

# Making a Nymbler Nymble using VERBS\*

Ryan Henry, Kevin Henry, and Ian Goldberg

Cheriton School of Computer Science  
University of Waterloo  
Waterloo, ON, Canada N2L 3G1  
{rhenry, k2henry, iang}@cs.uwaterloo.ca

**Abstract.** We propose a new system modeled after Nymble. Like Nymble, our scheme provides a privacy-preserving analog of IP address blocking for anonymizing networks. However, unlike Nymble, the user in our scheme need not trust third parties to maintain their anonymity. We achieve this while avoiding the use of trusted hardware and without requiring an offline credential issuing authority to guarantee that users do not obtain multiple credentials.

We use zero-knowledge proofs to reduce the capabilities of colluding third parties, and introduce a new cryptographic technique that we call *verifier-efficient restricted blind signatures*, or VERBS, to maintain efficiency. Signature verification with our VERBS are 1–2 orders of magnitude faster than existing restricted blind signatures.

**Key words:** Privacy, anonymity, authentication, anonymous blacklisting, revocation, anonymous credentials, zero-knowledge proofs

## 1 Introduction

Anonymity networks provide users with a means to communicate privately over the Internet. The Tor network [13] is the largest deployed anonymity network; it aims to defend users against traffic analysis attacks by encrypting users' communications and routing them through a worldwide distributed network of volunteer-run relays [29].

The ability to communicate without fear of network surveillance makes it possible for many users to express ideas or share knowledge that they might otherwise not be willing to reveal for fear of persecution, punishment or simply embarrassment. On the other hand, some users use the veil of anonymity as a license to perform mischievous deeds such as trolling forums or cyber-vandalism. For this reason, some popular websites (for example, Wikipedia [33] and Slashdot [14]) proactively ban any user connecting from a known anonymous communications network from contributing content, thus limiting freedom of expression.<sup>1</sup>

---

\* An extended version of this paper is available [18].

<sup>1</sup> Some IRC networks also block access to anonymous users (for example, see <https://wiki.torproject.org/noreply/TheOnionRouter/BlockingIrc>).

Therefore, a real need exists for systems that allow anonymous users to contribute content online, while preserving the ability of service providers to selectively (and subjectively) ban individual users without compromising their anonymity. Not only would such a system benefit the estimated hundreds of thousands of existing Tor users, but it might also be a boon to wider acceptance of Tor. Indeed, the need for an anonymous blacklisting mechanism has been acknowledged by several key people involved with The Tor Project [11, 12]. Thus, it is reasonable to expect that the operators of Tor might be willing to provide the infrastructure necessary to realize such a system, a situation that would greatly reduce the burden on service providers and lead to greater adoption.

Several schemes (e.g., [19, 20, 30–32]) have been proposed with the goal of allowing anonymous blacklisting of Tor users. The most well-known of these is Nymble [20, 32], which is the system after which we model our own.

### 1.1 An Overview of Nymble

Suppose a user Alice wishes to connect anonymously to a *Service Provider* (SP), such as a website, while the SP will allow connections only if it can ban a misbehaving user by IP address. To facilitate this, the Nymble system introduces two TTPs, the *Pseudonym Manager* (PM) and the *Nymble Manager* (NM). Before connecting to the SP, Alice connects directly to the PM, thus proving she has control over the specified IP address. The PM then issues Alice a pseudonym called a *Nym*, which is deterministically generated from her IP in such a way that the NM is able to verify that the pseudonym was in fact issued to Alice by the PM, but learns no information about Alice’s IP. Alice then connects to the NM over an anonymous channel and presents her Nym along with the name of the SP to which she wishes to connect. Using the pair  $(Nym, SP)$ , the NM computes and issues to Alice a set of *nymbles* — one for each time period left in the current *linkability window*. Within a linkability window, each successive nymble is generated from the previous one using a one way function (a hash function) and two secrets; one secret is known only to the NM, while the other is shared by the NM and the SP. In order to connect to the SP, Alice presents the nymble which corresponds to the current time period. The shared secret allows the SP to verify the validity of Alice’s nymble but not learn her IP address, nor compute or identify any of her other nymbles. Therefore, Alice’s connections within a time period are linkable, while her accesses across different time periods are not. The SP records the nymble used during a session; if it is later found that Alice misbehaved, the SP can complain to the NM by presenting it with a copy of the recorded nymble. The NM then issues the SP a *linking token*, which is essentially a trapdoor that allows the SP to compute all of Alice’s subsequent nymbles starting from the time period in which the complaint was made (up until the end of the linkability window). The one-way nature of hash functions guarantees that the trapdoor provides no way for the SP to compute *previous* nymbles; thus, backwards anonymity is preserved, while further connections from a misbehaving user can be detected and blocked.

## 1.2 The Not-So-Nymble Side of Nymble

Nymble provides an efficient framework for banning users of an anonymizing network; however, the simplicity and efficiency come at a cost. Recall that the PM knows the pair  $(IP, Nym)$  while the NM knows the pair  $(Nym, SP)$ . Therefore, if the NM and PM collude, it is trivial for them to determine which SP the user associated with a given IP address is accessing. Further, because it is trivial for an NM to retroactively compute a user’s nymbles, a colluding NM and SP can easily break backwards anonymity and link a user’s connections. If all three parties collude (i.e., the PM, NM, and SP), they can trivially link all actions of a given user back to that user’s IP, thus completely breaking anonymity.

## 1.3 Our Contributions

We present a new Nymble-like system, unimaginatively called *Nymbler*, that minimizes the capabilities of the PM, NM, and SP when colluding. This is accomplished through the use of anonymous credentials and a new *verifier-efficient* restricted blind signature scheme that we use to permit users to construct their own nymbles. Thus, our scheme eliminates the need to trust third parties with anonymity while maintaining the essential properties of Nymble.

**Outline** The remainder of this paper is outlined as follows: Previous work related to restricted blind signature schemes and blacklisting anonymous users are presented in §2, followed by an overview of the approach taken in this work in §3. We describe in detail our approach to verifier-efficient restricted blind signatures in §4 while our Nymbler scheme and the protocols involved are described in §5. In §6 we suggest appropriate values for security parameters and analyze the performance of our system with these choices. We conclude in §7 and outline some potential areas for future work.

# 2 Related Work

## 2.1 Restricted Blind Signature Schemes

In his seminal work [9], Chaum introduced the notion of a blind signature scheme; the idea was later elaborated in [10], where the first construction (based on RSA signatures) was given. Chaum’s scheme allows a user to obtain a cryptographic signature on a message without revealing *any* information about the message to the signer. Later, Brands [5] proposed *restricted blind signatures* in which a user obtains a blind signature on a message, while the signer gets to see certain parts of the structure of the message before signing. If this structure does not conform to certain rules, the signer can refuse to provide a signature; thus, the choice of message to be signed can be restricted by the signer. However, unlike Chaum’s blind signature scheme, where verification costs just one modular exponentiation (where the exponent can be chosen to be as small as 3), verifying

Brands’ restricted blind signatures has a computational cost dominated by a multi-exponentiation where each exponent is essentially random (modulo a large prime) and depends on the message to be signed.

Camenisch and Lysyanskaya [7] presented a versatile signature scheme (CL-signatures) that allows a *re-randomizable* restricted blind signature to be issued. The well-known CL-credential [3, 7] scheme is based on CL-signatures. In their scheme, the cost of verifying a signature is effectively one exponentiation and one multi-exponentiation, with each exponent approximately equal in size to the message to be signed.

Recently, Groth and Sahai [17] presented a zero-knowledge proof system based on bilinear pairings. Belenkiy et al. [2] proposed a restricted blind signature scheme called P-signatures and noninteractive anonymous credential system based on the Groth-Sahai framework. The cost of verification in their scheme is about one elliptic curve exponentiation and three pairing operations.

Our approach uses RSA-based signatures similar to Chaum’s, combined with zero-knowledge proofs that allow the user to prove certain properties about the message before it is signed. The key advantage of our approach over other restricted blind signature schemes is its extremely low cost verification algorithm (i.e., almost as efficient as Chaum’s *non-restricted* blind signatures with exponent 3). In particular, verifying a signature in our scheme costs just four modular multiplications, which is *1–2 orders of magnitude* faster than any previously proposed restricted blind signature scheme.<sup>2</sup>

## 2.2 Systems for Anonymous Blacklisting

Unlinkable Serial Transactions [28] was one of the first systems to allow anonymous blacklisting. The scheme prevents an SP from tracking the behaviour of its users, while protecting it from abuse due to simultaneous active sessions by a single user. Users are issued blind tokens from the SP and, in normal operation, these tokens are renewed at the end of a user’s transaction. If a user is judged to have misbehaved, the SP can block future connections from that user by refusing to issue further tokens. However, the scheme provides no way for the SP to ban a user if misbehaviour is detected *after* the end of the session in which it occurred.

The Nym system [19] was a first attempt at solving the problem of allowing anonymous edits on Wikipedia; it represents one of the first attempts at bringing accountability to users of anonymity networks. Unlike later approaches, Nym only provides *pseudonymity*, and thus is not an ideal solution. Later schemes — most notably Nymble — improve upon Nym to provide full anonymity.

Blacklistable Anonymous Credentials (BLAC) [30], proposed by several of the authors of Nymble, provides an anonymous credential system that does not make use of any TTP who can revoke the anonymity of all users. Instead, the

---

<sup>2</sup> In §6.2, we present experimental results indicating that the cost of verifying a signature in our scheme is almost forty times faster than computing a *single* modular exponentiation — an operation that is less expensive than the verification of any of the restricted blind signatures discussed above.

system allows an SP to add a credential to its blacklist if the owner of that credential is judged to have misbehaved. However, BLAC suffers from two major drawbacks. The first of these is the loss of efficiency when compared to a system like Nymble; if the blacklist grows large, say one thousand users, then several hundred kilobytes of communication and several seconds of computation are required (per access) to prove that a user is not on the blacklist [32]. For large services with many users, such as Wikipedia, the performance of this approach is unacceptable. The second downside is that the credentials are not tied to an IP address. Instead, the system assumes that some offline credential issuing authority will ensure that no user obtains more than a single credential.

Privacy-Enhanced Revocation with Efficient Authentication (PEREA) [31] is another system proposed by the same authors as BLAC. It improves upon BLAC by providing similar functionality but utilizing a cryptographic accumulator to offer computational requirements at the SP that do not depend on the size of the blacklist. To make this possible the system makes use of an *authentication window*, which is similar in concept to that of a linkability window, except that it specifies the *maximum number of subsequent connections* a user may make before it becomes impossible to block them due to behaviour during a previous session, instead of *the maximum time duration* that can elapse. However, although the cost of verification at the SP is constant regardless of the size of the blacklist, it is still several orders of magnitude slower than Nymble, taking about 0.16 seconds per authentication when the authentication window is 30 [31]. Moreover, as with BLAC, the credentials used in PEREA are not tied to an IP address and are issued by an offline credential authority that ensures no user can obtain more than one credential. In the next section we touch on the technical reasons why BLAC and PEREA cannot be adapted to use IP addresses as a unique resource.

### 3 Our Approach

This section provides a high-level overview of our scheme. Further details about how this approach is realized are presented in §4 and §5.

As a first step, we replace the pseudonymous Nym with an anonymous credential; thus, the PM is replaced by a *Credential Manager* (CM). The CM learns Alice’s IP address and issues a credential stating this fact, but the CM is unable to recognize this credential at a later time. This modification prevents the CM and NM from colluding to learn which SP a particular user is accessing.

We emphasize that our use of anonymous credentials — and the role of the CM in general — is fundamentally different from in BLAC and PEREA. For example, the CM is not required to keep track of the unique resources for which a credential has been issued; instead, the CM encodes each user’s unique resource *directly in the credential* that it issues. This prevents the enrolment issues addressed in [30], wherein a user’s credential is misplaced or compromised, from causing problems in our approach. In such a case, the CM simply issues the user with a new credential encoding the same unique resource, and all of their previous bans remain in effect. It is this property that allows us to continue to

use IP addresses as the unique resource (as in Nymble). Note that in BLAC and PEREA this choice of unique resource is unrealistic, since in those schemes an SP would have no way to distinguish two credentials encoding the same IP address from ones encoding different IP addresses.

Using her credential, our scheme allows Alice to construct her own set of nymbles in such a way that the NM is convinced of their validity without ever actually seeing them. The NM then issues Alice with *verifier-efficient restricted blind signatures* (VERBS) on her nymbles so that the SP can also be convinced of their validity. Note that from a security point of view there is no reason why the NM, and not the SP, must be responsible for verifying the integrity of Alice’s nymbles; indeed, the SP could verify Alice’s proofs directly and thus eliminate the role of the NM at this stage. Our motivation for using the NM at this stage in the protocol is simply to offload work from the SP to the NM.

In the case that Alice misbehaves and the SP wishes to ban her, the SP can present a nymble to the NM, who then performs a non-trivial amount of computation — i.e., solving a discrete log — to recover sufficient information to calculate Alice’s remaining nymbles. This is accomplished through the use of a trapdoor discrete log group, where parameters are selected so that performing discrete logs is possible using the NM’s private key but even so is sufficiently expensive that wholesale deanonymization is impractical. We emphasize that although the NM can compute subsequent nymbles from a starting point, even with the ability to solve discrete logs the NM cannot go backwards. Thus, breaking backwards anonymity in our system is much more difficult than in Nymble.

## 4 Verifier-Efficient Restricted Blind Signatures

In this section we introduce *verifier-efficient restricted blind signatures* (VERBS), a restricted blind signature scheme with an efficient verification protocol. Our scheme makes use of *commitments*, which can be Feldman commitments [15] (the commitment to  $x$  is  $\mathcal{CF}(x) = s^x$  for a known group element  $s$ ) or Pedersen commitments [24] (the commitment to  $x$  is  $\mathcal{CP}(x) = s^x r^\gamma$  for known group elements  $s, r$  where  $\log_s r$  is unknown, and  $\gamma$  is random).

We use several standard zero-knowledge proofs from the literature; in particular, we use the standard proof of knowledge of a committed value (i.e., a discrete logarithm) [27], proof that a commitment opens to a product of committed values [8], and proof of knowledge of a committed value that lies in a particular range [4]. We note that no proof is necessary for addition or scalar multiplication of committed values, as those operations are easily accomplished by multiplication or exponentiation of the commitments, respectively.

We also utilize a proof of *nested commitments* (a “nest proof”); that is, given  $A, B$ , prove that you know  $x$  such that  $A$  is a commitment to a commitment to  $x$  and  $B$  is a commitment to the same  $x$ . That is (for simplicity, we only show the Feldman case; the Pedersen case is similar), that you know  $x$  and  $G$  such that  $G = g^x$ ,  $A = s^G$ , and  $B = t^x$ . (All operations are in appropriate groups, and  $g, s, t$  are generators of those groups.)

This proof works the same way as the ordinary proof of equality of discrete logarithms: the prover chooses  $v$  and outputs  $g^v$  and  $t^v$ ; the verifier (or a hash function if the Fiat-Shamir [16] method is used) chooses a challenge  $c$ ; the prover outputs  $r = v - cx$ ; the verifier accepts if  $G^c g^r = g^v$  and  $B^c t^r = t^v$ . The twist in our scenario is that  $G$  is not available to the verifier; only its commitment ( $A = s^G$ ) is. We solve this problem by having the prover output  $s^{g^v}$  instead of  $g^v$ , and having the *prover* compute  $s^{G^c}$  (the commitment to  $G^c$ ) and prove in zero-knowledge that it was done correctly (see below). Then the verifier checks that  $(s^{G^c})^{g^r} = s^{g^v}$  (along with the unchanged  $B^c t^r = t^v$ ). In the event that  $g$  and  $t$  have different orders (which will be true in general, and in our case), the above range proof is also utilized to show that  $0 \leq x < \text{ord}(t)$ .

For the proof of an exponentiation of a committed value, we use a simplified version of the algorithm from [8]. In that paper, the *exponent* was also hidden from the verifier. In our situation, the exponent  $c$  is known, which makes matters considerably easier. The prover just performs any addition-and-multiplication-based exponentiation routine, and proves that each step was done correctly.

We next describe the four algorithms that make up VERBS. The full details are presented in [18]. We will state the algorithms in their noninteractive zero-knowledge form (such as by using Fiat-Shamir [16]); the adaptation of VERBS-Blind and VERBS-Sign to interactive zero-knowledge is straightforward. (The other two algorithms do not change.)

All computations are performed modulo an RSA number,  $\rho$ , whose factorization is known only to the signer. The **VERBS-Blind** algorithm is executed by the client. The algorithm takes as input a group element  $g$ , a commitment  $\mathcal{C}(x)$  (either Feldman or Pedersen) to a secret value  $x$ , and  $x$  itself (plus  $\gamma$  in the case of a Pedersen commitment). The role of this algorithm, much like its Chaumian counterpart, is to produce the blinded message  $S = f(\nu) \cdot \alpha^3 \bmod \rho$ , where  $\nu = g^x$  and the random blinding factor  $\alpha$  are hidden from the signer, and  $f(z) = z^2 + 1$  is a one-way function. (It is one-way since the factorization of the modulus  $\rho$  is unknown to the client.) It also produces  $\Pi$ , a zero-knowledge proof that the computation of  $S$  was performed correctly.

The **VERBS-Sign** protocol is run by the NM. It takes the tuple  $(S, p, q, \xi, \Pi)$  as input.  $S \in \mathbb{Z}_\rho^*$  is a blinding of the message to be signed.  $p$  and  $q$  are the factors of  $\rho$ .  $\xi \in \mathbb{Z}_\rho^*$  is a *context element* that encodes meta-information about the signature (see §5.1).  $\Pi$  is a zero-knowledge proof that  $S$  was correctly formed. It outputs the blinded signature  $\sigma' = (\xi \cdot S)^{\frac{1}{3}} \bmod \rho$  if all proofs in  $\Pi$  are valid; otherwise, it outputs  $\perp$ . Note that  $\sigma'$  is essentially just a Chaum blind signature.

The **VERBS-Unblind** protocol is run by the client. The algorithm takes the tuple  $(\sigma', \alpha)$  as input.  $\sigma'$  is a blind signature and  $\alpha$  is the blinding factor used to blind the signature. It outputs  $\sigma = \sigma' \cdot \alpha^{-1} \bmod \rho$ , the unblinded signature.

The **VERBS-Verify** algorithm is run by the SP. It takes the tuple  $(\nu, \sigma, \xi)$  as input;  $\nu$  is the message that was signed,  $\sigma$  is the (unblinded) signature, and  $\xi$  is the context element. It outputs **true** iff  $\sigma^3 \bmod \rho \stackrel{?}{=} \xi \cdot (\nu^2 + 1) \bmod \rho$ . Note that the cost of VERBS-Verify is just four modular multiplications.

## 5 An Improved Nymble

We next present our new anonymous blacklisting scheme modeled after Nymble. Our scheme aims to meet the same goals as Nymble, while making deanonymization of a user infeasible, regardless of which subset of third parties might collude against her. Before describing the approach in any detail, we briefly describe the third parties involved and explain their roles. Note that the SP must trust third parties to properly carry out their respective responsibilities; however, unlike in the original Nymble, the user need not trust them not to collude in order for her anonymity to be maintained.

The third parties are called the *Credential Manager* (CM) and the *Nymble Manager* (NM). The CM is responsible for issuing an anonymous credential to the user which encodes two pieces of information: an obfuscated version of the user’s IP address, and an expiration time. For added security, the CM may be distributed as outlined below. At any time before this expiration, the user can present her credential to the NM to receive a set of mutually unlinkable authentication tokens called *nymbles*, which can be used to anonymously access the services offered by a *Service Provider* (SP). The NM never sees the nymbles that it issues, but it does supply the user with a verifier-efficient restricted blind signature on each of them, which allows the user to efficiently convince the SP of their legitimacy. When the user connects to the SP over an anonymous channel she must present a valid nymble. The SP records the nymble that was used during each session. In the event of user misbehaviour, it presents a nymble to the NM, who then computes all subsequent nymbles for that user (and hence prevents her from connecting to the SP for the remainder of the *linkability window*).

### 5.1 System Parameters

In this subsection we introduce the system parameters used in our protocols. In §6.1 we discuss technical considerations in the selection of these parameters and suggest some reasonable values.

The system has a publicly known modulus  $n$ , where  $n$  is the product of two *unknown* (to anyone) large safe primes, and  $N = 2n + 1$  is prime. Such a modulus can be generated using a distributed protocol as described in [1], or with one-time trust in an entity which generates it, such as used in the erstwhile RSA Factoring Challenge [26]. Under the assumption that  $n$  is hard to factor, squaring modulo  $n$  is a *one-way function*. Thus, squaring modulo  $n$  is a one-way function that admits *efficient* zero-knowledge proofs of knowledge of preimages [8]. We fix  $a, b \in QR_N$ , the set of quadratic residues modulo  $N$ , so that  $\log_a b \bmod N$  is unknown. Choosing  $(a, b) = (4, 9)$  is fine.

Since IP addresses tend to change frequently, the system-wide parameter  $\Delta_t$  specifies the maximum time period for which an issued credential is valid. That is, after a time period of  $\Delta_t$  has elapsed, the user must reauthenticate with the CM to obtain a fresh credential encoding her current IP address, herein denoted IP.



As in the original Nymble, our scheme uses the concept of *linkability windows*. This prevents a malicious NM and SP from computing a trapdoor for a user that can be used to link that user’s actions indefinitely. The duration  $W$  of each linkability window is a parameter that can vary from SP to SP based on their own policies; reasonable values for this parameter might be, for example, twenty-four hours or one week. Each linkability window is indexed by a value  $d$ , which is used in the computation of nymbles during that time period. For example,  $d$  might be equal to the current year concatenated with the current day of the year, or the current year concatenated with the current week of the year (if twenty-four hour or one-week linkability windows are used, respectively). The method used to determine  $d$  for a given date and time should be public and easily computable by any user. Each linkability window  $d$  is further subdivided into  $\Gamma$  uniform-sized *time periods*, denoted  $\tau_{d,1}, \tau_{d,2}, \dots, \tau_{d,\Gamma}$ . A reasonable duration for these time periods might be fifteen minutes (in which case  $W = \Gamma \cdot 15$  minutes). Their duration determines how often a user is able to unlinkably access the service, as exactly one unique and unlinkable nymble is issued per IP address per time period in each linkability window.

Each SP possesses a *linking list*  $\mathcal{L}$  of the future nymbles associated with users who have misbehaved; these nymbles will not be accepted. The SP also possesses a *blacklist*  $\mathcal{B}$ , which contains one *canonical* nymble for each user in the linking list (i.e., that user’s nymble for the last time period of the linkability window), and is signed by the NM and published by the SP. Before attempting to connect to the SP a user will download a copy of this blacklist and confirm that she is not presently banned. (This is important since otherwise, if a user does not realize she is presently on the blacklist, the SP could link the user’s actions without her knowledge.) When receiving a request for a connection from a Nymble user, the SP consults the linking list to determine if the user is blacklisted. The techniques of [32] can be used to ensure that the user receives an up-to-date blacklist.

In our description of the protocols we assume that the credentials are Camenisch-Lysyanskaya (CL) credentials [3, 7], although our approach could be easily adapted to other credential systems. The CM’s public key is, therefore, the tuple  $(S, Z, R_1, R_2, m)$ , where  $m = m_p m_q$  is an  $\ell_m$ -bit product of two large safe primes of equal size,  $\langle S \rangle = QR_m$  (i.e.,  $S$  is a randomly chosen generator of the group  $QR_m$ ) and  $Z, R_1, R_2 \in_R QR_m$  are randomly chosen quadratic residues modulo  $m$ . Here  $\ell_m$  is a security parameter; in [3] the authors recommend  $\ell_m = 2048$ . The CM’s private key is the tuple  $(m_p, m_q, sk)$ , i.e., the factorization of  $m$  and a secret Message Authentication Code (**MAC**) key. For a distributed CM, each CM node would have an independent key pair.

The NM has public key  $\rho = pq$ , where  $\rho$  is an  $\ell_\rho$ -bit product of  $\ell_B$ -smooth primes  $p$  and  $q$  (that is,  $p - 1$  and  $q - 1$  are products of  $\ell_B$ -bit primes), such that  $R = 4\rho + 1$  is a prime.<sup>3</sup> It is required that  $\rho > n$ , but being just barely larger is sufficient. We note that a different  $\rho$  can be used in conjunction with each SP and linkability window, but for brevity, we will use a single  $\rho$  value in our

<sup>3</sup> We use  $4\rho + 1$  because it is easy to see that  $p$  and  $q$  must be congruent to 2 mod 3, and so  $2\rho + 1$  must be divisible by 3.

descriptions. Here  $\ell_B$  is chosen so that computing discrete logarithms modulo  $\rho$  in subgroups of order  $\approx 2^{\ell_B}$  is *costly but feasible*. In other words, given knowledge of the factorization of  $p - 1$  and  $q - 1$ , computing discrete logs modulo  $p$  and  $q$  (and hence, modulo  $\rho$ ) is feasible (but costly) using a technique like the parallel rho method of van Oorschot and Wiener [23].  $g$  is a generator of  $QR_\rho$ , and  $r$  and  $s$  are generators of the order- $\rho$  subgroup of  $\mathbb{Z}_R^*$  such that  $\log_r s$  is unknown. The NM's private key is then  $(p, q)$  and the factorization of  $\phi(\rho)$  (into  $\ell_B$ -bit primes), so  $\mathbb{Z}_\rho^*$  is a trapdoor discrete logarithm group with the NM's private key as its trapdoor.

Each SP is tied to a value  $h$ , which changes once per linkability window. Here  $h \in \mathbb{Z}_n^*$  and it is required that  $h$  has *large order* in  $\mathbb{Z}_n^*$ . More precisely, we require that  $\text{ord}(h) \geq \frac{(p-1)(q-1)}{4}$ . This requirement is guaranteed to hold if  $\text{gcd}(h, n) = \text{gcd}(h^2 - 1, n) = 1$ , which can easily be confirmed by any user. In practice, we also need to be sure that the relative discrete logarithm between the  $h$  values of different SPs, or the same SP at different linkability windows, is unknown. For this reason, we let  $h$  be the result of a strong cryptographic hash function applied to a concatenation of  $d$  and the SP's name (where  $d$  is the index of the linkability window for which nymbles derived from  $h$  will be valid). In the unlikely event that the result of the hash does not satisfy the order requirement, the hash is applied iteratively until an appropriate value for  $h$  is produced.

Every pair  $(SP, \tau_{d,j})$  is associated with a *context element* denoted by  $\xi_{d,j}^{SP} \in \mathbb{Z}_\rho^*$ . As the notation suggests, this context element encodes the SP, linkability window, and time period for which a particular nymble is valid; without it, the SP has no way to distinguish, for example, nymbles issued for a time period that has already passed or those intended for a different SP altogether. The values for  $\xi_{d,j}^{SP}$  can be precomputed and must be known by both the NM and the SP, as they are required in the VERBS-Sign and VERBS-Verify protocols. The client must also know  $\xi_{d,j}^{SP}$  in order to verify its own nymbles. A reasonable value for  $\xi_{d,j}^{SP}$  might be as simple as a hash of the SP's name,  $d$ , and  $j$ .<sup>4</sup>

## 5.2 Credential Issuing Protocol

When a user Alice wishes to gain anonymous access to an SP, she must first prove possession of IP to obtain a valid signed CL-credential from the CM. The following protocol describes this process.

1. Alice connects directly to the CM; this proves to the CM that Alice is in possession of IP.
2. The CM computes  $x = \mathbf{MAC}_{sk}(\text{IP})$  and  $t_{\text{exp}} = t_{\text{cur}} + \Delta_t$ , where  $\mathbf{MAC}_{sk}(\cdot)$  denotes a Message Authentication Code keyed by the CM's private key  $sk$ <sup>5</sup>,  $t_{\text{cur}}$  is the current time, and  $t_{\text{exp}}$  is the expiration time of the credential to be issued. The tuple  $(x, t_{\text{exp}})$  is transmitted to Alice.

<sup>4</sup> The security requirement is that the cube root modulo  $\rho$  of the ratio of any two of the  $\xi$  should be computable only with negligible probability if the factorization of  $\rho$  is unknown.

<sup>5</sup> A MAC of Alice's IP address is used instead of her plaintext IP address to frustrate brute-force attacks performed by a colluding NM and SP.

3. The CM then issues Alice a CL-credential  $\mathbf{Cred}(x, t_{\text{exp}}) = (A, e, v)$  encoding  $x$  and  $t_{\text{exp}}$ , where  $e \in_R [2^{\ell_e-1}, 2^{\ell_e-1} + 2^{\ell_e-1}]$  is a randomly chosen prime,  $v \in_R \mathbb{Z}_{\phi(m)}$ , and  $A = \left( \frac{Z}{R_1^x \cdot R_2^{t_{\text{exp}}} \cdot S^v} \right)^{1/e} \bmod m$ . Recall that  $R_1$ ,  $R_2$  and  $S$  and  $Z$  are part of the CM's public key. Here  $\ell_e$  and  $\ell'_e$  are security parameters; see §6.1 and [3] for more details.

### 5.3 Nymble Acquisition Protocol

Once a valid credential is obtained from the CM, the next step is for Alice to compute a set of nymbles and receive VERBS on each of them from the NM. These nymbles are computed using values associated with the SP to which she wishes to gain access, the time period and linkability window in which they will be valid, and Alice's IP address. Alice may choose to request any number of nymbles, provided that this number does not cause her nymble set to span multiple linkability windows and does not exceed a predefined limit  $K$  imposed by the particular SP. Let  $k$  be the number of nymbles which Alice requests, and let  $j \geq 1$  be the index of the time period  $\tau_{d,j}$  within the current linkability window ( $d$ ), for which the first nymble will be valid.

1. Alice rerandomizes  $\mathbf{Cred}(x, t_{\text{exp}})$  as follows [3]:
  - (a) she chooses  $v' \in_R \{0, 1\}^{\ell_m + \ell_\theta}$ , where  $\ell_\theta$  is a security parameter;
  - (b) she computes  $A' = A \cdot S^{v'} \bmod m$  and  $v'' = v - ev'$  (in  $\mathbb{Z}$ ).
 Her rerandomized credential is then  $\mathbf{Cred}'(x, t_{\text{exp}}) = (A', e, v'')$ .
2. Alice computes the public value  $h$  using the name of the SP and the index  $d$  of the linkability window during which she wishes to connect. That is, she computes  $h = \text{hash}(d||\text{name})$ , where  $\text{name}$  is the canonical name associated with the SP. She also verifies that these values satisfy the order requirements from §5.1, and iteratively reapplies the hash function otherwise. She transmits  $(h, \text{name}, k, j)$  to the NM.
3. The NM verifies that  $k \leq K$  and  $j + k \leq \Gamma$ , and aborts otherwise.
4. Alice verifiably computes her unique *seed* value  $h_j = h^{x2^j}$  as follows:
  - (a) she picks a random  $\gamma \in \mathbb{Z}_n$ , computes the Pedersen commitment (to  $h_j$ )  $Y_j = a^{(h^{x2^j} \bmod n)} b^\gamma \bmod N$ , and transmits  $Y_j$  to the NM;
  - (b) next she performs the statistical zero-knowledge proof of knowledge

$$PK \left\{ (e, v'', x, t_{\text{exp}}, \gamma) : \begin{array}{l} Z \equiv R_1^x \cdot R_2^{t_{\text{exp}}} \cdot S^{v''} \cdot (A')^e \bmod m \\ \wedge t_{\text{cur}} \leq t_{\text{exp}} \\ \wedge Y_j \equiv a^{(h^{x2^j} \bmod n)} b^\gamma \bmod N \\ \wedge x \in \pm\{0, 1\}^{\ell_{\text{MAC}}} \\ \wedge t_{\text{exp}} \in \pm\{0, 1\}^{\ell_t} \\ \wedge e - 2^{\ell_e-1} \in \pm\{0, 1\}^{\ell'_e + \ell_\theta + \ell_H + 2} \end{array} \right\}.$$

In the case of a distributed CM, this proof is repeated once for each CM node, except that the third statement is replaced by a single nest proof to the sum of the  $x$  values received from each CM. The first statement

in this proof of knowledge convinces the NM that Alice does indeed possess a credential from the CM; the second statement asserts that this credential is not yet expired; the third statement (a nest proof) proves that  $Y_j$  does indeed encode the secret  $x$  from the credential and the first time period for which the credential should be valid; the remaining three statements are just length checks to show that the credential is validly formed. For full details on how this statistical zero-knowledge proof is performed we refer the reader to §4 and to [3].

If the proof succeeds, the NM is convinced that  $Y_j$  is a Pedersen commitment to  $h_j = h^x 2^j \bmod n$  and encodes the same secret  $x$  as Alice's credential; otherwise, the NM terminates. Note that the NM has learned no nontrivial information regarding the values of  $x$  and  $h_j$ .

5. Alice computes her sequence of nymbles using  $h_j$  as a seed value. This proceeds as follows:
  - (a) Alice computes the sequence  $(h_{j+i})_{i=1}^{k-1}$  where  $h_{j+i} = (h_{j+(i-1)})^2 \bmod n = h_j^{2^i} \bmod n$ . Note that given any element of this sequence, it is easy to compute the next element, but being able to compute the previous element is equivalent to factoring  $n$  [22, Chap. 3]. She computes Pedersen commitments  $Y_{j+i} = a^{h_{j+i}} b^{\gamma_{j+i}} \bmod N$  ( $\gamma_{j+i} \in_R \mathbb{Z}_n$ ) to each  $h_{j+i}$  and transmits them, along with zero-knowledge proofs of multiplication to show that they were computed correctly, to the NM.
  - (b) The NM verifies each of the proofs, and terminates if any proof fails.
6. Alice computes (but *does not send*) her nymbles  $\nu_{j+i} = g^{h_{j+i}} \bmod \rho$ , for  $0 \leq i < k$ . (Here, the exponent is just taken as an integer in  $[1, n]$ .) She computes  $(\alpha_{j+i}, S_{j+i}, \Pi_{j+i}) = \text{VERBS-Blind}(g, Y_{j+i}, \nu_{j+i}, \gamma_{j+i})$ , and sends each blinded value  $S_{j+i}$  and proof  $\Pi_{j+i}$  to the NM.
7. For  $0 \leq i < k$ , the NM computes (or looks up) its context element  $\xi_{d,j+i}^{SP}$ , and computes the blind signature  $\sigma'_{j+i} = \text{VERBS-Sign}(S_{j+i}, p, q, \xi_{d,j+i}^{SP}, \Pi_{j+i})$  (Recall that  $p, q$  is part of the NM's secret key, and  $\rho = pq$ .)
8. Alice unblinds the blind signatures  $\sigma'_{j+i}$  by computing  $\sigma_{j+i} = \text{VERBS-Unblind}(\sigma'_{j+i}, \alpha_{j+i})$  for  $0 \leq i < k$ .
9. If all steps are completed successfully, the tuple  $(\nu_{j+i}, \sigma_{j+i})$  is a valid nymble for time period  $j+i$  for linkability window  $d$  and the given SP. Alice can verify the validity of the nymble by checking  $\text{VERBS-Verify}(\nu_{j+i}, \sigma_{j+i}, \xi_{d,j+i}^{SP})$ .
10. If  $j + k - 1 \neq \Gamma$  (i.e.,  $\tau_{d,j+k-1}$  is not the last time period in the current linkability window), then Alice also computes  $\nu_\Gamma$ . This is the value that the NM will compute and place on the blacklist if Alice is, or becomes, banned from the SP. Note that the NM need not see or verify this value, nor provide a signature on it, since Alice will never be expected to present it to the SP.

#### 5.4 Nymble Showing Protocol

The nymble showing protocol is extremely simple; Alice presents her nymble to the SP, the SP confirms that it is valid, that the associated context element  $\xi_{d,i}^{SP}$  matches the current time period and linkability window, and that the nymble

does not appear on the linking list. If each of these conditions is met, Alice is granted access.

1. Alice anonymously queries the NM for the current version number of the blacklist, and computes the current linkability window and time period  $\tau_{d,i}$ .
2. Alice then connects anonymously to the SP and requests a copy of the blacklist  $\mathcal{B}$ . She confirms its legitimacy and that it is up-to-date by verifying the version number and a signature from the NM encoded in the blacklist. Once convinced of the freshness of the blacklist, she verifies that she is not presently blacklisted. More specifically, she checks that  $\nu_\Gamma \notin \mathcal{B}$ . If she discovers that she *is* on the blacklist, she disconnects immediately. In this case, the SP learns only that “some blacklisted user” attempted to connect.
3. If she is not on the blacklist, Alice transmits the tuple  $(\nu_i, \sigma_i)$  to the SP.
4. The SP consults the linking list  $\mathcal{L}$  and confirms that Alice is not on the blacklist by checking that  $\nu_i \notin \mathcal{L}$ . If this check fails, the SP terminates and Alice is denied access.
5. The SP confirms that the given nymble is valid for the current time period  $i$  and linkability window  $d$  by confirming that  $\text{VERBS-Verify}(\nu_i, \sigma_i, \xi_{d,i}^{SP})$  is true. If so, Alice is granted access for the remainder of the time period; otherwise, Alice is denied access.
6. The SP adds the tuple  $(\nu_i, \sigma_i, i)$  to a log file, so that if it determines at a later time (in the current linkability window) that Alice’s behaviour in  $\tau_{d,i}$  constitutes misbehaviour, it can present it to the NM to have Alice blacklisted.

## 5.5 Blacklisting Protocol

Suppose that Alice misbehaves in time period  $i^*$  and her misbehaviour is discovered in time period  $i'$  of the same linkability window. In this case, the SP initiates the following protocol with the NM to have Alice added to the blacklist.

1. The SP transmits the tuple  $(SP, \nu_{i^*}, \sigma_{i^*}, i^*, h, \mathcal{B}, \mathcal{L})$  to the NM.
2. The NM verifies that  $h$  is valid for the SP and the current linkability window, that  $\mathcal{B}$  and  $\mathcal{L}$  are up-to-date, and that  $\text{VERBS-Verify}(\nu_{i^*}, \sigma_{i^*}, \xi_{d,i^*}^{SP})$  is true. If so, the NM uses its private knowledge (the factorization of  $\rho = pq$  and the factorization of  $\phi(\rho)$  into  $\ell_B$ -bit primes) to solve the discrete logarithm of  $\nu_{i^*} = g^{h_{i^*}} \bmod \rho$  with respect to  $g$  to recover the exponent  $h_{i^*}$ ; otherwise, the NM terminates.
3. The NM then computes  $h_{i^*+1}, \dots, h_\Gamma$  using the recurrence equation  $h_{i+1} = h_i^2 \bmod n$  and computes  $\nu_{i'}, \dots, \nu_\Gamma$  as  $\nu_i = g^{h_i} \bmod \rho$ .
4. The NM computes the set  $L = \{\nu_{i'}, \nu_{i'+1}, \dots, \nu_\Gamma\}$  and then computes the new linking list  $\mathcal{L}' = \mathcal{L} \cup L$  and the new blacklist  $\mathcal{B}' = \mathcal{B} \cup \{\nu_\Gamma\}$ .
5. The NM increments the version number and signs the new blacklist and then returns both the signed blacklist and the linking list to the SP.<sup>6</sup>

<sup>6</sup> We adopt the same approach as the original Nymble system in order to ensure to the user that the blacklist they view is up-to-date, however, we omit many of the

## 6 Implementation

We have implemented the key components of our system in order to measure its performance. In the next subsection we discuss reasonable choices for various system parameters, while in the following subsection we present performance benchmarks carried out using these values.

### 6.1 Parameter Choices

First let us examine the relevant computations in more detail. In order to place a user on the blacklist, the NM needs to compute a discrete log in a trapdoor group. We intentionally make this non-trivial in order to deter bulk deanonymization (in the sense that the users would become linkable, but not have their identities revealed); our target is about one minute of computation (wall-clock time) per discrete log computation. We also seek to ensure that, without the NM’s private key, factoring and computing discrete logs mod  $\rho$  are infeasible; thus, we suggest setting  $\ell_\rho = 1536$ . The CM’s public key  $n$  should be as large as possible, while ensuring that  $n < \rho$ , so we pick  $\ell_n = 1534$ .

The discrete log computation takes about  $c \cdot \ell_\rho / \ell_B \cdot 2^{\ell_B/2}$  modular multiplications, which are almost completely parallelizable, for some constant of proportionality  $c$ . If the NM has a parallelism factor of  $P$  (i.e.,  $P$  is the number of cores available to the NM), this will be about  $\frac{\ell_\rho}{\ell_B} \cdot \frac{c \cdot 2^{\ell_B/2}}{P \cdot M}$  minutes to compute a discrete log, where  $M$  is the number of modular multiplications that can be computed by one core in one minute. So we want to choose  $\ell_B$  such that

$$\frac{2^{\ell_B/2}}{\ell_B} \approx \frac{T \cdot M \cdot P}{\ell_\rho \cdot c}, \quad (1)$$

where  $T$  is the desired wall-clock time (in minutes) to solve a discrete log. (In §6.2, we measure  $c \approx 0.57$  and  $M$  to be about 23.1 million for  $\ell_\rho = 1536$ .) Thus, for  $T = 1$  and  $P = 32$  we get  $\ell_B \approx 50$ ; for  $T = 1$  and  $P = 64$  we get  $\ell_B \approx 52$ .

On the other hand, it takes at least about  $\frac{3}{5} \cdot 2^{\ell_B}$  modular multiplications to factor  $\rho$ , taking advantage of its special form by using Pollard’s  $p - 1$  factoring algorithm [25]. However, this algorithm is inherently sequential [6]; only a small speedup can be obtained, even with a very large degree of parallelism.<sup>7</sup> This means it will take about  $\frac{3}{5} \cdot \frac{2^{\ell_B}}{M}$  minutes to factor  $\rho$ . Assuming  $M = 23100000$ , then  $\ell_B = 50$  yields over 55 years to factor  $\rho$ , and  $\ell_B = 52$  yields over 222 years to factor  $\rho$ . (Remember again that this is *wall-clock* time, not CPU time.) Note also that a different  $\rho$  can be used for each SP and for each linkability window, thus reducing the value of expending even that much effort.

---

details here for brevity. In a later version of Nymble the authors propose the use of “daisies” to ensure blacklist freshness. This approach could easily be used in our scheme as well. The interested reader should consult [20, 32] for these details.

<sup>7</sup> Of course, with *arbitrarily* large parallelism, other algorithms can factor  $\rho$  more quickly *without* taking advantage of the special form of  $\rho$ ; massively parallel trial division is an extreme example.

The reason we seem to be making the unusual claim that  $2^{50}$  security is sufficient is twofold: first, a minor point, these are counts of multiplications modulo an  $\ell_\rho$ -bit modulus, each of which takes about  $2^{12.8}$  cycles for our suggested  $\ell_\rho = 1536$ ; thus, we are really proposing about  $2^{62}$  security here. More importantly, these are counts of *sequential operations*. When one typically speaks of  $2^{80}$  security (of a block cipher, for example), one assumes that the adversary can take advantage of large degrees of parallelism, which is not the case here.

Moreover, as noted in [21, §4], since the complexity of factoring increases with  $2^{\ell_B}$  while the complexity of computing discrete logs increases with  $2^{\ell_B/2}$ , as cores get faster ( $M$  increases) and more numerous ( $P$  increases), the time to factor  $\rho$  only goes *up* with respect to the time to compute discrete logs. In particular, if  $M$  can be increased by a factor of  $f$ , then this leads to a net security increase of a factor of  $f$ , whereas if  $P$  can be increased by a factor of  $g$ , this leads to a net security increase of a factor of  $g^2$ . These calculations suggest that the Nymbler construction will get *even more secure over time*.

Suggested values for parameters related to CL-credentials are taken directly from [3]. In particular, reasonable choices are  $\ell_m = 2048$ ,  $\ell_\emptyset = 80$ ,  $\ell_e = 596$ ,  $\ell'_e = 120$ , and  $\ell_H = 256$ . We also suggest using  $\ell_{\text{MAC}} = 256$  and  $\ell_t = 24$ .

## 6.2 Performance Evaluation

In this subsection we present measurements obtained with our C++ implementation of the protocol. These include the average times taken to: 1) compute a nymble (and the associated proof of correct computation) at the client; 2) verify the client’s proofs and issue a VERBS at the NM; 3) verify the signature on a nymble at the SP; and, 4) solve an instance of the discrete log problem at the NM. In order to compute  $M$  used in the previous subsection, and for comparison with other restricted blind signature schemes, we also show the time required to compute modular multiplications and exponentiations, respectively. Note that the bulk of the computation in our scheme is in the Nymble Acquisition Protocol — particularly in computing and verifying the zero-knowledge proofs.

We emphasize that our implementation is incomplete and unoptimized; it is used simply to demonstrate that both the time-sensitive and cost-intensive

**Table 1.** Timings for essential computations in Nymbler.

Operation	Host	Mean Time	Trials	Reps/trial
Compute $k$ nymbles	Client	360 ms + 397 $k$ ms ( $R = .9974$ )	10	1
Issue $k$ blind signatures	NM	300 ms + 252 $k$ ms ( $R = .9803$ )	10	1
Verify signature	SP	11.2 $\mu$ s $\pm$ 0.3 $\mu$ s	10	100,000
Solve DL instance <sup>a</sup>	NM	25 m 38 s $\pm$ 2 m 16 s	10	1
Modular exponentiation <sup>b</sup>	—	403 $\mu$ s $\pm$ 6 $\mu$ s	10	100,000
Modular multiplication <sup>c</sup>	—	2.59 $\mu$ s $\pm$ 0.02 $\mu$ s	10	100,000

<sup>a</sup> This is the time to solve discrete logs with the parallel rho method on a single core and a 1536-bit 50-smooth modulus; using 32 cores should reduce this time to about 48 s  $\pm$  5 s. Solving for  $c$  in §6.1, Equation 1 with this value yields  $c \approx 0.57$ .

<sup>b</sup> Computed using a 1536-bit base, 160-bit exponent and 1536-bit modulus.

<sup>c</sup> Computed using random 1536-bit multiplicands and 1536-bit modulus; this yields  $M \approx 23,100,000 \pm 100,000$  modular multiplications per minute.

portions of our protocols can be carried out in an acceptable amount of time. In particular, there is still significant room for optimizations in our implementation of the VERBS-Blind algorithm and perhaps elsewhere in the protocols. For example, in order for the user to prove correct exponentiation of a committed value, our prototype implementation uses the naive “square-and-multiply” algorithm, but more efficient algorithms can be plugged in very simply. Moreover, all of our computations are single-threaded despite the highly parallelizable nature of the protocols. Finally, we note that the as-of-yet unimplemented components of the system are not expected to significantly alter these measurements.

The performance benchmarks in Table 1 were obtained on a 2.83GHz Intel Core 2 Quad Q9550 running Ubuntu 9.10 64-bit.

These measurements compare favourably with BLAC and PEREA. In BLAC, the time required for a user to construct a proof that she is not banned, and for the SP to verify the proof, scales linearly with the size of the blacklist. In [30] the authors give measurements which indicate that the cost is about 1.8 ms and 1.6 ms *per entry on the blacklist*, at the user and SP, respectively. Thus, when the blacklist reaches a size of 385 entries<sup>8</sup>, the cost per authentication in BLAC is roughly equal to the cost of obtaining a nymble in our scheme. (Half of the cost in our scheme is constant overhead which can be amortized over the cost of acquiring several nymbles.) We also note that, in this case, the cost at the SP is about 142 times higher in BLAC. In PEREA (with an authentication window of 30), [31] reports that an authentication takes about 160 ms at the SP (regardless of blacklist size) and up to 7 ms per entry on the blacklist for the user.

## 7 Conclusion and Future Work

We have presented a new system, inspired by Nymble, for providing an anonymous implementation of IP blocking over an anonymity network. Our approach is based on anonymous credentials, verifier-efficient restricted blind signatures, and a trapdoor discrete logarithm group. Compared to the original Nymble, our scheme severely limits the ability of malicious third parties to collude in order to break a user’s anonymity. Although our system is not as efficient as the original Nymble, most of the added cost has been introduced in the Nymble Acquisition Protocol; verifying a nymble’s authenticity at the SP is still very inexpensive.

One may pursue several directions to further improve our system.<sup>9</sup> For example, if the NM detects that a user has attempted to cheat in the Nymble Acquisition Protocol, we would like to be able to temporarily ban this user from any further use of Nymbler. One way that this could potentially be accomplished would be to use another Nymbler-like system (i.e., a *meta-Nymbler*) to allow blacklisting of Nymbler users from the system; however, this approach is

<sup>8</sup> An appendix in the extended version of this paper [18] gives recent usage and banning statistics from Wikipedia, which indicate that the average size of the blacklist is currently around 1200 entries — more than a factor of four larger than this figure.

<sup>9</sup> The interested reader should consult the extended version of this paper [18] for a more comprehensive list of future research directions.



actually overkill. Since this type of misbehaviour can always be detected *during* a session, a simpler technique, such as Unlinkable Serial Transactions [28], would be sufficient. We leave further investigation of this idea to future work.

Eventually, we envision that the CM services may be offered by the Tor directory servers, or the Tor entry nodes themselves, as they are already trusted with users' IP addresses. If we use P-signatures [2] instead of CL-signatures in the credentials obtained from the CM, each entry node can have its own public key, certified by the directory server, and the NM will not be able to tell which entry node certified the user's IP address. There are nontrivial issues with this simple proposal, however, and we leave addressing them to future work as well.

**Acknowledgments** We would like to extend our thanks to Aniket P. Kate, Greg Zaverucha, Jalaj Upadhyay, Mark Giesbrecht, Femi Olumofin, Urs Hengartner, Greta Coger, and the PETS 2010 anonymous reviewers for their helpful suggestions which greatly improved this paper. This work is supported by NSERC, MITACS, and a David R. Cheriton Graduate Scholarship.

## References

1. Algesheimer, J., Camenisch, J., Shoup, V.: Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In: Yung, M. (ed.) CRYPTO. LNCS, vol. 2442, pp. 417–432. Springer (2002)
2. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and non-interactive anonymous credentials. In: Canetti, R. (ed.) TCC. LNCS, vol. 4948, pp. 356–374. Springer (2008)
3. Bichsel, P., Binding, C., Camenisch, J., Groß, T., Heydt-Benjamin, T., Sommer, D., Zaverucha, G.: Cryptographic protocols of the Identity Mixer Library, v. 1.0. Computer Science Research Report RZ3730, IBM Research GmbH, Zurich (2009)
4. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: EU-ROCRYPT. pp. 431–444 (2000)
5. Brands, S.A.: Untraceable off-line cash in wallets with observers. In: Stinson, D.R. (ed.) CRYPTO. LNCS, vol. 773, pp. 302–318. Springer (1993)
6. Brent, R.P.: Parallel Algorithms for Integer Factorisation. Number Theory and Cryptography pp. 26–37 (1990)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EU-ROCRYPT. LNCS, vol. 2045, pp. 93–118. Springer (2001)
8. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes (1998)
9. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO. pp. 199–203 (1982)
10. Chaum, D.: Blind signature system. In: CRYPTO. p. 153 (1983)
11. Dingledine, R.: Tor development roadmap, 2008–2011. Roadmap, The Tor Project (2008)
12. Dingledine, R., Mathewson, N., Syverson, P.: Deploying low-latency anonymity: Design challenges and social factors. IEEE Security and Privacy 5(5), 83–87 (2007)
13. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium. pp. 303–320. USENIX (2004)

14. Dingleline, R. ([arma@freehaven.net](mailto:arma@freehaven.net)): Re: Banned from Slashdot, <http://archives.seul.org/or/talk/Jun-2005/msg00002.html>, [Private e-mail message to Jamie McCarthy; 01-June-2005]
15. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS. pp. 427–437. IEEE (1987)
16. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO. LNCS, vol. 263, pp. 186–194. Springer (1986)
17. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT. LNCS, vol. 4965, pp. 415–432. Springer (2008)
18. Henry, R., Henry, K., Goldberg, I.: Making a Nymble Nymble using VERBS. Tech. Rep. CACR 2010-05, Centre for Applied Cryptographic Research, Waterloo (2010), <http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-05.pdf>
19. Holt, J.E., Seamons, K.E.: Nym: Practical pseudonymity for anonymous networks. Internet Security Research Lab Technical Report 2006-4, Brigham Young University, Provo, UT (2006)
20. Johnson, P.C., Kapadia, A., Tsang, P.P., Smith, S.W.: Nymble: Anonymous IP-address blocking. In: Borisov, N., Golle, P. (eds.) Privacy Enhancing Technologies. LNCS, vol. 4776, pp. 113–133. Springer (2007)
21. Maurer, U.M., Yacobi, Y.: A non-interactive public-key distribution system. Designs, Codes and Cryptography 9(3), 305–316 (1996)
22. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
23. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with application to hash functions and discrete logarithms. In: ACM CCS. pp. 210–218 (1994)
24. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO. LNCS, vol. 576, pp. 129–140. Springer (1991)
25. Pollard, J.M.: Theorems on factorization and primality testing. Proceedings of the Cambridge Philosophical Society 76(03), 521 (1974)
26. RSA Laboratories: RSA Laboratories - the RSA factoring challenge FAQ, <http://www.rsa.com/rsalabs/node.asp?id=2094>, [Online; accessed 11-January-2010]
27. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO. LNCS, vol. 435, pp. 239–252. Springer (1989)
28. Syverson, P.F., Stubblebine, S.G., Goldschlag, D.M.: Unlinkable serial transactions. In: Hirschfeld, R. (ed.) Financial Cryptography. LNCS, vol. 1318, pp. 39–56. Springer (1997)
29. The Tor Project, Inc.: Tor: Overview, <https://www.torproject.org/overview.html.en>, [Online; accessed 21-October-2009]
30. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: Blacklistable Anonymous Credentials: Blocking misbehaving users without TTPs. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM CCS. pp. 72–81. ACM (2007)
31. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: PEREA: Towards practical TTP-free revocation in anonymous authentication. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS. pp. 333–344. ACM (2008)
32. Tsang, P.P., Kapadia, A., Cornelius, C., Smith, S.W.: Nymble: Blocking misbehaving users in anonymizing networks. IEEE TDSC (To appear) (2009)
33. Wikipedia: Wikipedia talk:blocking policy/tor nodes — Wikipedia, the free encyclopedia, [http://en.wikipedia.org/wiki/Wikipedia\\_talk:Blocking\\_policy/Tor\\_nodes](http://en.wikipedia.org/wiki/Wikipedia_talk:Blocking_policy/Tor_nodes), [Online; accessed 18-October-2009]