel{?}{=} hash(v, \tilde{y}, v_{i_j}, X_{i_j}^2, v^{g_{i_j}(s!)^2} \cdot v_{i_j}^{-c_{i_j} \cdot (E/(e \cdot \eta^{K_{i_j}}))}, \tilde{y}^{g_{i_j}} \cdot X_{i_j}^{-2c_{i_j}}).$$
If verification fails for any i_j, then U: 1) discards X_{i_j}, 2) selects $\text{VI}_{i_{j'}} \in_R \mathbf{VI} - S$, 3) sets $S = (S \cup \{\text{VI}_{i_{j'}}\}) - \{\text{VI}_{i_j}\}$, and 4) executes from Step 2 for $\text{VI}_{i_{j'}}$.
9. Let $K_{\text{exp}} = \min\{K_{i_j} \mid \text{VI}_{i_j} \in S\}$. (So K_{exp} is the latest VVP for which all verinym shares are valid.) For each $1 \le j \le t$, if $K_{i_j} > K_{\text{exp}}$, then U backdates X_{i_j} to get $X'_{i_j} = X_{i_j}^{\eta^{(K_{i_j} - K_{\text{exp}})}}$; otherwise, U sets $X'_{i_j} = X_{i_j}$. Each X'_{i_j} is now a verinym share for VVP K_{exp}.
10. U recombines her shares as follows:
    a) U computes
    $$\omega = \prod_{\text{VI}_{i_j} \in S} (X'_{i_j})^{2\lambda_{i_j}} = Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})} \bmod n$$
    where λ_{i_j} is the integer
    $$\lambda_{i_j} = (s!) \cdot \prod_{\{i | \text{VI}_i \in S\} - \{i_j\}} \frac{i}{i - i_j}.$$
    b) U uses the Extended Euclidean Algorithm to find a and b such that
    $$a \cdot 4(s!)^5 + b \cdot e \cdot \eta^{K_{\text{exp}}} = 1.$$
    c) U computes her verinym by first computing the signature $x_{K_{\text{exp}}}$:
    $$\begin{aligned} x_{K_{\text{exp}}} &= \omega^a Y^b \\ &= \big(Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}\big)^a Y^b \\ &= \big(Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}\big)^a \big(Y^{e \cdot d}\big)^b \\ &= \big(Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}\big)^a \big(Y^{(e \cdot \eta^{K_{\text{exp}}}) \cdot d/(\eta^{K_{\text{exp}}})}\big)^b \\ &= \big(Y^{a \cdot 4(s!)^5 + b \cdot e \cdot \eta^{K_{\text{exp}}}}\big)^{d/(\eta^{K_{\text{exp}}})} \\ &= (Y)^{d/(\eta^{K_{\text{exp}}})} \\ &= (y^2 + (y \bmod \xi))^{d/(\eta^{K_{\text{exp}}})} \bmod n. \end{aligned}$$

At the conclusion of the Verinym Acquisition Protocol, U outputs a signature $x_{K_{\text{exp}}}$; given this value, she can compute her verinym x_0 by backdating this signature as follows: $x_0 = x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}}}}$. More to the point, she can compute (and prove in zero-knowledge to the NI that)
$$\big(y^2 + (y \bmod \xi)\big) \equiv \Big(x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}}}}\Big)^e \bmod n.$$
Figure 3 illustrates the Verinym Showing Protocol. In this example, U is requesting J nymbles from the NI and her current verinym expires in VVP K_{exp}. In the Nymble Showing Protocol, U is required to prove that her verinym is valid for VVP $K^* = \min\{\lfloor (K_{\max} \cdot (T_{\text{cur}} + J))/T_{\max} \rfloor, K_{\text{exp}}\}$ (i.e., K^* is the VVP containing the last time period associated with her Jth nymble). She backdates her verinym $x_{K_{\text{exp}}}$ (locally) as follows: $x_{K^*} = x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}} - K^*}} \bmod n$. She then commits to y, x_{K^*} and $x_0 = x_{K^*}^{\eta^{K^*}} \bmod n$ and produces a zero-knowledge proof that x_0 is indeed $x_{K^*}^{\eta^{K^*}} \bmod n$ and that $(y^2 + (y \bmod \xi)) \equiv x_0^e \bmod n$. Note that the exponent η^{K^*} in this proof is public. That U is able to compute this proof with exponent η^{K^*} proves to the NI that U's verinym is still valid in VVP K^*. When the NI is convinced by U's proof, the Nymble Acquisition Protocol runs as usual using x_0 as U's verinym.

Let α_n, β_n be known generators of the order-n subgroup modulo N. U invokes the **Verinym Showing Protocol** to prove to the NI that she has a valid verinym. The protocol works as follows:

U does the following:

1. U computes $K^* = \min\{\lfloor (K_{\max} \cdot (T_{\text{cur}} + J))/T_{\max} \rfloor, K_{\text{exp}}\}$, $x_{K^*} = x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}}-K^*}} \bmod n$ and $x_0 = x_{K^*}^{\eta^{K^*}} \bmod n$; VVP K^* is the VVP that contains the last time period for which she is requesting a nymble.
2. U chooses $\gamma_1, \gamma_2, \gamma_3 \in_R \mathbb{Z}_n$ and computes Pedersen [26] commitments $\bar{y} = \alpha_n^y \beta_n^{\gamma_1} \bmod N$, $\overline{x_{K^*}} = \alpha_n^{x_{K^*}} \beta_n^{\gamma_2} \bmod N$ and $\overline{x_0} = \alpha_n^{x_0} \beta_n^{\gamma_3} \bmod N$ and the ZKP $\Pi_{x_{K^*}}$ (using Camenisch-Stadler notation [8])

$$PK\left\{\begin{pmatrix}\gamma_1,\\ \gamma_2,\\ \gamma_3,\\ y,\\ x_0,\\ x_{K^*}\end{pmatrix} : \begin{array}{lr} \bar{y} = \alpha_n^y \beta_n^{\gamma_1} & \bmod N \\ \wedge \overline{x_{K^*}} = \alpha_n^{x_{K^*}} \beta_n^{\gamma_2} & \bmod N \\ \wedge \overline{x_0} = \alpha_n^{x_0} \beta_n^{\gamma_3} & \bmod N \\ \wedge x_0 = x_{K^*}^{\eta^{K^*}} & \bmod n \\ \wedge x_0^e = y^2 + (y \bmod \xi) & \bmod n \\ \wedge 0 \le y < 2^{\ell_{\text{hash}}} & \\ \wedge 0 \le x_{K^*} < n & \end{array}\right\}.$$

3. Finally, U transmits $\bar{y}$, $\overline{x_{K^*}}$ and $\overline{x_0}$, along with $\Pi_{x_{K^*}}$, to the NI.

The NI does the following:

4. The NI receives and verifies $\Pi_{x_{K^*}}$. If verification fails, it aborts; otherwise, it accepts the verinym x_0 as valid.

The remainder of the protocol works just like the scheme with a non-threshold VI. As Figure 3 illustrates, U computes the seed value h_0 for her nymbles with a one-way function $F(x_0, \text{SP})$. She iteratively applies a second one-way function f to h_0 a total of T_{cur} times to compute the first nymble seed in her chain. From here (assuming the nymble format has the ZK-pseudonym property), U computes each of her nymbles according to the established protocols and issues ZKPs to prove that she has done so.

B. Performance measurements

We implemented key components of the distributed (t, s)-threshold Verinym Issuer construction in order to obtain performance measurements. Our implementations were written in C++ using NTL to handle multiprecision arithmetic. The performance measurements were obtained on a 2.83 GHz Intel Core 2 Quad Q9550 running Ubuntu 9.10 64-bit. All code is single-threaded and all experiments used a single core.

Table I summarizes performance measurements for U and the VIs in the Verinym Acquisition Protocol. The protocol was executed on a single machine using a $(3, 7)$-threshold construction with $\eta = 17$ and each trial used a new pseudorandomly generated 12-bit e with hamming weight 3. Thus, the performance measurements contained herein represent the combined computational expense for all three VIs (indeed, these could easily be run in parallel) and do not account for the expected latency due to communication between U and the VIs (no inter-VI communication is required). Note, however, that such communication costs will be low; in particular, each VI sends just four values to U (the verinym share X_{i_j}, the verification values c_{i_j} and g_{i_j}, and the expiration VVP K_{i_j}). Both X_{i_j} and g_{i_j} are about the same size as n, while c_{i_j} and K_{i_j} are (much) smaller. Our implementation uses a value of $\kappa_1 = 30$ for the security parameter (as recommended by Fouque and Stern [14]). For each bit length of n, we repeated the experiment 100 times and report here the mean execution time ($\pm$ the standard deviation) in milliseconds.

Table II summarizes performance measurements for U and the NI in the Verinym Showing Protocol. For each of the 100 experiments of the Verinym Acquisition Protocol, we used the resulting verinym to perform a trial of this experiment. Thus, each measurement also used $\eta = 17$, a pseudorandomly generated 12-bit e value with hamming

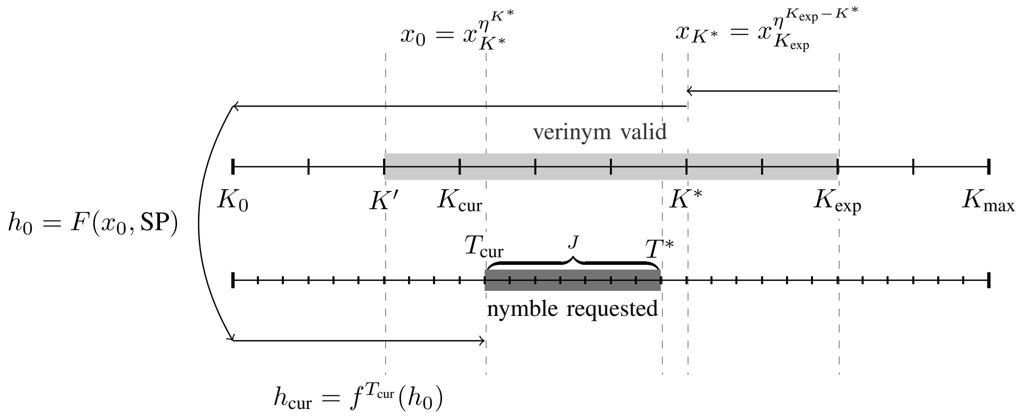
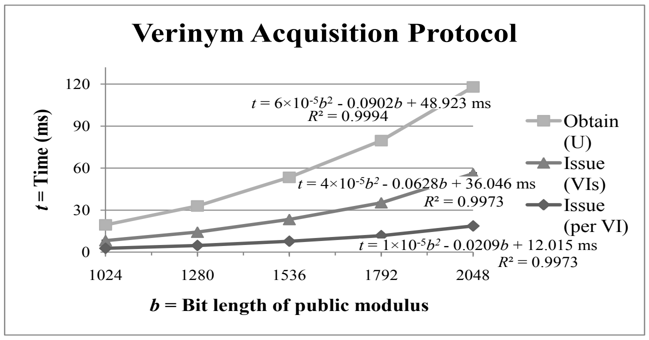

$x_0 = x_{K^*}^{\eta^{K^*}}$
$x_{K^*} = x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}}-K^*}}$
verinym valid
K_0
K'
K_{cur}
K^*
K_{exp}
$K_{\max}$
$h_0 = F(x_0, \text{SP})$
T_{cur}
J
T^*
nymble requested
$h_{\text{cur}} = f^{T_{\text{cur}}}(h_0)$

Figure 3. **Verinym showing procedure:** This figure outlines our threshold verinym construction. The upper timeline displays VVPs while the lower displays time periods. The **light grey** bar is the range of VVPs for which the verinym is valid (K' through K_{exp}). The **dark grey** bar is the range of time periods for which U is requesting nymbles (T_{cur} through T^*). U proceeds as follows: she first computes x_{K^*} by raising $x_{K_{\text{exp}}}$ to the power $\eta^{K_{\text{exp}}-K^*}$. Next, U computes x_0 from x_{K^*} (together with a proof of correct computation) by raising x_{K^*} to the power η^{K^*}; this convinces the NI that U's verinym is valid for the required VVP. The seed value for nymble construction is then computed via $F(x_0, \text{SP})$, and the one-way function f is applied iteratively (T_{cur} times) to obtain the seed for time period T_{cur}. (The functions F and f are part of the underlying scheme.)

Table I
PERFORMANCE MEASUREMENTS FOR U AND THE VI IN THE VERINYM ACQUISITION PROTOCOL.

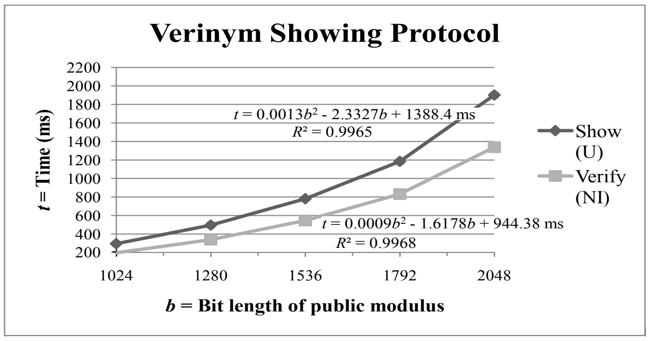
Operation	Host	Bit length of modulus	Mean execution time ± standard deviation (ms)
Issue verinym	VI	1024	8.3 ms ± 0.34 ms
		1280	14.3 ms ± 0.35 ms
		1536	23.3 ms ± 0.43 ms
		1792	35.3 ms ± 0.53 ms
		2048	56.1 ms ± 0.60 ms
Obtain verinym	U	1024	19.5 ms ± 0.42 ms
		1280	33.0 ms ± 0.64 ms
		1536	53.4 ms ± 0.56 ms
		1792	79.7 ms ± 0.89 ms
		2048	118.0 ms ± 1.58 ms



Computed using pseudorandomly chosen 12-bit e values with hamming weight 3, $\eta = 17$ and 12 VVPs in a $(3, 7)$-threshold scheme. Issuing times represent combined computational expense for all three VIs. Each experiment was repeated 100 times; the mean execution time ($\pm$ the standard deviation) across all trials is reported here. Error bars are displayed but are so small as to be only barely visible.

Table II
PERFORMANCE MEASUREMENTS FOR U AND THE NI IN THE VERINYM SHOWING PROTOCOL.

Operation	Host	Bit length of modulus	Mean execution time ± standard deviation (ms)
Show verinym	U	1024	295.4 ms ± 25.91 ms
		1280	495.7 ms ± 27.80 ms
		1536	780.7 ms ± 24.28 ms
		1792	1184.2 ms ± 28.68 ms
		2048	1901.2 ms ± 28.97 ms
Validate verinym	NI	1024	196.6 ms ± 11.48 ms
		1280	338.2 ms ± 12.09 ms
		1536	546.4 ms ± 12.27 ms
		1792	831.5 ms ± 19.71 ms
		2048	1338.2 ms ± 15.33 ms



Computed using pseudorandomly chosen 12-bit e values with hamming weight 3, $\eta = 17$ and 12 VVPs. Each trial shows the verinym with a random VVP between 1 and 12 (inclusive) as the expiration time. Each experiment was repeated 100 times; the mean execution time ($\pm$ the standard deviation) across all trials is reported here. Error bars are displayed but are so small as to be only barely visible.

weight 3, and at most 12 VVPs. For each trial, the client chose a random VVP j between 1 and 12 (inclusive) to use in the showing protocol.

Distributed key generation in our implementation has been simulated using the dealer-based version of the protocol from [9, §3]; we therefore omit timing measurements for this portion of the protocol. We reiterate that distributed key generation only needs to occur once (or, at the very worst, infrequently), during the initial setup of the protocol. Thus, performance measurements associated with this portion of the protocol are not critical to the system's overall performance.

C. *Improving efficiency*

The exponentiation proofs (i.e., the fourth and fifth lines of $\Pi_{x_{K^*}}$, in which U proves that she knows x_{K^*} such that $(x_{K^*}^{\eta^{K^*}})^e = Y$) dominate the cost of the Verinym Showing Protocol. Our implementation uses the naive square-and-multiply algorithm for this proof. It outputs commitments to, and a ZKP of correct multiplication for, each intermediate result in the computation. Of course, a more sophisticated algorithm might be able to reduce the number of steps in the exponentiation. Alternatively, because a small number of exponents are reused a large number of times, the VI or the NI could compute (and publish) short addition chains for each exponent. Batch ZKPs of correct multiplication would likely further reduce the cost. (This works just like the batch proof of knowledge of discrete logarithms with common exponent from [1], [27].)

The strategy that our implementation uses is to fix $\eta = 17$ and to choose each public exponent e with a short bit length and low hamming weight.[5] We then compute $x_j^{e\cdot\eta^j}$ in $j + 1$ stages: U computes $x_{i-1} = x_i^{\eta}$ for $0 < i \leq j$, then raises x_0 to the e^{th} power. This reduces the number of multiplication steps in square-and-multiply to just $j + 2$. (e is computed with two multiplies, and each power of η uses one additional multiply, as the high-order bit does not require a multiply.) The number of squaring steps is $j \cdot \lfloor \lg \eta \rfloor + \lfloor \lg e \rfloor$. Our measurements indicate that computing $x_j^{e\cdot\eta^j}$ in $j + 1$ stages reduces the cost of the Verinym Showing Protocol by a factor of about two over a naive

[5]Of course, e must still satisfy $\gcd(e, \varphi(n)) = \gcd(e, s!) = 1$ and so cannot be chosen to be arbitrarily small.

computation. We also sacrifice the unconditional hiding of Pedersen commitments to further halve the cost of the exponentiation proof for U (and reduce it by about one third for the VI). The exponentiation algorithm therefore uses discrete logarithm commitments [13] instead of Pedersen commitments, to reduce the cost. To maintain unlinkability, U chooses a random group element b modulo N, computes $B = b^4 \bmod N$, and sends (b, B^{x_j}) along with proof that x_j is the same value previously committed to. The remainder of the algorithm then runs as usual.

Where Pedersen commitments are used, the NI's cost may be reduced by having $\delta = \log_{\alpha_n}(\beta_n) \bmod N$ known to the NI but kept secret from U. Then, multi-exponentiations of the form $\alpha_n^x \beta_n^\gamma$ can be reduced to single exponentiations of the form $\alpha_n^{x+\delta\gamma}$ by the NI.

III. Long-term revocation

Blacklisting U is intended to be preventative, not retributive; thus, the duration of the block should somehow reflect the probability that, and cost to the SP if, the offensive behaviour is repeated [36]. SPs typically forgive disruptive behaviour after a brief time (say, around 24 hours); this is usually sufficient to put an end to edit wars on Wikipedia or flame wars on IRC, for example. Less frequently, U's misbehaviour warrants a long-term revocation; we call revocation that spans two or more linkability windows **inter-window revocation**. Our method of providing this feature does not increase the computational cost for the SP or adversely affect user privacy.

Supporting inter-window revocations requires each SP to maintain a blacklist for each prior linkability window from which some user is still blocked. The VIs then issue U with the verinym for the past few linkability windows.[6,7] When U wishes to obtain a set of nymbles for an SP, she uses the appropriate verinym to prove that her SP-specific pseudonym from a past linkability window is not on the associated blacklist. This is reminiscent of the BLAC [32]–[34] and EPID [6] approach to blacklisting, which, as discussed elsewhere [19], raises performance and scalability concerns. However, five important distinctions with our approach warrant mention:

1) Since most IP address bans are short term, most revoked users will not appear on a long-term blacklist. This significantly reduces the expected size of the blacklist against which a user must generate a proof.
2) U only forms a proof once, during the Nymble Acquisition Protocol; the SP need not verify any expensive ZKPs and the proof will not affect the observed interaction latency between U and the SP.
3) The values that appear on the blacklist need not contain the trapdoor information that is stored in a nymble. All that is required is collision resistance among users. Thus, we reduce the blacklist entries modulo a sufficiently large prime to reduce computational costs in the protocols.[8]
4) An idea due to Brands *et al.* [4], [5] enables the user to perform this non-membership proof using a number of exponentiations proportional to $\sqrt{M}$, where M is the size of the blacklist against which the user generates a proof. For large blacklists, this method dramatically outperforms the linear exponentiations approach used in both BLAC [32]–[34] and EPID [6].
5) Finally, we can use blinded *verification tokens* to let U prove *that she already proved* that her SP-specific pseudonym is not on a blacklist, thus eliminating much redundant computation. The largely static nature of long-term blacklists makes this possible.

Considered together, these five observations make our approach highly practical.

A. Construction

We illustrate how the **Non-membership Proof Protocol** works for a generic nymble format that satisfies the ZK-pseudonym property; modifying this procedure to work with any concrete construction satisfying the ZK-pseudonym property with verinyms that are computed deterministically is straightforward (only the proof that the nymble is correctly formed would change), although the exact procedure depends on the nymble format used by the scheme.

Let ℓ_ρ be a parameter specifying the desired bit-length of entries on the blacklist and let ℓ_P be a bit length for which computing discrete logarithms modulo an ℓ_P-bit prime is infeasible. Choose ρ and q_ρ prime such that:

1) $\lceil \log_2(\rho) \rceil = \ell_\rho$,
2) $\lceil \log_2(q_\rho) \rceil = \ell_P - \ell_\rho - 1$, and
3) $P = 2\rho q_\rho + 1$ is prime.

All entries on a blacklist are reduced modulo ρ and ZKPs are in the order-ρ subgroup modulo P. We suggest $\ell_\rho = 256$ (indeed, $\rho = 2^{256} - 183$ is a good choice) and $\ell_P = 1536$.

Let $\mathcal{B}_{(\mathrm{SP},L^*)} = \{\nu_1 \bmod \rho, \ldots, \nu_M \bmod \rho\}$ be the list of nymbles (reduced modulo ρ) that still appear on SP's blacklist from linkability window L^*. For ease of presentation, we will assume that $|\mathcal{B}_{(\mathrm{SP},L^*)}| = M$ is a perfect square and let $m = \sqrt{M}$. Also, let α_ρ, β_ρ be generators of the order-ρ subgroup modulo P; α_ρ and β_ρ are publicly known and $\delta = \log_{\alpha_\rho}(\beta_\rho) \bmod P$ is known only to the NI.

Thus, to prove to the NI that she is not subject to a ban from linkability window L^*, U proves that her pseudonym from linkability window L^* (reduced modulo ρ), which we denote by ν_{L^*}, does not appear on $\mathcal{B}_{(\mathrm{SP},L^*)}$. We use

[6]The number of linkability windows is a system parameter. It should be carefully chosen to provide a good balance between privacy (if U learns a verinym, she can recognize blacklist entries corresponding to past owners of her unique resource) and functionality (the number of linkability windows determines the maximum duration of an inter-window revocation).

[7]This is not possible in schemes like Jack [22] and BNymble [23] that base verinyms on user-chosen randomness; this seems to be an inherent limitation that comes with the unconditional unlinkability of such verinyms. Of course, these schemes could be adapted to use the approach of this section by sacrificing unconditional unlinkability in exchange for computational unlinkability with an honest-majority assumption using, e.g., our (t, n)-threshold verinym construction.

[8]This observation may also help to reduce the cost of downloading the blacklist in other schemes.

the following technique, due to Brands *et al.* [4], [5], to implement this proof.

For $1 \leq i \leq m$, let $j = (i-1) \cdot m$ and define the polynomial

$$\begin{aligned} p_i(\tau) &= (\tau - \nu_{j+1})(\tau - \nu_{j+2}) \cdots (\tau - \nu_{j+m}) \\ &= a_{i,m}\tau^m + \cdots + a_{i,1}\tau + a_{i,0} \bmod \rho \end{aligned}$$

U does the following:

1. U computes a Pedersen commitment $\overline{x_0} = \alpha_\rho^{x_0}\beta_\rho^{\gamma_1}$ to her verinym x_0, and uses the Verinym Showing Protocol to prove that it is valid. She also computes her SP-specific pseudonym ν_{L^*} for linkability window L^*.
2. U chooses $r_1, \ldots, r_m \in_R \mathbb{Z}_\rho$ and, for $1 \leq i \leq m$, computes
   a) $v_i = p_i(\nu_{L^*}) \bmod \rho$, the evaluation of p_i at ν_{L^*};
   b) $w_i = a_{i,m}r_m + \ldots + a_{i,1}r_1 \bmod \rho$;
   c) $C_i = \alpha_\rho^{(\nu_{L^*})^i}\beta_\rho^{r_i} \bmod P$, and
   d) $C_{v_i} = \alpha_\rho^{v_i}\beta_\rho^{w_i} \bmod P$.
3. U transmits each commitment C_i and C_{v_i} to the NI. She also transmits Π_1, which is a zero-knowledge proof that C_1 commits to the (correctly formed) SP-specific pseudonym associated with x_0 for L^*, and

$$\Pi_2 = PK\left\{\begin{pmatrix}\nu_L, \\ r_i, \\ w_i, \\ v_i\end{pmatrix} : \begin{array}{l} C_i = \alpha_\rho^{\nu_L^i}\beta_\rho^{r_i} \bmod P, \\ \wedge\, C_{v_i} = \alpha_\rho^{v_i}\beta_\rho^{w_i} \bmod P, \\ \wedge\, v_i \not\equiv 0 \bmod \rho, \\ \text{for all } 1 \leq i \leq m \end{array}\right\}, \tag{1}$$

which proves that: 1) the commitments C_i hide consecutive powers of ν_L, and 2) the commitments C_{v_i} each hide nonzero values. Note that, combined with Equation 2, this proves to the NI that C_{v_i} is a commitment to $p_i(\nu_{L^*})$, and that this evaluation is nonzero.

The NI does the following:

4. The NI verifies that, for each $1 \leq i \leq m$,

$$C_{v_i} \overset{?}{\equiv} (C_m)^{a_{i,m}}(C_{m-1})^{a_{i,m-1}} \cdots (C_1)^{a_{i,1}}\alpha_\rho^{a_{i,0}} \bmod P \tag{2}$$

If any of these equivalences fails, then the NI aborts.
5. The NI verifies the ZKPs Π_1 and Π_2. If either verification fails, the VI aborts.

Note Step 5 seems to require $m(m+1) = M + m$ modular exponentiations; however, collapsing all m of these verifications into a single batch verification using techniques of Bellare *et al.* [1] reduces this to just $2m+1$ modular exponentiations. To do this, the NI chooses random $s_1, \ldots, s_m \in_R \{1, \ldots, \kappa_2\}$, where κ_2 is a security parameter, and checks if

$$\prod_{i=1}^{m} C_{v_i}^{s_i} \overset{?}{\equiv} \alpha_\rho^{\sum_{i=1}^m a_{i,0}\cdot s_i} \cdot \prod_{i=1}^{m} C_i^{\sum_{j=1}^m a_{j,i}\cdot s_j} \bmod P. \tag{3}$$

If the verification fails, at least one C_{v_i} is incorrect, and the NI aborts; otherwise, all of the C_{v_i} are correct with probability at least $1 - 1/\kappa_2$ and the NI accepts the proof.

B. Correctness

Let us briefly examine why this proof convinces the NI that ν_{L^*} is not on $\mathcal{B}_{(\mathrm{SP},L^*)}$. First observe that, by way of construction, for $1 \leq i \leq m$, the zeros of $p_i(\tau)$ are exactly those values appearing on the sublist of $\mathcal{B}_{(\mathrm{SP},L^*)}$ defined by $\{\nu_{j+1}, \nu_{j+2}, \cdots, \nu_{j+m}\}$, where $j = (i-1)\cdot m$, and that these sublists cover the entire blacklist. Combined with Π_1, the first line of Π_2 proves that C_i hides the ith power of ν_{L^*}; thus, if Equation 2 holds then it follows that C_{v_i} is a commitment to an evaluation of $p_i(\tau)$ at the point ν_{L^*}, since

$$\begin{aligned} C_{v_i} &\equiv (C_m)^{a_{i,m}}(C_{m-1})^{a_{i,m-1}} \cdots (C_1)^{a_{i,1}}\alpha_\rho^{a_{i,0}} \\ &\equiv \alpha_\rho^{(a_{i,m}\nu_{L^*}^m + \cdots + a_{i,1}\nu_{L^*} + a_{i,0})}\beta_\rho^{(a_{i,m}r_m + \cdots + a_{i,1}r_1)} \\ &\equiv \alpha_\rho^{p_i(\nu_{L^*})}\beta_\rho^{w_i} \bmod P. \end{aligned}$$

The remainder of the proof convinces the verifier that no C_{v_i} hides the value zero, from which it concludes that ν_{L^*} is not a root of any $p_i(\tau)$ and, consequently, that $\nu_{L^*} \notin \mathcal{B}_{(\mathrm{SP},L^*)}$.

C. Computational complexity

We now analyze the computational and communication complexity of the Non-membership Proof Protocol. Let μ_ρ (resp. μ_P) be the cost of multiplication modulo ρ (resp. modulo P), let ι_ρ (resp. ι_P) be the cost of a modular inversion modulo ρ (resp. P), and let χ_ρ (resp. χ_{κ_2}) be the cost of exponentiation modulo P with ℓ_ρ-bit (resp. $\lceil\log_2(\kappa_2)\rceil$-bit) exponent.

Upon updating the blacklist, the PE computes each $p_i(\tau) \bmod \rho$. Since each of these degree-m polynomials is monic, there are only m coefficients modulo ρ to send for each polynomial. Consequently, transmitting these polynomials instead of the blacklist requires zero additional bandwidth. Thus, we do not consider the computational cost of this step in our analysis.

Evaluating each polynomial at the point ν_{L^*} requires $m \cdot \mu_\rho$ work using Horner's method; thus, the cost for this step is $m \cdot m \cdot \mu_\rho = M \cdot \mu_\rho$. Similarly, computing w_i requires $m \cdot \mu_\rho$ work; thus, the cost for this step is also $m \cdot m \cdot \mu_\rho = M \cdot \mu_\rho$. So far, this is $2M \cdot \mu_\rho$ work for U.

Computing the commitment C_i requires two exponentiations with ℓ_ρ-bit exponents, plus one multiplication, all modulo P. The cost of computing all m such commitments is then $2m(\chi_\rho + \mu_P)$. The same analysis applies to computing each C_{v_i}, thus yielding a total of $4m(\chi_\rho + \mu_P) + 2M \cdot \mu_\rho$ work for U.

The bandwidth cost for U to upload each commitment C_i and C_{v_i} to the NI is just $2m \cdot \ell_P$ bits. Together with the cost of U downloading the blacklist from the NI, this yields a total communications complexity of $M \cdot \ell_\rho$ bits download and $2m \cdot \ell_P$ bits upload for U (and vice-versa for the NI).

The left-hand side of the batch proof, Equation 3, requires m exponentiations with small exponent and $m-1$ multiplications modulo P; this is $m \cdot \chi_{\kappa_2} + (m-1)\cdot\mu_P$ work. The right-hand side requires $m(m+1)$ multiplications modulo ρ to compute the exponents, followed by $m+1$ exponentiations with exponents modulo ρ and m multiplications modulo P; this is $m \cdot \chi_\rho + M \cdot \mu_\rho + m \cdot \mu_P$ work.

The costs of the Verinym Showing Protocol and of the proof Π_1 that the SP-specific pseudonym is computed correctly are dependent on the details of the underlying Nymble-like scheme, but are in any event independent of the blacklist size M. Therefore, we omit them from our detailed analysis of the dependence of the protocol's cost on M. All that is left to be considered, then, is the cost of the large zero-knowledge proof, Π_2, in Equation 1.

The first line of Π_2 requires U to compute $4(m-1)$ exponentiations with ℓ_ρ-bit exponents, $2m$ multiplications modulo P and $3m$ multiplications modulo ρ for a total cost of $4(m-1)\cdot\chi_\rho + 2m\cdot\mu_P + 3m\cdot\mu_\rho$. U is also required to upload $2\cdot(m-1)\cdot\ell_P + 3(m-1)\cdot\ell_\rho$ bits. Here, U is proving that each C_i, $i>1$, is a commitment to the product of the values committed to in C_{i-1} and C_1. Verifying this requires $2m\cdot\chi_\rho$ work by the NI, using knowledge of $\delta=\log_{\alpha_\rho}(\beta_\rho)$ to reduce the number of exponentiations.

The second and third lines of Π_2 require U to compute m multiplicative inverses modulo P, m multiplications modulo P, m multiplicative inverses modulo ρ, $2m$ multiplications modulo ρ and two exponentiations with ℓ_ρ-bit exponents. Thus the cost for U is $m(\iota_P+\mu_P+\iota_\rho+2\mu_\rho)+2\chi_\rho$. Similarly, the NI can verify these proofs with $m\chi_\rho + m\iota_P$ work. This is done using Brands' NOT proof [3, §3]: to prove

$$PK\big\{(v_i,w_i) : C_{v_i} = \alpha_\rho^{v_i}\beta_\rho^{w_i} \bmod P \wedge (v_i \neq 0)\big\},$$

U simply performs a proof of knowledge of a discrete log representation of α_ρ with respect to C_{v_i} and β_ρ. That is, U proves that she knows γ and ζ such that $\alpha_\rho = C_{v_i}^{\gamma}\beta_\rho^{\zeta} \bmod P$; in this case, $\gamma = v_i^{-1} \bmod \rho$ and $\zeta = -w_i\gamma \bmod \rho$. This convinces the NI that U knows v_i and w_i since they are easily computable from γ and ζ, and that v_i is nonzero (since otherwise v_i^{-1} would be undefined). U transmits $2m$ group elements modulo ρ for a total of $2m\cdot\ell_\rho$ bits communication.

Thus, the overall computational cost of this protocol for U is $2(M+5m)=O(M)$ multiplications modulo ρ, $7m=O(m)$ multiplications modulo P, $m=O(m)$ multiplicative inverses modulo ρ, $m=O(m)$ multiplicative inverses modulo P, and $8m-2=O(m)$ exponentiations modulo P with ℓ_ρ-bit exponents. The cost for the NI is $M=O(M)$ multiplications modulo ρ, $2m-1=O(m)$ multiplications modulo P, $m=O(m)$ multiplicative inverses modulo P, $4m=O(m)$ exponentiations modulo P with ℓ_ρ-bit exponents, and $m=O(m)$ exponentiations modulo P with ℓ_{κ_2}-bit exponents. Communication costs are $M\cdot\ell_\rho$ bits download and $((7m-3)\cdot\ell_\rho + 2(m-1)\cdot\ell_P)$ bits upload for U, and vice versa for the NI. As noted in [20], Wikipedia currently blocks just under 7000 anonymous users per month; this gives a reasonable estimate for the upper bound on the size of a long-term blacklist for that site. With our suggested parameters of $\ell_\rho=256$ and $\ell_P=1536$, this means the blacklist will be 224 KB (assuming all 7000 SP-specific pseudonyms appear on the same long-term blacklist), and U will upload less than 50 KB to the NI to perform the non-membership proof.

As pointed out in [4], [5], much of this cost can be converted to precomputation by using Brands' error correction factors technique. We refer the reader to [4] or [3, §5.4.2] for details.

D. Performance measurements

We implemented key components of the long-term blocking construction in order to obtain performance measurements. As with our implementation of the distributed (t,n)-threshold VI, our long-term blacklisting construction is written in C++ with NTL. All benchmarks were single-threaded, and run on the same system described in §II-B.

Table III summarizes performance measurements for the Non-membership Proof Protocol. We ran both U and the NI on a single machine; thus, the performance measurements contained herein represent computational expense only and do not account for the expected latency due to communication between U and the NI. Moreover, we omit the necessary call to the Verinym Showing Protocol in our experiments. (Thus, these timings correspond precisely to the parts of the protocol that we analyzed in §III-C.) Note, however, that our implementation does not take advantage of the VI's knowledge of $\delta=\log_{\alpha_\rho}(\beta_\rho) \bmod P$ to improve the efficiency of the verification equation, nor have we implemented Brands' error correcting factors technique. Either of these modifications could improve on the performance measurements we report.

We took our performance measurements for BLAC [32] and PEREA [33],[9] found in Figures 4 and 5, directly from those schemes' respective papers. Thus, the comparison between our approach to long-term revocation and the authentication protocols of BLAC and PEREA is only an approximation. Moreover, the cost of our scheme will increase (by a constant additive factor) when one considers the additional cost of the Verinym Showing Protocol (which needs to be run whether or not the user computes a non-membership proof). Nonetheless, our measurements indicate that even our unoptimized implementation dramatically outperforms the approach taken by BLAC, EPID, and PEREA for even moderate-sized blacklists. Moreover, we reiterate that, unlike in those schemes, our non-membership proof only needs to be executed during the Nymble Acquisition Protocol, and therefore does not place additional load on the SP nor affect the observed interaction latency between the user and the SP. Indeed, our approach is practical even for extremely large SPs, such as Wikipedia and Slashdot, which are two oft-cited examples motivating the development of Nymble-like systems.

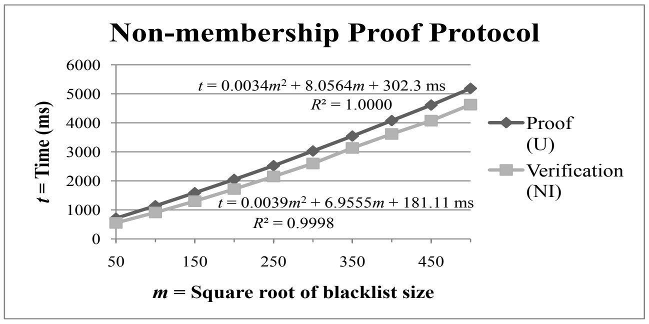
E. Improving efficiency

1) Hybrid algorithm: Observe that, for small blacklist sizes, the naive linear-time non-membership proof employed by BLAC outperforms the more sophisticated square-root-time algorithm recommended in this section. In particular, BLAC's approach scales linearly in the size of the blacklist

[9]We omit direct comparison with EPID because we do not have access to performance measurements for that scheme. However, EPID has the same asymptotic complexity as BLAC; i.e., it scales linearly in the size of the blacklist [33].

Table III
PERFORMANCE MEASUREMENTS FOR U AND THE NI IN THE BLACKLIST NON-MEMBERSHIP PROOF.

Operation	Host	m	$M = m^2$	Mean execution time $\pm$ standard deviation (ms)
Non-membership proof	U	100	10,000	1141.6 ms $\pm$ 30.57 ms
		200	40,000	2049.0 ms $\pm$ 53.22 ms
		300	90,000	3028.8 ms $\pm$ 76.94 ms
		400	160,000	4076.6 ms $\pm$ 113.95 ms
		500	250,000	5185.0 ms $\pm$ 137.46 ms
Non-membership verify	NI	100	10,000	913.3 ms $\pm$ 16.46 ms
		200	40,000	1718.9 ms $\pm$ 34.81 ms
		300	90,000	2599.1 ms $\pm$ 55.80 ms
		400	160,000	3615.5 ms $\pm$ 107.93 ms
		500	250,000	4626.2 ms $\pm$ 181.07 ms



All experiments used a 256-bit subgroup modulo a 1536-bit prime; thus, in all experiments the blacklist consists of M pseudorandom 256-bit integers. Each experiment was repeated 100 times; the mean execution time ($\pm$ the standard deviation) across all trials is reported here. We omit error bars from the graph because the error range is too small for them to be visible. Note that, while linear in m with respect to the number of exponentiations performed, the computational complexity of the non-membership proof protocol is quadratic in m (linear in M) with respect to the number of multiplications performed. Thus, by the time $m = 500$ the algorithm performs around $k{\cdot}250000$ additional multiplications, which is the same as about $^{(k\,\cdot\,250000)}/_{(3/2\,\cdot\,256)} \approx k{\cdot}147$ extra exponentiations. ($k \approx 2$ for U and $k \approx 1$ for the NI.) This is why the trend lines for these curves are quadratic in m, albeit with a very small quadratic coefficient.

with about 1.8 ms per entry at the client and 1.6 ms per entry at the server [32]; our proposed algorithm scales with the square root of the size of the blacklist but has a larger additive constant term. Our measurements indicate that for blacklists of fewer than about 250 entries BLAC's linear approach outperforms our own. For this reason, we propose a hybrid approach wherein a naive linear time non-membership proof is employed for small blacklist sizes (smaller than 250, in this case) and the above square root time non-membership proof is employed for larger blacklist sizes. Figures 4 and 5 compare the relative complexities of the different non-membership proofs for various blacklist sizes.

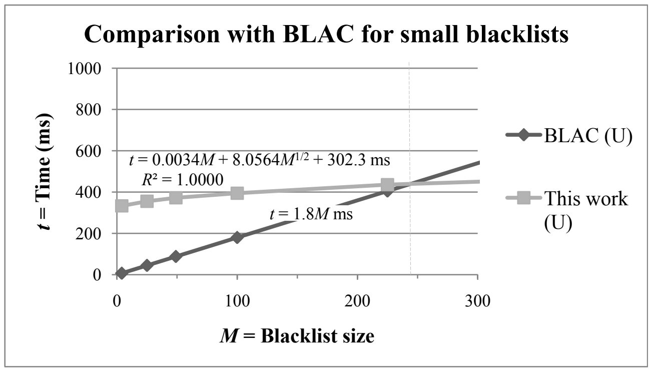


Figure 4. **Non-membership proof comparison for small blacklists**: This figure compares the performance of our proposed non-membership proof against that of BLAC [32]–[34]. We took our timing measurements for BLAC directly from the paper in which it first appeared, while we have measured the timing information for our approach empirically; hence, the coefficients on each line are only an approximation to the schemes' actual performance. All experiments used a 256-bit subgroup modulo a 1536-bit prime; thus, in all experiments the blacklist consists of M pseudorandom 256-bit integers. We omit error bars from the graph for two reasons: 1) the error range for our approach is too small for them to be visible, and 2) we computed the measurements for BLAC from data in the original BLAC paper instead of measuring it experimentally.

2) Verification tokens: Since long-term blacklists are expected to be relatively static, both U and the NI can avoid much redundant computation by—upon successful completion of the Non-membership Proof Protocol—negotiating an efficiently verifiable token that certifies that U has already proved that her SP-specific pseudonym is not on a list. In other words, once U proves that her pseudonym ν_{L^*} is not on $\mathcal{B}_{(\mathrm{SP},L^*)}$, she and the NI negotiate a blinded token certifying that the NI has verified this proof. Of course, these lists are not entirely static; indeed, the SP will add and remove entries as it detects and forgives misbehaviour. We thus associate a version number `ver` with each blacklist. When the PE *adds* an entry, the version number gets incremented; when the PE *removes* an entry, the version number is not changed. If U has a verification token for blacklist version `ver`, she engages in one of the following two procedures: if `ver` is the version of the current blacklist, she shows the token. If the blacklist version has been incremented since her last authentication, U finds the sublist of SP-specific pseudonyms added *since* version `ver`. She then shows her token and a proof for the smaller sublist.

Of course, this approach leaks some information to the VI about U. In particular, the VI learns: 1) that some user is requesting nymbles for a particular SP, 2) that this same user has previously obtained nymbles for this same SP, and 3) the approximate time (i.e., the blacklist version) that this user last requested nymbles for this SP. While we feel that this information leakage is acceptable for most users, we point out that this is an opt-in feature and U is free to make her own choice about whether this information leakage is acceptable. If U chooses not to use verification tokens, then she is indistinguishable from any user that has not recently engaged in a non-membership proof for this SP.

In what follows, $\mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})}$ will denote version `ver` of the SP's long-term blacklist for linkability window L^*. That is, $\mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})}$ contains pseudonyms of users that were blocked in linkability window L^* and whose misbehaviour has not yet been forgiven. U and the NI run the following protocol after the NI has accepted a proof from U. Such a proof may consist of U: 1) proving directly that $\nu_{L^*} \notin \mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})}$, 2) presenting a valid verification token for $\mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})}$ from a previous session, or 3) presenting a valid token for $\mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver}')}$ and then proving that $\nu_{L^*} \notin \mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})} - \mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver}')}$.

Once U has successfully convinced the NI that her nymble is not on a blacklist $\mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})}$, she obtains an anonymous credential from the NI encoding the attributes $(\mathrm{SP}, \nu_{L^*}, \mathtt{ver}')$, where $\mathtt{ver}'$ is the current blacklist version number for SP. This anonymous credential is U's verification token. The choice of anonymous credential scheme used here is unimportant, although we remark that the user shows a credential only once; thus, unlinkability between different showings of the credential is not required. For this reason, we recommend using Brands credentials [2], [3], as they outperform the other well-known schemes in the literature [4], [5].

In order to use a verification token to convince the NI that U does not appear on $\mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})}$, U reveals $\mathtt{ver}'$ and SP, then proves in zero-knowledge that the value ν_{L^*} encoded in the credential is indeed associated with her unique identifier. If $\mathtt{ver}' < \mathtt{ver}$, then U additionally proves that $\nu_{L^*} \notin \mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver})} - \mathcal{B}_{(\mathrm{SP},L^*,\mathtt{ver}')}$ using the proof technique presented in §III-A. Note that U already proves that the verification token encodes the correct nymble in the proof in §III-A (and, thus, she does not repeat it). Also note that, rather than issuing one authentication token per blacklist (i.e., one for each linkability window), a single verification token could include information pertaining to all blacklists (i.e., for each linkability window) against which U has just proven that her pseudonym is not present; this modification is straightforward and will not be discussed further.

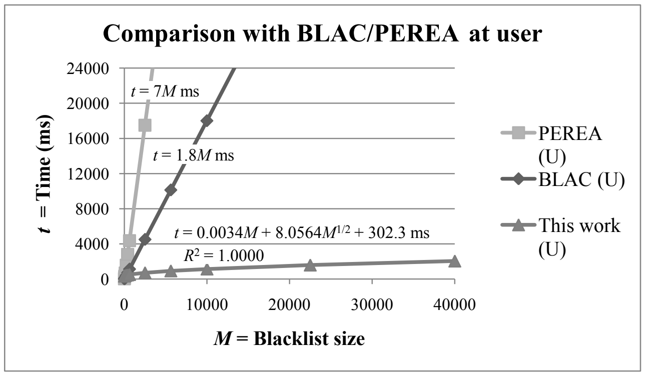
F. Blacklist sharing

It is common on the Internet for a single organization to operate several different websites. For example, the Wikimedia foundation hosts several popular websites: Wikipedia, Wiktionary, Wikiquote, etc. Using a single canonical SP name (and a common blacklist) would let Wikimedia revoke U from all of their services simultaneously; however, from a privacy standpoint, U should be able to access all Wikimedia services *concurrently* and *unlinkably*, and this would not be the case in such a setup.

Our approach to long-term revocation makes it possible for an SP to block users that engage in serious misbehaviour from multiple services while preserving the ability of honest users to access each of these services concurrently and unlinkably; we call this *blacklist transferability* [18]. More precisely, it provides blacklist transferability for inter-window revocations. In particular, SP_1 can choose some subset of entries from SP_2's long-term blacklist, and require U to prove during the Nymble Acquisition Protocol that none of these entries are hers. In fact, SPs can implement more sophisticated access structures (which we will not describe in detail) to gain extremely fine-grained control over which sites a misbehaving user is able to access; e.g., Wikimedia may revoke U from all of their services for 7 days, and just Wikipedia itself for an additional 7 days. This would have essentially zero additional impact on U's privacy and would introduce minimal overhead to the Nymble Acquisition Protocol.

IV. Increasing availability

Nymble-like systems help to increase availability of certain web services for users of anonymous communications networks. However, there are two important (and often overlooked) cases where existing approaches fail. First, the operators of Tor exit relays are unable to participate in Nymble-like systems that use IP address as a unique resource; this case is particularly important since one side effect of SPs blocking access from Tor is that they also



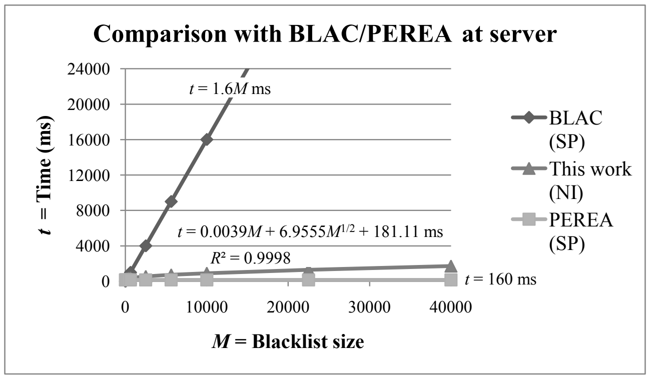


Figure 5. **Non-membership proof comparison for large blacklists**: This figure compares the performance of our proposed non-membership proof against two existing approaches in the literature: BLAC [32]–[34] and PEREA [33]. We took the timing measurements for these two schemes directly from their original papers, while we have measured the timing information for our approach empirically; hence, the coefficients on each line are only an approximation to the schemes' actual performance. Nonetheless, the graphs do capture the impact of the asymptotic behaviour of each approach. We omit error bars from these graphs for two reasons: 1) the error range for our approach is too small for them to be visible, and 2) the measurements for BLAC and PEREA are computed from data in their respective papers instead of being measured experimentally.

prevent exit node operators from using their services *even when when they do not route their connections through Tor*. Second, users located behind a firewall that censors access to Tor and the VIs are unable to obtain a verinym and are therefore unable to use the system. This section presents solutions to each of these issues.

A. Verinym acquisition for Tor exit relays

SPs cannot distinguish connections *originating at a Tor exit relay* from connections made over Tor and routed *through that exit relay*. One consequence of this is that SPs that block access from Tor also block access from Tor exit relay operators, even when their connections are not coming through Tor. Previous Nymble-like systems provided a viable solution to the availability problem for Tor's regular users, but neglected to show how their systems could support operators of Tor exit relays.

Fortunately, Tor implements a public key infrastructure (PKI) among its relays. In particular, each Tor relay has a long-term public signing key called an *identity key* [11]. Thus, the VIs can demand a ZKP of knowledge of the secret portion of an exit relay's identity key at the start of the Verinym Acquisition Protocol. In this way, the VIs prevent U from obtaining nymbles under the guise of an exit relay, while permitting the operator of that exit relay to obtain nymbles for his own use.

Suppose E is an exit node operator who wishes to connect to an SP using a Nymble-like system for anonymous authentication. We outline the additional proof required from E below. Recall that S is a set of t VIs and z is E's IP address.

E does the following:

1. E connects directly to each $\mathrm{VI}_{i_j} \in S$.

$\mathbf{VI_{i_j}}$ does the following:

2. VI_{i_j} checks z against the directory list of Tor exit relays.
3. If z not on the list, VI_{i_j} proceeds as usual for the Verinym Acquisition Protocol; otherwise, VI_{i_j} chooses a random challenge c and sends it to E.

E does the following:

4. E receives the challenge c and prepares the standard request R for a verinym.
5. E computes a signature ψ_R on $(c\|R)$ using his private identity key.
6. E transmits the tuple (R, ψ_R) to VI_{i_j}.

$\mathbf{VI_{i_j}}$ does the following:

7. VI_{i_j} receives the ψ_R and verifies the signature. If ψ_R is incorrect, VI_{i_j} aborts; otherwise, VI_{i_j} proceeds as usual for the Verinym Acquisition Protocol.

B. Verinym acquisition for censored users

Access to Tor is restricted in several countries due to government censorship (for example, the 'Great Firewall of China'). To solve this problem, the Tor network uses *bridges* [10]. A bridge is essentially just a regular Tor relay that the directory does not list. Using a variety of different techniques, censored users can obtain portions of the list of bridge relays, and thereby find an entry point into the Tor network. Obtaining the entire list, however, is intentionally a very difficult task. The goal is to make it infeasible for an adversary to block all of the bridge relays.

This solves the availability problem for Tor; i.e., censored users can still access the Tor network by using bridges. However, it seems prudent to expect that when the entire Tor network is blocked, then so too will be the VIs. This will prevent censored users from obtaining a verinym in the usual way. What we need, it appears, is a Nymble-like system analog of bridges.

In particular, we envision a set of simple volunteer-run entities, which we call **Identity Verifiers** (IVs). The IVs are simple servers (perhaps an Apache module running on volunteer machines) distributed throughout the Internet. Each IV possesses a public-private key pair for signature generation. The list of IP addresses and public keys for all available IVs is known to the VIs. Ideally, no single VI will possess the entire list, lest that VI be compromised; instead, each VI may have approximately $(1/s)^{\text{th}}$ of the list.

It should be difficult for an attacker to gain access to large portions of the IV list. In fact, bridge relays could double as IVs, making the problem of obtaining the lists of bridges and IVs equivalent. Alternatively, the list of bridge relays and IVs could be linked in such a way as to make the task of obtaining large portions of each list equivalent. However, we leave further development of these considerations to future work.

The IVs offer the following functionality: upon receiving a challenge bit string c from a user U with IP address z, an IV responds with a signature on $hash(c\|z)$. The additional part of the protocol works as follows:

U does the following:

1. U connects to an arbitrary bridge relay B and builds a circuit and SSL connection to VI_{i_j} through B; U sends her claimed IP address z to VI_{i_j} through this connection.

$\mathbf{VI_{i_j}}$ does the following:

2. VI_{i_j} receives z from U and replies with a random challenge c and the IP address of an IV selected by VI_{i_j}. (The method of selection can be arbitrary: random, the IV most trusted by the VI, etc.)

U does the following:

3. U receives the challenge c and IP address of an IV from VI_{i_j}; she connects to the IV and sends c.

The IV does the following:

4. The IV receives c and determines z empirically from the IP connection header. It replies by sending ψ_z, which is a signature on $hash(c\|z)$.

U does the following:

5. U receives ψ_z from the IV and forwards it to VI_{i_j}.

$\mathbf{VI_{i_j}}$ does the following:

6. VI_{i_j} receives ψ_z and checks the signature. If ψ_z is incorrect, VI_{i_j} aborts; otherwise, VI_{i_j} proceeds as usual for the Verinym Acquisition Protocol.

The naive protocol just described is susceptible to the following attack: a malicious U chooses a random IP address and initiates the protocol with that as its self-reported address. In the (unlikely) event that U receives the address of a colluding IV, she obtains a signature on the fake IP address, thereby convincing the VI to issue a share of a verinym. Otherwise, U chooses a new random IP address and tries again. To protect against this attack, we can require that: a) the VIs somehow trust the IVs, b) the VIs choose multiple IVs and require U to obtain a signature from each, or c) a combination of these approaches.

V. Objective blacklisting

Schwartz *et al.* proposed *contract-based revocation* [18] in their Contractual Anonymity papers [28]–[30], whereby U enters into an *anonymity contract* with the SP. This contract assures U of her anonymity as long as she does not violate the contract; if, on the other hand, she violates the contract, then a Group Manager (GM) can revoke her anonymity. Schwartz *et al.* use ideas from trusted computing to construct a contract-based revocation system based on group signatures. In their scheme, U uses remote attestation to verify that the software running on the GM will only deanonymize her if she does indeed violate the contract.

In [22], Lin and Hopper describe an objective blacklisting extension for Jack. Their approach uses the *label* field in Camenisch and Shoup's verifiable encryption scheme [7] to force the PE to include a contract in its trapdoor computation. The idea here is that if the provided contract is incorrect, then the trapdoor computation will fail. It is reasonable to argue that any added security offered by this approach is illusional (see the discussion regarding additional trust assumptions below); nonetheless, one can easily incorporate a similar mechanism into the nymble constructions of other Nymble-like systems as outlined below. Because different Nymble-like systems do not necessarily share a common trapdoor function, we propose to use a hash c of the contract as an input parameter to the one-way function used to compute the subsequent nymbles in a sequence. When incorporated into Nymble and Nymbler, this idea works as follows:

- **Nymble:** use c as the HMAC key for the 'top' chain (see Figure 1).
- **Nymbler:** replace Rabin's function $f(z) = z^2 \bmod n$ with $f'(z, c) = c \cdot z^2 \bmod n$.

In order to have U blacklisted, the SP transmits her nymble, a copy of the contract, and proof of her misbehaviour to the PE. The PE verifies that the behaviour does indeed violate the contract before computing the remainder of U's nymbles (the computation of which requires c). If the SP then provides U's nymble to the PE with an incorrect contract, then any nymbles output by the PE will not be linkable back to U. Note that, unlike in [22], this approach does not leak information to the PE or SP about whether the given contract is enforced on U. This means, for example, that with our approach different users may have different rights in their contracts, without partitioning the anonymity set.

As in [22] (though the authors of [22] never explicitly stated it), this solution requires the following additional trust assumptions:

- U must trust the PE to verify that she did indeed violate the contract.
- The PE must trust the SP to not forge proofs of contract violations.[10]
- The SP must trust the PE not to divulge any potentially sensitive information it witnesses while verifying that misbehaviour has occurred.

To construct an objective blacklisting solution that does not require additional trust assumptions or reliance on trusted computing remains an interesting open problem.

VI. Conclusion

We have presented several extensions to the Nymble framework. In particular, we proposed a new threshold Verinym Issuer construction, an efficient way to achieve inter-window revocation and blacklist transferability, alternative verinym acquisition techniques for Tor exit relays and censored users, and contract-based revocation. These extensions improve the liveness, security and functionality of Nymble-like schemes built from the extended framework, and solve a number of open problems identified in the future work sections of papers on particular Nymble-like systems [19]–[22]. Nonetheless, there are still several open problems identified in those papers that constitute exciting directions for future research. For example, system-wide banning of cheaters [20], NAT-aware IP blocking [20], and banning of entire subnets without reducing privacy [20], [21]; please see the respective papers for more details on these problems. Moreover, another worthwhile direction for future work is to investigate the use of other unique identifiers to use in place of IP addresses; such work would likely have broader implications with respect to protecting against Sybil attacks [12]. Finally, as stated in §V, the design of an objective blacklisting mechanisms that does not require trusted hardware remains an open problem.

Acknowledgements: We thank Aniket Kate and the anonymous reviewers for their helpful comments. This work was supported by NSERC, MITACS, and a David R. Cheriton Graduate Scholarship.

References

[1] M. Bellare, J. A. Garay, and T. Rabin, "Fast Batch Verification for Modular Exponentiation and Digital Signatures,"

[10]One possible way to eliminate this additional trust is to have U sign her actions using her pseudonym as a key. Then, upon extracting U's pseudonym from her nymble, the PE would be able to verify the signature before adding U to the blacklist. Forged evidence of a violation would result in failed signature verification. However, this solution would require U to prove in zero-knowledge (i.e., without revealing the key) that the signature was computed correctly each time an action is performed, thus violating the user- and verifier-efficiency requirements of the scheme.

in *Proceedings of EUROCRYPT 1998*, Espoo, Finland, May 1998.

[2] S. Brands, “Restrictive Blinding of Secret-Key Certificates,” in *Proceedings of EUROCRYPT 1995*, Saint-Malo, France, May 1995.

[3] S. A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000.

[4] S. A. Brands, L. Demuynck, and B. D. Decker, “A Practical System for Globally Revoking the Unlinkable Pseudonyms of Unknown Users.” Department of Computer Science, K.U.Leuven, Technical Report CW472, 2006.

[5] S. A. Brands, L. Demuynck, and B. D. Decker, “A Practical System for Globally Revoking the Unlinkable Pseudonyms of Unknown Users,” in *Proceedings of ACISP 2007*, Townsville, Australia, July 2007.

[6] E. Brickell and J. Li, “Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities,” in *Proceedings of WPES 2007*, Alexandria, VA, October 2007.

[7] J. Camenisch and V. Shoup, “Practical Verifiable Encryption and Decryption of Discrete Logarithms,” in *Proceedings of CRYPTO 2003*, Santa Barbara, CA, August 2003.

[8] J. Camenisch and M. Stadler, “Efficient Group Signature Schemes for Large Groups (Extended Abstract),” in *Proceedings of CRYPTO 1997*, Santa Barbara, CA, August 1997.

[9] I. Damgård and M. Koprowski, “Practical Threshold RSA Signatures without a Trusted Dealer,” in *Proceedings of EUROCRYPT 2001*, Innsbruck, Austria, May 2001.

[10] R. Dingledine and N. Mathewson, “Design of a Blocking-Resistant Anonymity System.” The Tor Project, Technical Report, 2006.

[11] R. Dingledine, N. Mathewson, and P. F. Syverson, “Tor: The Second-Generation Onion Router,” in *Proceedings of USENIX Security 2004*, San Diego, CA, August 2004.

[12] J. R. Douceur, “The Sybil Attack,” in *Proceedings of IPTPS 2002*, Cambridge, MA, March 2002.

[13] P. Feldman, “A Practical Scheme for Non-interactive Verifiable Secret Sharing,” in *Proceedings of FOCS 1987*, Los Angeles, CA, October 1987.

[14] P.-A. Fouque and J. Stern, “Fully Distributed Threshold RSA under Standard Assumptions,” in *Proceedings of ASIACRYPT 2001*, Gold Coast, Australia, December 2001.

[15] Y. Frankel, P. D. MacKenzie, and M. Yung, “Robust Efficient Distributed RSA-Key Generation,” in *Proceedings of STOC 1998*, Dallas, TX, May 1998.

[16] I. Goldberg, “A Pseudonymous Communications Infrastructure for the Internet,” Ph.D. dissertation, UC Berkeley, 2000.

[17] S. Goldwasser, S. Micali, and R. L. Rivest, “A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks,” *SIAM Journal on Computing (SICOMP)*, vol. 17, no. 2, pp. 281–308, April 1988.

[18] R. Henry and I. Goldberg, “Formalizing Anonymous Blacklisting Systems,” in *Proceedings of IEEE S&P 2011*, Oakland, CA, May 2011.

[19] R. Henry, K. Henry, and I. Goldberg, “Making a Nymbler Nymble using VERBS,” in *Proceedings of PETS 2010*, Berlin, Germany, July 2010.

[20] R. Henry, K. Henry, and I. Goldberg, “Making a Nymbler Nymble using VERBS (Extended Version).” Centre for Applied Cryptographic Research, UWaterloo, Technical Report CACR 2010-05, 2010.

[21] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith, “Nymble: Anonymous IP-Address Blocking,” in *Proceedings of PETS 2007*, Ottawa, ON, June 2007.

[22] Z. Lin and N. Hopper, “Jack: Scalable Accumulator-based Nymble System,” in *Proceedings of WPES 2010*, Chicago, IL, October 2010.

[23] P. Lofgren and N. Hopper, “BNymble (A Short Paper): More Anonymous Blacklisting at Almost No Cost,” in *Proceedings of FC 2011*, St. Lucia, February 2011.

[24] A. Lysyanskaya, “Signature Schemes and Applications to Cryptographic Protocols,” Ph.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, 2002.

[25] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996, fifth Printing (August 2001).

[26] T. P. Pedersen, “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing,” in *Proceedings of CRYPTO 1991*, Santa Barbara, CA, August 1991.

[27] K. Peng, C. Boyd, and E. Dawson, “Batch Zero-Knowledge Proof and Verification and Its Applications,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 2, May 2007, Article No. 39.

[28] E. J. Schwartz, “Contractual Anonymity,” Master’s thesis, Information Networking Institute, Carnegie Mellon, 2009.

[29] E. J. Schwartz, D. Brumley, and J. M. McCune, “Contractual Anonymity.” School of Computer Science, Carnegie Melon, Technical Report CMU-CS-09-144, 2009.

[30] E. J. Schwartz, D. Brumley, and J. M. McCune, “A Contractual Anonymity System,” in *Proceedings of NDSS 2010*, San Diego, CA, February 2010.

[31] V. Shoup, “Practical Threshold Signatures,” in *Proceedings of EUROCRYPT 2000*, Bruges, Belgium, May 2000.

[32] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, “Blacklistable Anonymous Credentials: Blocking Misbehaving Users Without TTPs,” in *Proceedings of CCS 2007*, Alexandria, VA, October 2007.

[33] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, “PEREA: Towards Practical TTP-free Revocation in Anonymous Authentication,” in *Proceedings of CCS 2008*, Alexandria, VA, October 2008.

[34] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, “BLAC: Revoking Repeatedly Misbehaving Anonymous Users without Relying on TTPs,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 4, October 2010, Article No. 39.

[35] P. P. Tsang, A. Kapadia, and S. W. Smith, “Anonymous IP-address Blocking in Tor with Trusted Computing (Short Paper: Work in Progress),” in *Proceedings of WATC 2006 (Fall)*, Tokyo, Japan, November 2006.

[36] Wikimedia Foundation, “Wikipedia:Blocking policy — Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Wikipedia:Blocking_policy