Nymbler: Privacy-enhanced Protection from Abuses of Anonymity

by

Ryan Henry

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2010

Abstract

Anonymous communications networks help to solve the real and important problem of enabling users to communicate privately over the Internet. However, by doing so, they also introduce an entirely new problem: How can service providers on the Internet—such as websites, IRC networks and mail servers—allow anonymous access while protecting themselves against abuse by misbehaving anonymous users?

Recent research efforts have focused on using **anonymous blacklisting systems** (also known as *anonymous revocation systems*) to solve this problem. As opposed to revocable anonymity systems, which enable some trusted third party to deanonymize users, anonymous blacklisting systems provide a way for users to authenticate anonymously with a service provider, while enabling the service provider to revoke access from individual misbehaving anonymous users *without* revealing their identities. The literature contains several anonymous blacklisting systems, many of which are impractical for real-world deployment. In 2006, however, Tsang *et al.* proposed Nymble, which solves the anonymous blacklisting problem very efficiently using trusted third parties. Nymble has inspired a number of subsequent anonymous blacklisting systems. Some of these use fundamentally different approaches to accomplish what Nymble does without using third parties at all; so far, these proposals have all suffered from serious performance and scalability problems. Other systems build on the Nymble framework to reduce Nymble's trust assumptions while maintaining its highly efficient design.

The primary contribution of this thesis is a new anonymous blacklisting system built on the Nymble framework—a *nimbler* version of Nymble—called Nymbler. We propose several enhancements to the Nymble framework that facilitate the construction of a scheme that minimizes trust in third parties. We then propose a new set of security and privacy properties that anonymous blacklisting systems should possess to protect: 1) users' privacy against malicious service providers and third parties (including other malicious users), and 2) service providers against abuse by malicious users. We also propose a set of performance requirements that anonymous blacklisting systems should meet to maximize their potential for real-world adoption, and formally define some optional features in the anonymous blacklisting systems literature.

We then present Nymbler, which improves on existing Nymble-like systems by reducing the level of trust placed in third parties, while simultaneously providing stronger privacy guarantees and some new functionality. It avoids dependence on trusted hardware and unreasonable assumptions about non-collusion between trusted third parties. We have implemented all key components of Nymbler, and our measurements indicate that the system is highly practical. Our system solves several open problems in the anonymous blacklisting systems literature, and makes use of some new cryptographic constructions that are likely to be of independent theoretical interest.

Acknowledgements

I would like to thank my supervisor Ian Goldberg for his collaboration and assistance with this research, my readers Urs Hengartner and Mark Giesbrecht for their many helpful suggestions, and my brother Kevin Henry for his role in the class project that spawned Nymbler. I am also extremely grateful to Greta Coger for her careful proofreading, which greatly improved the quality of presentation of this thesis.

Dedication

This thesis is dedicated to the two special ladies in my life: Lindsay and Mocha.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Chapter 1

Introduction

1.1 Background

Anonymous communications networks help to solve the real and important problem of enabling users to communicate privately over the Internet. The largest deployed anonymous communications network is a worldwide-distributed network of about 2000 volunteer-run *relays* called Tor [35, 81]. On an average day, Tor currently helps to protect between 100,000 and 300,000 privacy-conscious Internet users located in hundreds of countries around the world [58]. These users first connect to a *directory server* to obtain the list of online relays. They then form a random *circuit* (i.e., a path through some subset of the relays in the network) through which they route their communications. A typical Tor circuit passes through three relays; the last relay in the circuit is the *exit relay*. Before a packet is sent over the circuit, it is first encrypted in several layers, with each layer containing only the routing information necessary to deliver that packet to the next relay in the circuit (and eventually to its final destination). Each relay then strips off one layer of encryption and forwards the resulting packet on to the next. When a packet finally reaches the end of the circuit, the exit relay strips off the last layer of encryption and forwards the plaintext packet to its final destination. This approach, called *onion routing*, prevents a local adversary (who may be watching the user's Internet connection) from learning with which service providers—such as websites, IRC networks or mail servers—the user is interacting. It also prevents those service providers from learning the user's identity, location and IP address. The privacy that Tor provides serves many important purposes, as elaborated on the Tor Project website [82]: journalists use Tor to protect their sources; oppressed citizens use Tor to bypass government-level censorship; law enforcement agencies use Tor to protect the integrity of their investigations; and ordinary Internet users use Tor to protect themselves against irresponsible corporations, marketers, and identity thieves. Nonetheless, 'a few bad onions' take advantage of this anonymity for nefarious activities such as anonymously harassing users, trolling forums and chat rooms, and committing cyber-vandalism. In response, several service providers now prevent anonymous users from participating in, or contributing content to, their online communities; notable examples include Wikipedia [93], Slashdot [43], and most major IRC networks [80]. These service providers prevent access from Tor users by blacklisting the IP address of every Tor exit relay listed by the directory servers.

This is bad news for user privacy. If users are discouraged from using Tor because certain service providers are unavailable through it, then the privacy of those who still use Tor suffers in consequence. This is because the level of privacy that Tor can provide to a user depends on the size of that user's *anonymity set*.[1] In other words, the greater the number of Tor users, the greater the difficulty for an attacker to figure out which one of them initiated a particular connection. The number of volunteers that run Tor exit relays also affects the level of privacy that Tor can provide. This is because Tor is susceptible to *timing attacks*: if an attacker controls both the first relay (called an *entry guard*) and the exit relay in a user's circuit, then it can correlate (indeed, introduce) timing patterns in the streams passing through these two relays. This may enable such an attacker to infer which users are communicating with which service providers. The number of available exit relays for a user to choose from influences the probability that an attacker will control the exit relay that they choose. Unfortunately, one side effect of service providers blocking Tor exit relays by IP address is that this also blocks *nonanonymous* access from exit relay operators. The prospect of losing access to parts of the Internet likely discourages many would-be exit relay operators from volunteering.

Thus, there is a real and pressing need for a privacy-preserving mechanism that enables service providers on the Internet to allow anonymous access, while simultaneously protecting them against abuse by individual misbehaving anonymous users. Recent research efforts [11,13,14,54, 57,83–86,88–90] have focused on using **anonymous blacklisting systems** (which are sometimes called *anonymous revocation systems*) to fulfill this need. As opposed to revocable anonymity systems [55], which enable some trusted third party to deanonymize users, anonymous blacklisting systems provide users with a way to authenticate anonymously with a service provider (like Wikipedia), while enabling the service provider to revoke access from any users that misbehave. (The key difference is that in anonymous blacklisting systems *the user is never deanonymized*; neither the service provider, nor any other party, ever learns the identity of a revoked user.) This enables service providers to protect themselves against abuse by anonymous users in much the same way as they already protect themselves against abuse from nonanonymous users. In particular, anonymous blacklisting systems allow for *subjective revocation*, wherein each service provider is able to revoke a user's access to their own services at their own discretion. In contrast, revocable anonymity systems cannot allow subjective revocation, since this would enable service providers to deanonymize users at will.

This thesis presents a new anonymous blacklisting system called *Nymbler*. Nymbler improves on existing work by weakening the trust assumptions and providing stronger privacy guarantees together with some new functionality. Our approach avoids dependence on trusted

[1] A user's **anonymity set** is the largest set of users on the network such that the user is not identifiable within it.

hardware or unreasonable assumptions about noncollusion between trusted third parties to guarantee its security and privacy properties.[2] We have implemented the key components of Nymbler and our performance measurements indicate that the system is highly practical even for use at extremely large service providers like Wikipedia and Slashdot. We use a number of novel cryptographic constructions, including our distributed threshold Verinym Issuer, our Verifier-Efficient Restricted Blind Signatures (VERBS) scheme, trapdoor discrete logarithms, and a highly efficient zero-knowledge proof of non-membership in a list. These tools enable us to solve several open problems in the anonymous blacklisting systems literature.

1.2 Nymble Framework

Tsang *et al.* [54, 88–90] proposed Nymble as a solution to the problem of allowing service providers on the Internet 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 [13, 14, 50, 51, 57, 83–87].) Two of these schemes operate within the same general framework as Nymble; they change only low-level details to weaken the trust assumptions and to provide stronger privacy guarantees and some new functionality. We will return to both of these schemes in the following chapters and will not discuss them further here; we do remark here, however, that one of them is our own Nymbler system and is the primary contribution of this thesis. It shall be instructive, though, to first examine the Nymble framework in detail before focusing on our own proposal.

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

[2]As we will see later, our approach does not altogether eliminate trust in third parties; however, given some assumptions about the computational power of the third parties that do exist, we can quantify and minimize the level of trust that users of the system must place in them.

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**) that is computed deterministically 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. (See Figure 1.1.) 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. Given any nymble in U's sequence, the NM can always compute U's *last* nymble for the current linkability window; thus, we refer to this last nymble as U's **SP-specific pseudonym**. If U somehow abuses the services offered by the SP, then the SP can send 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 a **linking list** (there is a different linking list for each time period) and place U's SP-specific pseudonym on the SP's **blacklist**. By consulting the 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 1.1 illustrates the basic nymble construction.

At the start of each new linkability window, a change in system parameters causes all subsequent nymbles and SP-specific pseudonyms to change unlinkably (even to the NM). Thus, at the start of each linkability window, all SPs must reset their blacklist and forgive all prior misbehaviour. This brings *dynamism* and *forgiveness* to Nymble [54]; that is, it ensures that U's misbehaviour can (and eventually will) be forgiven without the NM or SP having the ability to subsequently track U.[3] On the other hand, from the perspective of the SP, it also limits the flexibility of the system. Thus, in our scheme we propose a mechanism by which SPs can enforce revocation that spans linkability windows.

Note that Nymble's security relies heavily on strong assumptions about the noncollusion of its two TTPs. In particular, if a malicious PM and NM collude then they can easily determine with which SPs U is interacting; further, if a malicious NM colludes with any number of SPs

[3]This is important because, for example, Wikipedia's blocking policy states that "IP addresses should rarely, if ever, be blocked indefinitely" since "IP address blocks can affect many users, and IPs can change" [94, "IP address blocks"].

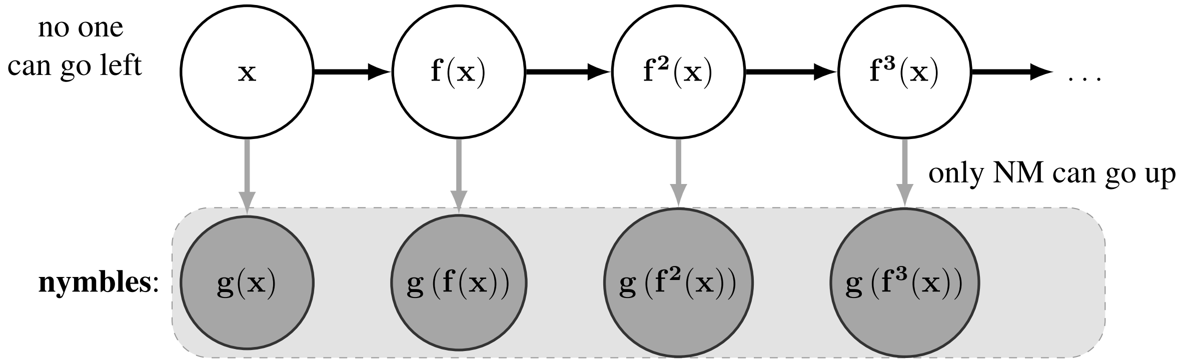

Figure 1.1: **Nymble construction:** In Nymble, the **black** arrows (i.e., $f(\cdot)$) are implemented with an HMAC and the **grey** arrows (i.e., $g(\cdot)$) are implemented with symmetric-key encryption. Each SP shares a secret key with the NM; the NM uses this to compute an HMAC on each nymble (not illustrated) to provide authentication for the SP. The initial value $\mathbf{x}$ is the result of an HMAC (with a linkability-window-specific key) applied to U's Nym and the SP's canonical name.

then they can easily link all of U's actions at those SPs. Indeed, if a malicious PM, NM, and SP combine these attacks, then U is completely deanonymized.

Nymble's sensitivity to attacks involving the PM exists because the PM computes U's Nym deterministically from a verinym[4] (her IP address). Although a one-way function maps IP addresses to Nyms, the space of valid IP addresses is cryptographically small; hence, given U's Nym and knowledge of the PM's one-way function, a polynomial-time adversary can easily determine U's identity with a brute-force attack. The output of this one-way function is, therefore, also a verinym for U. For the remainder of this manuscript we shall thus refer to U's Nym as her **verinym**, and to the entity that issues U's verinym as a **Verinym Issuer (VI)**.

To protect against attacks like the one above, we suggest a distributed (threshold) VI, and the following property, as security requirements of Nymbler and future Nymble-like systems.

Definition 1 (ZK-verinym)**.** A Nymble-like system satisfies the **zero-knowledge verinym** (ZK-verinym) property

1. if no single entity can compute the verinym associated with a given IP address, and
2. if no (possibly colluding) third parties can extract nontrivial information about U's verinym by observing her interactions with the system, provided U correctly follows the protocols. (In the case of a threshold VI, we assume that no colluding set of third parties contains a threshold number of VIs.)

[4]A **verinym** is any piece of identifying information—such as an IP address, credit card number, etc.—that can single a user out of a crowd of potential candidates [44].

The goal of this property is to minimize the potential for information leakage due to the use of verinyms.

Nymble is also sensitive to attacks involving the NM. The ZK-verinym property reduces this threat moderately because it prevents a malicious NM from linking U's actions at one SP with her actions at any other SP. On the other hand, the ZK-verinym property does nothing to prevent the NM from linking all of U's actions at a single SP. For this reason, we also suggest the following complementary property as a security requirement of Nymbler and future Nymble-like systems.

Definition 2 (ZK-pseudonym)**.** A Nymble-like system satisfies the **zero-knowledge pseudonym** (ZK-pseudonym) property

1. if during nymble acquisition, no party other than U herself learns any nontrivial information about the nymbles issued to U, and
2. if no entity is capable of extracting U's SP-specific pseudonym from a nymble without knowledge of the NM's secret key.

The goal of this property is to minimize the potential for proactive deanonymization by a malicious NM. If the ZK-pseudonym property is satisfied, then the NM must extract U's pseudonym from a nymble revealed by the SP before it is possible for the NM to link U's actions. Below, we will modify the framework in such a way as to render such wide scale deanonymization by a malicious NM infeasible.

At this point we observe that the NM in Nymble plays two related but distinct roles: on the one hand, the NM is responsible for issuing nymbles to users; on the other hand, the NM is responsible for revoking access using a trapdoor computation. Indeed, these two roles are logically orthogonal, and two distinct servers can fill them. Thus, we shall replace the NM by two separate entities: the **Nymble Issuer (NI)**, which is responsible for issuing nymbles to users, and the **Pseudonym Extractor (PE)**, which is responsible for extracting SP-specific pseudonyms from nymbles to revoke misbehaving users. Figure 1.2 depicts each of the various parties in our extended Nymble framework and illustrates the interactions between them.

The ZK-pseudonym property protects against a malicious NI, but it does not protect against a malicious PE that has sufficient resources to extract many SP-specific pseudonyms from nymbles in real time. In this case, the SP can immediately forward every nymble it receives to the PE for instant deanonymization. To combat this threat, we propose a trapdoor computation with tuneable computational cost; in particular, given t_{dl}, the desired wall-clock time per trapdoor computation, as input, the system should be able to output a trapdoor function that takes t_{dl} time to evaluate on average. Moreover, the PE should be able to prove in zero-knowledge that random problem instances actually do require this much time to solve (on average) with the trapdoor. If the PE revokes at most K users on average per L minutes, then t_{dl} can be set to just under L/K minutes. This renders wide-scale deanonymization economically infeasible for SPs with

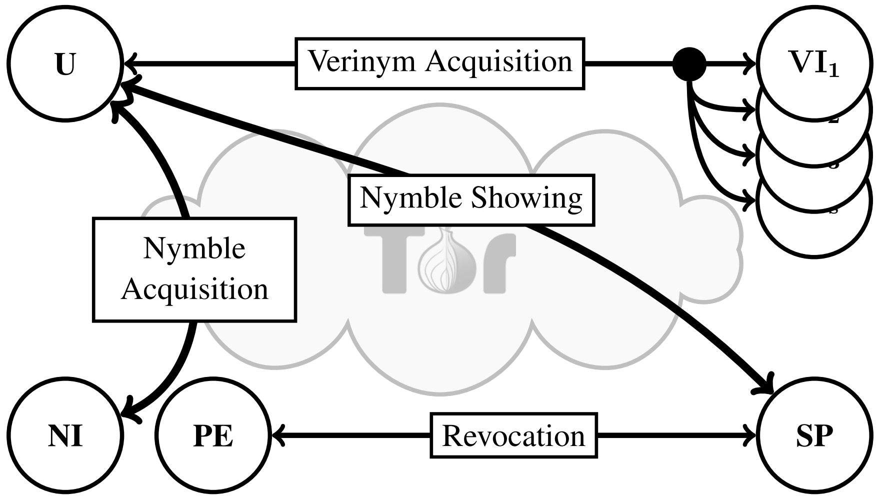

Figure 1.2: **Architecture of our extended Nymble framework:** This figure illustrates the parties in our extended Nymble framework and the interactions between them. Arrows that pass through the Tor cloud represent anonymous connections, while those that do not pass through the Tor cloud represent direct (nonanonymous) connections.

sufficiently high traffic volumes. To facilitate this, we propose a new list called a **revocation queue**, which works as follows: as soon as the SP files a complaint with the PE, the PE places the associated nymble on the revocation queue. If the revocation queue is non-empty, then the PE is always processing the request at the front of the queue; when the PE finishes extracting an SP-specific pseudonym from the nymble at the front of the queue, it pops the entry from the queue and adds the SP-specific pseudonym to the blacklist. As appropriate, it also adds nymbles to the SP's linking lists (starting from the time period immediately *after* the complaint is filed). This enables the SP to trace (and clean up after) any actions performed by U between the time that a complaint was filed against her with the PE and the time that her SP-specific pseudonym was actually added to the blacklist. Thus, when U consults the blacklist, she must also consult the revocation queue to ensure that her present actions will not soon become linkable.

In Nymble, the NI always issues U with the entire sequence of nymbles for her verinym, right up to the end of the current linkability window. In Nymbler, we will instead 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 a verinym to U, this verinym is only valid until some future VVP; no nymble will be issued to U for any time period in a VVP after her verinym expires. This capability allows greater flexibility in the choice of system parameters. For example, if U obtains her IP address through DHCP, then she may receive a new address daily; some SPs, however, may wish to use, say, one-week linkability windows. With VVPs the VIs can set the duration of a linkability window to one week while still requiring U to renew her verinym each day. Note that by changing input parameters to the function that maps verinyms to nymble seeds (i.e, the function that outputs the value x from Figure 1.1), particular SPs are able to choose a duration for their own linkability windows that is *shorter* than the duration set by the VIs (but not longer). However, for ease of presentation, we will assume that all SPs use the global linkability window duration set by the VIs.

IP addresses as a unique identifier. Both the original Nymble and our Nymbler system use U's IP address as a verinym. Thus, if two different users share the same IP address (such as two users who share a common terminal, or two users who share an Internet connection via NAT [70]), then they will also *share a common identity* in the system. Likewise, if a single user has access to two different IP addresses (such as by having access to two different terminals on different Internet connections, or via DHCP [37]), then they can assume *two different identities* in the system. We argue that this limitation is acceptable for two reasons: first, IP address blocking has the same limitation *even when the user is not connecting through an anonymous communications network*, yet IP address blocking is still the *de facto* standard method for revoking access from (unauthenticated) abusive users; and second, the design of Nymble and Nymbler does not necessitate the use of IP addresses as a unique identifier. Indeed, if future research reveals a viable alternative to blocking users by IP address, it should be trivial to modify these schemes to support this new method. In several prior works, the authors have considered alternative resources to distinguish between diffferent users; we refer the interested reader to one of our earlier papers [49] for an overview. We emphasize, however, that at present, IP addresses seem to be the only practical solution in the literature, and that the proposal of a superior (and still practical) alternative will have much broader implications for protecting against Sybil attacks in a more general context [36].

Infrastructure providers. Several key people involved with The Tor Project have recently acknowledged the need for a real-world deployment of an anonymous blacklisting system [31, 34]. Therefore, it may be reasonble to expect that the operators of Tor (and its financial sponsors) would be willing to provide the infrastructure necessary to deploy and maintain such a system in the wild.

Chapter 2

System Requirements

This chapter proposes a set of security and privacy properties that Nymble-like systems should possess to protect: 1) U's privacy against malicious SPs and third parties (including other malicious users), and 2) SPs against abuse by malicious users. We then propose a set of new performance requirements that should be satisfied to maximize any anonymous blacklisting system's potential for real-world adoption, and provide formal definitions of some optional features already found in the literature on anonymous blacklisting systems.

2.1 Security Requirements

Each of the security properties presented in this section has previously appeared in the literature on anonymous blacklisting systems; however, the formal definitions we propose here are our own. We formally define each security property in terms of a security game played against a probabilistic polynomial-time adversary. A function $\epsilon(\cdot)$ is *negligible* if, for all $c > 0$ there exists a κ_0 such that $\epsilon(\kappa) < 1/\kappa^c$ for all $\kappa > \kappa_0$. An event occurs with *negligible probability* (resp. *overwhelming probability*) if the probability that the event occurs is bounded above by $\epsilon(\kappa)$ (resp. below by $1 - \epsilon(\kappa)$), where ϵ is a negligible function and κ is an appropriate security parameter. The formal definitions of the security games are located in Appendix A.

Definition 3 (Correctness)**.** The system is **correct** if, with overwhelming probability[1], an honest SP will accept any nymble from an unrevoked user, as long as it is properly generated according to the established protocols of the system.

[1]We cautiously define correctness in terms of overwhelming probabilities (instead of with absolute certainty) because, with negligible probability, the one-way functions used in a scheme may have collisions. Thus, with negligible probability, an authentication request by U may fail because of a collision with some revoked user.

Misauthentication resistance

Informally, we define *misauthentication resistance* as follows: with overwhelming probability, verification should succeed *only* on those nymbles that are the result of a user correctly executing the established protocols. Note that our definition of misauthentication resistance does not try to capture the notion of *revocability* (Definition 7). Formally, we define misauthentication resistance in terms of Security Game A.1 in Appendix A.

Definition 4 (Misauthentication resistance)**.** The system provides **misauthentication resistance** if no probabilistic polynomial time adversary can win Security Game A.1 with non-negligible probability.

Backward anonymity

Informally, we define backward anonymity as follows: given a nymble from some member of a set of at least two users, it should be infeasible for an attacker to determine which user that nymble belongs to, with more than negligible advantage over a random guess. Moreover, this property should hold even if some SPs later revoke any subsets of these users. Formally, we define backward anonymity in terms of Security Game A.2 in Appendix A.

Definition 5 (Backward anonymity)**.** The system provides **backward anonymity** if no probabilistic polynomial time adversary can win Security Game A.2 with probability non-negligibly greater than $1/2$.

Unlinkability

Informally, we define *unlinkability* as follows: given two or more nymbles from some set of at least two users, it should be infeasible for an attacker to distinguish nymbles that belong to the same user from those that belong to different users, with more than negligible advantage over a random guess. This property should hold both within a single SP and among multiple SPs. Formally, we define unlinkability in terms of Security Game A.3 in Appendix A.

Definition 6 (Unlinkability)**.** The system provides **unlinkability** if no probabilistic polynomial time adversary can win Security Game A.3 with probability non-negligibly greater than $1/2$.

Revocability

Informally, we define *revocability* as follows: given one of U's nymbles from an authentication with an SP, it should be possible for that SP to have U's access revoked. This mechanism should have the property that no coalition of revoked (or unregistered) users should be able to produce a nymble that the SP will accept. Revocability is related to—but distinct from—the previously introduced notion of misauthentication resistance (Definition 4). Formally, we define revocability in terms of Security Game A.4 in Appendix A.

Definition 7 (Revocability)**.** The system provides **revocability** if no probabilistic polynomial time adversary can win Security Game A.4 with non-negligible probability.

Revocation auditability

Informally, we define *revocation auditability* as follows: U should be able to check her revocation status at an SP before trying to authenticate. If revoked, U can then disconnect without revealing any nymbles to the SP. This is important to avoid the situation in which a malicious SP accepts nymbles from U after she is revoked, thus tracing U's actions without her knowledge. Formally, we define revocation auditability in terms of Security Game A.5 in Appendix A.

Definition 8 (Revocation auditability)**.** The system provides **revocation auditability** if no probabilistic polynomial time adversary can win Security Game A.5 with probability non-negligibly greater than $^1/_2$.

Non-frameability

Informally, we define *non-frameability* as follows: no coalition of third parties should be able to get U revoked from an SP.[2] This definition assumes that no coalition contains the PE or a user that shares U's IP address. Formally, we define non-frameability in terms of Security Game A.6 in Appendix A.

Definition 9 (Non-frameability)**.** The system provides **non-frameability** if no probabilistic polynomial time adversary can win Security Game A.6 with non-negligible probability.

Remarks

1. Backward anonymity and unlinkability together imply the usual notion of **anonymity** for unrevoked users; adding revocation auditability makes this full anonymity for all users (revoked or otherwise).

[2]A coalition of third parties may be any subset of the following: the VIs, the NI, any *other* SPs, and all other users.

2. Roger Dingledine raised the following question [32]: if some subset of users chooses to use the system pseudonymously (e.g., by participating in online chat rooms or logging in to a website with a persistent alias), what is the privacy impact on the other users? With our definitions, pseudonymous users can be considered to be under adversarial control (see Appendix A); thus, if the system provides backward anonymity and unlinkability, then there is essentially no impact on user privacy.

2.2 Performance Requirements

The security requirements outlined in the previous section are necessary but not sufficient for a useful Nymble-like system. We further believe that all anonymous blacklisting systems should possess certain crucial performance characteristics. Our requirements contain a bias towards producing an extremely lightweight component for the SP; we do this because many SPs appear to consider the input of anonymous users to be of generally low quality, and are thus content to block access from anonymous communications networks. To maximize the system's potential for real-world adoption it is therefore important to cap the level of computational and communications overhead for the SP; we call this property *verifier efficiency*. The system should also take care to avoid affecting the observed interaction latency between U and the SP by ensuring that any computation that U must perform to authenticate is independent of the blacklist size, and by forcing significant computations to be precomputable. We call this latter property *user efficiency*.

Verifier efficiency

Informally, we define *verifier efficiency* as follows: the benefits that the system brings to an SP must clearly compensate for any burden placed on the SP. As such, the SP's cost in terms of communications and computational complexity, storage requirements, hardware requirements, and maintenance costs, should be extremely low.

Definition 10 (Verifier efficient)**.** The system is **verifier efficient** if the cost of the system for an SP is low. In particular,

1. the computational costs for the SP to verify U's nymbles or have U's access revoked are both small and independent of the number of (revoked or unrevoked) users,
2. the code required for the SP is small and contains no computationally expensive operations, and
3. the SP does not require any specialized hardware.

User efficiency

Informally, we define *user efficiency* as follows: the system should be accessible to all users and should not negatively affect users' online experiences. We note that one of Tor's target audiences is citizens of countries with oppressive leadership that censors access to certain information. Unfortunately, users from these countries may not have access to state-of-the-art computers or high-bandwidth Internet connections. Thus, requiring U to run specialized hardware like a trusted platform module (TPM), to consume lots of bandwidth, or to solve computational puzzles limits the system's usefulness.

Definition 11 (User efficient)**.** The system is **user efficient** if the cost for U to use the system is low. In particular,

1. the computational cost of authenticating with an SP is independent of the number of (revoked or unrevoked) users,
2. any computationally expensive operations required by U can be precomputed, and
3. U does not require any specialized hardware.

2.3 Optional Features

Some anonymous blacklisting systems may offer other useful features. We propose formal definitions for four optional features already found in the literature. Note that our proposed Nymbler scheme supports each of the features discussed in this section.

Objective revocation

Currently, all Nymble-like systems in the literature support the notion of *subjective revocation*, wherein the SP decides subjectively and unilaterally whether to revoke U. With *objective blacklisting systems*, however, the PE can only revoke U if she violates a *contract* that specifies the SP's terms of service [72–74]. In this case, there exists a function $M : \{0,1\}^* \to \{\texttt{true}, \texttt{false}, \perp\}$, called a *morality function*, which takes as input a bit string $proof$ describing U's behaviour and outputs a boolean value (or $\perp$). The output indicates whether the behaviour described in $proof$ violates the contract (if $\perp$ is returned, this means that subjective judgment is required to make a final determination).

Definition 12 ((Strictly) objective blacklisting)**.** The PE is said to **enforce a contract** on SP if the PE will only extract pseudonyms from a nymble if the SP provides a string $proof$ that is provably associated with the nymble and for which $M(proof) \neq \texttt{false}$. In this case, the system is said to support **objective blacklisting**; if the range of M is restricted to the set $\{\texttt{true}, \texttt{false}\}$ then the system is said to enforce **strictly objective blacklisting**.

The system is said to enforce **contract-based revocation** if the enforced contract is encoded in each nymble and is known to and agreed upon by both U and the SP at the time of nymble acquisition. It provides **contract auditability** if U knows the precise semantics of the morality function (thus enabling U to determine if a specific activity constitutes misbehaviour before deciding whether to engage in it).

Morality functions. Online forum software packages routinely incorporate mechanisms that can automatically filter some forms of unwanted behaviour; for example, they often filter vulgar language, scripts, external links, and unsolicited advertisements. Nonetheless, nearly all active forums also have moderators that enforce their less-tangible rules; this is because it is difficult or infeasible to write an algorithm that can effectively capture certain behaviours. Moreover, users routinely find ways to circumvent existing mechanisms, even for simple situations like swearing and posting external links, and such circumvention is likely to be at least as easy in situations where the disallowed behaviour is more complex. To the extent that we can formally specify and computationally verify a nontrivial morality function, contract-based revocation risks being overly restrictive to the SPs on which it is enforced. Thus, constructing a useful morality function appears to be a difficult problem to solve in practice; it is therefore unclear how practical objective blacklisting is in many real-world situations.

Nonetheless, our Nymbler construction supports a limited form of contract-based revocation that is inspired by the objective blacklisting extension to Jack [57]; however, we do not attempt to develop full details for the specification of a morality function.

Rate-limited access

SPs often find it useful to rate-limit U's access; this limits the amount of disruption a single user can cause. Many large SPs use rate limiting even for non-anonymous users. We return to the previous example of online forums: to keep spam levels reasonably low, forums often rate-limit the posting of new messages. The original Nymble [54, 88, 89] incorporates rate limiting by design, while subsequent anonymous blacklisting systems [57, 83–86] have proposed the use of an auxiliary scheme such as k-Times Anonymous Authentication [79] or Periodic n-Times Anonymous Authentication [15] to provide rate limiting.

Definition 13 (Rate limiting)**.** A system provides **rate limiting** if, for a given interval of time T, the number of pair-wise mutually unlinkable nymbles that U can use at a given SP is bounded above by some monotone increasing (in the length of T) function.

Inter-window 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 [94]. 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; revocation that spans two or more linkability windows is called **inter-window revocation**. The notion of inter-window revocation was first introduced by Lin and Hopper [57] for their Jack scheme, although the authors did not refer to it by this name in that work.

Blacklist transferability

Commonly on the Internet a single organization operates several different websites. For example, the Wikimedia foundation [92] hosts several popular websites: Wikipedia [93], Wiktionary [96], Wikiquote [95], 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*, which would not be possible in the above setup.

Definition 14 (Blacklist transferability)**.** The system provides **blacklist transferability** if U can authenticate unlinkably and concurrently with SP_1 and SP_2 while SP_1 can force U to prove that she is not revoked from SP_2 before granting her access.

Tsang *et al.*'s Blacklistable Anonymous Credentials scheme [83–85] appears to be the only prior anonymous blacklisting system to support blacklist transferability. Note the similarities between blacklist transferability and inter-window revocation; we will later exploit this relationship in Nymbler to support blacklist transferability through our inter-window revocation mechanism.

Chapter 3

Related Work

This chapter discusses existing anonymous and pseudonymous blacklisting systems in the literature, as well as some (restricted) blind signature schemes.

3.1 Anonymous and Pseudonymous Blacklisting Systems

We divide our discussion of anonymous and pseudonymous blacklisting systems into three separate categories: the *pseudonym systems*, the *Nymble-like systems*, and the *revocable anonymous credential systems*.

3.1.1 Pseudonym Systems

The first class of blacklisting systems are the **pseudonym systems**. As the name implies, pseudonym systems provide U with pseudonymity instead of full anonymity. That is, U's identity at an SP is not linkable back to her real identity (nor are U's identities at different SPs linkable with each other), but her individual actions at a particular SP are all easily linked with each other. Because U interacts with the SP using a persistent pseudonym, revocation is as simple as adding the pseudonym to a blacklist and denying access to any user with a pseudonym on the blacklist. Existing pseudonym systems get their security and privacy properties from one of three primary sources: 1) *private credentials*, 2) *blind signatures*, or 3) *group signatures*.

Schemes based on private credentials

Chaum [24] proposed pseudonym systems as a way for U to control the transfer of information about herself between different organizations. To enable this, he proposed that U first establish a pseudonym with every organization with which she wishes to interact. Then, to transfer information about herself from one organization to another, U obtains a *credential on a pseudonym* from the first organization, which encodes this information. U then transforms this credential into the "same" credential on one of her other pseudonyms. This enables her to prove to a second organization that the first organization has certified the information encoded in the credential, without necessarily revealing information about her pseudonym at the first organization. Chaum and Evertse [25] presented the first construction of a pseudonym system based on RSA in the year following Chaum's proposal. Shortly thereafter, Damgård [29] proposed a provably secure (assuming the existence of claw-free functions)—though impractical—scheme based on zero-knowledge proofs. Later, Chen [27] proposed a practical construction for Damgård's model based on discrete logarithms.

In her Master's thesis, Lysyanskaya [60, 62] presented a new model for pseudonym systems that incorporates the ability for an organization to revoke access to a credential on a pseudonym; thus, her model makes blacklist transferability possible. That is, SP_1 can require U to show a credential indicating that she has authorized access to SP_2. If U is later revoked at SP_2, then her credential will also be revoked, thus preventing her from showing it to SP_1 in the future. Moreover, by having SP_1 verify that U possesses a non-revoked access credential *for SP_1 itself*, a fully anonymous scheme can be built by allowing U to *rerandomize* her pseudonyms between showings. Camenisch and Lysyanskaya [16] extend the idea to do just that, resulting in the first of the *revocable anonymous credential systems*.

Schemes based on blind signatures

The first use of pseudonym systems specifically as a revocation mechanism appears to be by Holt and Seamons [52]. Their Nym system was proposed as a way to enable the popular collaborative website Wikipedia [93] to revoke access from misbehaving Tor users. It does away with much of the sophisticated functionality offered by the earlier schemes based on private credentials to build an extremely simple mechanism for U to establish pseudonyms with an SP. Nym was the first scheme to issue pseudonyms based on a unique identifier (they recommend U's IP address or email address). In Nym, U proves possession of her unique identifier in exchange for a blind RSA signature on a (user specified) random nonce. She later exchanges the unblinded signature for a client certificate which, because of the unconditional hiding of blind RSA signatures, is completely unlinkable to her real identity. Abbot *et al.* [1, 2] describe a similar system, called Closed Pseudonymous Groups (CPG), wherein members of some 'real-world' group (for example, students of the same class or subscribers to a service) register pseudonyms to participate in a closed online community. Since pseudonymous access in their system is restricted only to members of a certain 'real-world' group, Abbot *et al.* discuss approaches to revoking U based on her real-world actions (for example, if she drops the class or lets her membership to the service lapse).

Schemes based on group signatures

In 1991, Chaum and van Heyst [26] proposed group signatures, wherein each member of a group can sign any message *on behalf of the group*. Anyone can verify a group signature using the group's public key, but only a special entity known as the **Revocation Manager (RM)** can determine which group member produced a particular group signature. (Sometimes the RM is the same entity that distributes private keys, in which case it is typically called the **Group Manager (GM)**.) If the RM is trusted, then group signatures make it easy to construct a closed community in which non-revoked users are fully anonymous (within the anonymity set of all non-revoked members of the group)[1]. If U misbehaves, then the RM can link her past and future actions and thus revoke her anonymity.

Recently, Schwartz *et al.* [72–74] proposed contract-based revocation. They leverage ideas from trusted computing to construct a contract-based revocation system, called RECAP, using group signatures as the underlying primitive. In particular, they use remote attestation to allow U to confirm that the software running on the RM will only deanonymize her in the event that she violates a pre-agreed-upon (by U and the SP) contract. Their reliance on trusted computing means that user privacy is not arbitrarily entrusted to the RM.

Remark. We reiterate that pseudonym systems provide users with pseudonymity instead of full anonymity. Therefore, even under normal operation, users (misbehaving or otherwise) are subject to a loss of privacy as compared to using an anonymous communications network *without* a pseudonym system. The remaining classes—i.e., Nymble-like systems and revocable anonymous credential systems—improve on pseudonym systems by adding unlinkability and hence full anonymity.

3.1.2 Nymble-like Systems

The second class of anonymous blacklisting systems are the **Nymble-like systems**. This class gets its name from Nymble [54, 88, 89]. Nymble was already presented in detail in §1.2 so we omit a detailed description here. Since its proposal in 2006, there have been two additional proposals for Nymble-like systems in the literature.[2] Our own Nymbler system—which appeared as [50, 51], but which we present here in a significantly improved form—was the first of these proposals. The second scheme is Lin and Hopper's Jack [57]. However, before discussing these systems further we briefly examine a predecessor to Nymble called Unlinkable Serial Transactions (UST), first proposed by Syverson *et al.* [77, 78] in 1997.

[1]We place schemes based on group signatures in the class of pseudonym systems due to the fact that, upon revocation, all of U's actions at an SP become linkable, thus turning her anonymity into pseudonymity.

[2]Recently, a third Nymble-like system due to Lofgren and Hopper [59], called BNymble, was accepted for publication; however, because we have not yet been able to obtain a copy of this paper, we will not discuss it here.

UST is a protocol for online subscription services that prevents the SP from tracking the behaviour of a subscriber U while protecting it from abuse due to simultaneous active sessions by U's subscription. UST introduced the concept of having U authenticate with temporally related—but mutually unlinkable—authentication tokens. In the scheme, U and the SP negotiate a blind authentication token that U later exchanges for services from the SP. At the end of U's session, she and the SP negotiate a new blind authentication token for her next session. Thus, at any given time, U possesses just one valid and unused token; this prevents a second anonymous user from using U's subscription at the same time as U. If U is judged by the SP to have misbehaved (for example, by attempting to use the same token twice concurrently), then the SP can revoke U's access by ending her session without issuing a new authentication token. However, due to UST's token generation method, the scheme provides no way for the SP to revoke U for misbehaviour that is recognized after her session has ended. This makes UST an unsuitable blacklisting system for many real-world applications.

Lin and Hopper [57] proposed Jack in late 2010. Jack builds on the Nymble framework to weaken Nymble's trust assumptions. It reduces the level of trust in the VI (called the **Credential Manager (CM)** in that scheme) by eliminating the link between U's actual identity and her nymbles. U's Nym (we revert to this terminology since a pseudonym in Jack is decidedly not a verinym) in Jack is instead based on user-chosen randomness; thus, the ZK-verinym property is easily satisfied. The VI in this case must keep a log whenever it issues a credential to a user; it must then refuse to issue a second credential to any user who already appears in the log. In the case of a distributed VI, it is important that each VI possesses an up-to-date copy of the log at all times; indeed, care must be taken to avoid race conditions that occur if U requests credentials from several VIs concurrently. We also note that this approach to credential issuing does not work well with using IP addresses to identify users. This is because IP addresses are neither permanent nor unique [94]; some users may regularly obtain new IP addresses via DHCP, and some users may share the same IP address via NAT. In both instances, legitimate users are likely to encounter availability problems.[3] Lin and Hopper suggest leveraging an existing public key infrastructure (PKI) so that client certificates can identify users.

Jack also eliminates the role of the NI. Instead, U computes her own nymbles on demand using Camenisch and Shoup's Verifiable Encryption of Discrete Logarithms [20]. Along with each nymble, she also computes a zero-knowledge proof to convince the SP that: a) the output is an encryption of her correctly formed SP-specific pseudonym, and b) her SP-specific pseudonym is not on the blacklist. The blacklist in Jack takes the form of a cryptographic accumulator; thus, proof that an SP-specific pseudonym is not on the blacklist takes the form of a proof of non-membership in an accumulator. This ensures that authentication times are independent of the blacklist size at the SP. However, the cost of this approach is still considerably higher for the SP as compared to Nymble and Nymbler.

[3]As more Internet services transition to IPv6 [30], these problems may all but vanish; at present, however, they are real problems that one must consider when using IP addresses to identify users.

Jack also introduces some new functionality to the Nymble framework. For example, Jack is the first Nymble-like system to support a limited form of contract-based revocation. Moreover, rate limiting in their scheme is an optional add-on (it is mandatory in Nymble and Nymbler) and they utilize blacklists of different durations to provide a weak form of inter-window revocation. As originally presented, our Nymbler scheme did not support any of these features (except for rate limiting); however, the improved version we present here has similar support for contract-based revocation and versatile inter-window revocation.

Both Nymble and Jack are highly susceptible to linking attacks by a malicious NI/PE. In Nymble, the malicious NI just sends all nymbles it computes to the SP, which results in instant deanonymization of all users. In Jack, the attack works in reverse: the SP just sends all nymbles it receives to the PE to have their SP-specific pseudonyms extracted. While the attack in Jack is more costly than in Nymble, an adversarial PE can still deanonymize all Jack users in real time. This is the case because extracting an SP-specific pseudonym in Jack takes about 26 ms of computation [57]; thus, a malicious PE can deanonymize about 2,300 authentications *per minute* using a single processor. On the other hand, the work required by the SP to verify the zero-knowledge proofs that accompany each nymble is nearly 18 times that which is required for the PE to extract SP-specific pseudonyms from those nymbles. This means that an SP can only support about 128 authentications per minute on the same hardware. Our Nymbler scheme protects against a malicious NI by enforcing the ZK-verinym and ZK-pseudonym properties, while rendering the cost of attacks by a malicious PE economically infeasible with a tuneable-cost trapdoor function for pseudonym extraction.

The cost of authentication (that is, verifying that a nymble is valid and checking its revocation status) at the SP is constant in all three Nymble-like schemes. In Nymble, the SP computes an HMAC to verify that the nymble is valid and consults a hash map (with constant amortized lookup time) to ensure that U's SP-specific pseudonym is not on the blacklist. In Nymbler, the SP checks a *verifier-efficient restricted blind signature* to verify that the nymble is valid, and consults a hash map (much like Nymble) to ensure that U's SP-specific pseudonym is not on the blacklist. In Jack, the SP verifies two zero-knowledge proofs: one proof certifies that the nymble is valid; the second certifies that U's SP-specific pseudonym does not appear on the blacklist. Computation for U is essentially zero in Nymble; it is constant—though somewhat higher than in Nymble—in Nymbler. (Fortunately, most of the additional computation in Nymbler is precomputable.) In Jack, U must perform $O(|\Delta\mathcal{B}|)$ modular multiplications and exponentiations for each authentication, where $\Delta\mathcal{B}$ is set of updates to the SP's blacklist since U's last authentication.

3.1.3 Revocable Anonymous Credential Systems

The third class of anonymous blacklists systems are the **revocable anonymous credential systems**. These schemes take a heavyweight approach to security and privacy by completely replacing TTPs with zero-knowledge proofs (ZKPs). Unfortunately, the high computational overhead associated with them means that they are often of theoretical interest only.

As noted in our discussion of pseudonym systems based on private credentials, the first scheme in this class is the anonymous credential system of Camenisch and Lysyanskaya [16]. Since the introduction of the Camenisch-Lysyanskaya credential system, several other general-purpose anonymous credential systems with revocation capabilities have appeared. Our focus here is only on those that specialize specifically as anonymous blacklisting systems.

Brands *et al.* [10, 11] construct an anonymous blacklisting system for the setting of single sign-on systems using Brands' private credential scheme [8, 9]. Because Brands' credentials are not rerandomizable (and thus different showings of the same credential are linkable), the system calls for U to obtain a set of credentials upon registration; each credential in the set can then be used for one authentication. Each SP maintains a blacklist of credentials of revoked users. To prove that she is not on the blacklist, U computes a zero-knowledge proof of non-membership on the blacklist and sends it to the SP. The authors utilize a novel proof technique that enables both U and the SP to do this using a number of exponentiations that scales with the square root of the list size (as opposed to linearly, as in other schemes). They accomplish this with a modified form of the batch verification techniques of Bellare *et al.* [4].

Tsang *et al.* [83–85] (in fact, many of the same authors as Nymble) proposed Blacklistable Anonymous Credentials (BLAC) in the following year. BLAC removes the trust assumptions from Nymble by eliminating the role of the NM entirely. Similar to the work of Brands *et al.* [10, 11], authentication with an SP in BLAC requires U to prove that her credential is not present on a blacklist of revoked credentials. Unfortunately, BLAC is impractical for most real-world applications because the number of modular exponentiations in the non-membership proof scales linearly in the size of the blacklist. (For each blacklist entry, the proof takes about 1.8 ms for U to prepare and 1.6 ms for the SP to verify [83].) If the blacklist grows to just one thousand users, then several hundred kilobytes of communication and several seconds of computation are required (per access) to prove that U is not on the blacklist [89]. For 7000 users—which we argue on page 63 is a reasonble upper bound on blacklist size—this is over 1.3 megabytes download for the blacklist alone, and ≈ 24 seconds of computation to prepare and verify the proof. For large SPs with many users (such as Wikipedia), the performance of this approach is unacceptable. Note that Nymbler's inter-window revocation mechanism employs a similar proof of non-membership on a blacklist. With our approach, this same scenario requires just 224 kilobytes of download for the blacklist, and ≈ 1.5 seconds of computation to prepare and verify the proof.

Concurrently and independently, Brickell and Li [13, 14] proposed Enhanced Privacy ID (EPID). EPID is similar to BLAC, but is specially designed to enable a TPM with an embedded private key to authenticate anonymously, while enabling SPs to revoke access from compromised TPMs. The non-membership proof in EPID is slightly faster than the one in BLAC, but the scheme requires U to have specialized hardware and the cost is still prohibitively expensive since the number of modular exponentiations also scales linearly in the size of the blacklist.

Privacy-Enhanced Revocation with Efficient Authentication (PEREA) [86] is a second revocable anonymous credential system proposed by the same authors as BLAC. It uses a cryptographic accumulator to reduce the linear-time non-membership proof to a constant-time non-membership proof (for the SP). To facilitate this, the system uses an **authentication window**, which is similar in concept to that of a linkability window, except that it specifies the *maximum number of subsequent connections* U may make—as opposed to *the maximum duration of time* that can elapse—before it becomes impossible to revoke her for her prior behaviour. However, while the accumulator approach makes the cost of verification at the SP constant, it is still several orders of magnitude slower (about 160 ms per authentication [86]) than authentication in the Nymble-like systems. Moreover, U must still perform linear work to compute non-membership witnesses (about 7 ms per entry on the blacklist, if the authentication window is of size 30 [86]).

3.2 (Restricted) Blind Signature Schemes

In his seminal work, Chaum [22] proposed the idea of **blind signatures**, wherein a user obtains a cryptographic signature on a message without revealing *any* nontrivial information about the message to the signer. In the following year, Chaum [23] gave the first construction of blind signatures based on RSA signatures. One particularly appealing property of Chaum's RSA-based blind signatures is that, by choosing the public RSA exponent to be 3, the cost of the verification equation is essentially just that of computing two modular multiplications.

Later, Brands ([7] and [9, Chap. 4]) proposed **restricted blind signatures**, wherein 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 signer can *restrict* the set of messages that it is willing to sign. If the user later modifies the structure of the message in any way, then the signature will become invalid. However, unlike Chaum's RSA-based blind signatures, verifying Brands' restricted blind signatures has a computational cost that is dominated by a multi-exponentiation, where each exponent is essentially random (modulo a large prime) and depends on the message to be signed. In particular, the signer cannot select parameters to ensure that these exponents will be small values (such as 3).

Camenisch and Lysyanskaya [16–18] propose a versatile restricted blind signature scheme (often called the CL-signature scheme) that allows restricted blind signatures to be *rerandomized*. One full-length exponentiation and one multi-exponentiation (with each exponent approximately equal in size to the message being signed) dominate the cost of verifying a CL-signature. The well-known CL-credential [5, 16, 17] scheme is constructed from CL-signatures.

Recently, Groth and Sahai [47] proposed a zero-knowledge proof system based on bilinear pairings. Belenkiy *et al.* [3] constructed restricted blind signatures—which they called P-signatures—and a noninteractive anonymous credential system from the Groth-Sahai framework.

The cost of signature verification in their scheme is about one elliptic curve exponentiation and three pairing operations.

In this work, we propose **Verifier-Efficient Restricted Blind Signatures** (VERBS), a restricted blind signature with an efficient verification protocol. Similar to Chaum's blind signatures, we base our approach on RSA signatures; however, we utilize zero-knowledge proofs in the blinding and signing protocols to 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 the extremely low cost of the verification protocol (i.e., verification in our scheme is almost as efficient as Chaum's *non-restricted* blind signatures with an exponent of 3). In particular, the cost of verifying a signature is dominated by three modular multiplications; this yields a verification protocol that is *1–2 orders of magnitude* faster than any previously proposed restricted blind signature scheme for realistic parameter sizes. We use VERBS to help the SP distinguish validly formed nymbles from random values; thus, the low cost of VERBS verificaton is necessary to ensure that our scheme satisfies the verifier-efficiency property.

Chapter 4

Nymbler

This chapter presents Nymbler, our approach to anonymous blacklisting. It first provides an overview of the architecture and the protocols involved. Then, following a discussion of the system parameters, it gives a detailed exposition of the construction and each of Nymbler's protocols. Implementation details and performance measurements are presented in Chapter 5.

4.1 Architectural Overview

Nymbler utilizes the general Nymble framework as presented in §1.2. The system therefore includes three different third parties to facilitate its operation. The Verinym Issuer (VI) is a distributed entity that is responsible for issuing a user (U) with a verinym; U then uses her verinym to obtain a set of nymbles for some Service Provider (SP) from the Nymble Issuer (NI). U can later authenticate with the SP using one of these nymbles; if U misbehaves, the SP transmits her nymble to the Pseudonym Extractor (PE) to have her blacklisted. Please refer back to §1.2 for a more comprehensive overview of how the framework operates.

Eight protocols comprise Nymbler; we mention each protocol here and briefly discuss: 1) what the protocol does, 2) who is involved in the protocol, and 3) when the protocol is executed.

The **Verinym Acquisition Protocol** allows U to obtain a verinym from the VIs. If U is a Tor exit relay or a user whose access to the VIs is blocked by a third party, then a modified version of the Verinym Acquisition Protocol is used. (See §4.3.3 for details.) U uses the **Verinym Showing Protocol** to convince the NI that she has a valid verinym without revealing any sensitive information. If U wishes to connect to an SP, she uses the **Nymble Acquisition Protocol** to obtain a set of nymbles from the NI for that SP. This protocol requires U to use the Verinym Showing Protocol to prove that she has a valid verinym, as well as the **Non-membership Proof Protocol** to prove that she is not subject to an inter-window revocation. She can then connect

Table 4.1: Trust assumptions required by U and the SP to guarantee Nymbler's security properties.

Security Property	Who	Whom
Misauthentication resistance	SP	———
Backward anonymity	U	VI
Unlinkability	U	PE
Revocability	SP	VI, NI, PE
Revocation auditability	U	PE
Non-frameability	U	VI

This table lists Nymbler's security properties, and specifies *who* must trust *whom* to be honest (but curious) for these properties to hold. When the VI must be trusted, this means that at most $t - 1$ VIs can be corrupt.

to the SP (or PE) to execute the **Revocation Audit Protocol**, which informs her if she appears on the current blacklist or revocation queue. Finally, she invokes the **Nymble Showing Protocol** with the SP to authenticate herself. If she misbehaves, the SP connects to the PE to invoke the **Revocation Protocol**. This causes the PE to use the **Pseudonym Extraction Protocol** to evaluate the trapdoor function.

4.1.1 Trust and Threat Model

Nymbler's threat model allows for any subset of users or SPs to be compromised and hence under adversarial control. The security and privacy properties of the system therefore make no assumptions about the honesty of its users, and require that an SP is honest only to guarantee the availability of that SP's own services. As noted by Tsang *et al.* [89, Page 4], "not trusting these entities is important because encountering a corrupt server and/or user is a realistic threat." However, as with Nymble, our approach does require certain trust assumptions regarding the VIs, NI and PE. Table 4.1 summarizes these trust assumptions and describes which security properties rely on each assumption. Note that Nymbler requires a dramatically reduced level of trust as compared to Nymble (cf. [89, Fig. 3]).

4.1.2 Parameters and Notation

Our Nymbler construction uses several system parameters. This section briefly introduces these parameters and our notation; we divide it into sections corresponding to each entity in the system. In some instances in this section, we may appear to be placing arbitrary conditions on certain values; we provide the rationales for these conditions in the appropriate part of the text. Our goal here is simply to concisely list each of the symbols we use for ease of reference. We refer the reader to Figure 4.1 on page 36 for an illustration of the relationship between linkability windows, VVPs and time periods.

U $hash(\cdot)$ is a strong cryptographic hash function; ℓ_{hash} is the bit length of its output. (We suggest SHA256 as a reasonable choice for $hash$.) U's verinym x_0 is computed from her IP address z; computations involving z are usually performed on a hash $y = hash(z)$.

VI There are s VIs and U must obtain a verinym share from at least $t \leq s$ of them to construct her verinym. There are L_{max} linkability windows for a given public key (of the VIs); the current linkability window is L_{cur}. Each linkability window contains K_{max} VVPs; the current VVP is K_{cur}. A verinym is valid for at most K_{lim} VVPs. Using a distributed protocol, the VIs generate a public RSA modulus $n = p_n q_n$, such that $N = 4n + 1$ is prime. Provided an honest majority exists among the VIs, the factorization of n is unknown to anybody. The VIs choose a master public key $E = \eta^{K_{\text{max}}-1} \cdot \prod_{L=0}^{L_{\text{max}}-1} e_L$ such that $\eta > s$ and $e_L > s$ (for $0 \leq L < L_{\text{max}}$) are distinct primes and $E < \varphi(n)$; the tuple (n, e_L) is then the VI's public key for linkability window L. Each VI$_i$ has share s_i of the master private key $D = E^{-1} \bmod \varphi(n)$ and a verification value v_i; a VI can easily convert its share into a share of the inverse of any public key $(\eta^K e_L)^{-1} \bmod \varphi(n)$ for $0 \leq L < L_{\text{max}}$ and $0 \leq K < K_{\text{max}}$. If VI$_i$ computes a share of U's verinym, we denote this share by X_i; when t shares are recombined, the result is $x_{K_{\text{exp}}}$ (where K_{exp} is the expiration VVP). U computes her verinym x_0 from $x_{K_{\text{exp}}}$ via $x_0 = x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}}}}$. Both α_n and β_n are generators of the order-n subgroup modulo N with $\log_{\alpha_n}(\beta_n)$ unknown. The details of these computations will be given in §4.3.

NI There are T_{max} time periods per linkability window (where K_{max} divides T_{max}). The NI has a public RSA modulus $m = p_m q_m$, such that $M = 2m + 1$ is prime, and private key $d_m = 3^{-1} \bmod \varphi(m)$; these keys are used for VERBS. Each SP is associated with a canonical name '`name`' and a group element $h = hash(k \| L_{\text{cur}} \| \texttt{name})$, where k is the smallest 4-byte value that makes h a quadratic residue modulo n. U requests nymbles for at most the next $J \leq T_{\text{lim}}$ time periods. For time period T, U's nymble seed is $h_T = h^{x2^T} \bmod n$ and her nymble is $\nu_T = g^{h_T} \bmod \zeta$, where g is a known generator of the order-ζ subgroup modulo Z (see below). The NI also provides an ℓ_ρ-bit prime ρ, an $(\ell_P - \ell_\rho - 1)$-bit prime q_ρ for which $P = 2\rho q_\rho + 1$

is prime, and a random ℓ_{hash}-bit integer ξ. (Here ℓ_P is the bit length of a prime modulo which computing discrete logarithms is infeasible.)

PE The PE has a public RSA modulus $\zeta = p_\zeta q_\zeta$, such that $Z = 4\zeta + 1$ is prime, $p_\zeta - 1$ and $q_\zeta - 1$ are both products of primes in the range $[{}^{B}/_{2}, B]$ (where $B \approx 2^{\ell_B}$), and $\varphi(\zeta) > n$. The PE's private key is the factorization of $p_\zeta - 1$ and $q_\zeta - 1$ (into ℓ_B-bit primes).

SP Each SP has a blacklist $\mathcal{B}_{\text{SP}}$ and revocation queue $\mathcal{R}_{\text{SP}}$ whose entries are reduced modulo ρ. Computation with these values is done in the order-ρ subgroup modulo P; α_ρ and β_ρ are generators of this group with $\delta = \log_{\alpha_\rho}(\beta_\rho)$ known only to the NI. Each SP also has one linking list $\mathcal{L}_{(\text{SP},L_{\text{cur}},T)}$ for each $0 \leq T < T_{\text{max}}$.

4.2 Cryptographic Preliminaries

This chapter makes use of ZKPs about values in **commitments**. Commitments can be **discrete log commitments** (the commitment to x is $C_x = \alpha^x$ for a known group element α) or **Pedersen commitments** [66] (the commitment to x is $C_x = \alpha^x \beta^\gamma$ for known group elements α, β where $\log_\alpha(\beta)$ is unknown, and γ is random). In both cases, the commitment is said to *hide* the value x; a commitment can be *opened* by revealing x (and γ).

We use several standard ZKPs from the literature; in particular, we use the standard proof of knowledge of a committed value [71], proof that a commitment opens to a product of committed values [19], and proof that a committed value is in a particular range [6]. No proof is necessary for addition or scalar multiplication of committed values, as multiplication or exponentiation of the commitments, respectively, can easily compute these.

We also use a proof of nested commitments (a "nest proof"); that is, given A, B, to prove knowledge of 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 discrete log case; the Pedersen case is similar), to prove knowledge of x and G such that $G = g^x$, $A = \alpha^G$, and $B = \beta^x$. (All operations are in appropriate groups, and g, α, β are generators of those groups.) This proof works the same way as the ordinary proof of equality of discrete logarithms: the prover chooses r and outputs g^r and β^r; the verifier (or a hash function if the Fiat-Shamir [39] method is used) chooses a challenge c; the prover outputs $v = r - cx$; the verifier accepts if $G^c g^v = g^r$ and $B^c \beta^v = \beta^r$. The twist in our scenario is that G is not available to the verifier; only its commitment ($A = \alpha^G$) is. We solve this problem by having the prover output α^{g^r} instead of g^r, and having the *prover* compute $A = \alpha^{G^c}$ (the commitment to G^c) and provide a ZKP that this was done correctly (see below). Then the verifier checks that $(\alpha^{G^c})^{g^v} = \alpha^{g^r}$ (along with the unchanged $B^c \beta^v = \beta^r$). In the event that g

and β have different orders (which will be true in general, and in our case), the above range proof is also utilized to show that $0 \leq x < ord(t)$.

For the proof of an exponentiation of a committed value, we use a simplified version of the algorithm of Camenisch and Michels [19]. In their 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 produces a ZKP for each step.

We write our ZKPs using Camenisch-Stadler notation [21]; for example,

$$PK\left\{(x, \gamma) \ : \ C_x = \alpha^x \beta^\gamma\right\}$$

denotes a ZKP of knowledge of the opening of the Pedersen commitment C_x, and

$$PK\left\{(x, y, \gamma_1, \gamma_2, \gamma_3) \ : \ \begin{array}{rr} & C_x = \alpha^x \beta^{\gamma_1} \\ \wedge & C_y = \alpha^y \beta^{\gamma_2} \\ \wedge & C_z = \alpha^{xy} \beta^{\gamma_3} \end{array}\right\}$$

denotes a ZKP that C_z opens to the product of the values committed to in C_x and C_y. The tuple of symbols are to remain secret during the proof, while the relations listed after them specify the theorem to prove; any symbol that appears in some relation but not in the aforementioned tuple is public.

4.3 Threshold Verinyms

We now present Nymbler's threshold verinym construction in full detail. We first present the intuition behind how our verinyms are created with a discussion of threshold signatures. We go on to describe precisely how the system initializes the VIs and their public-private key pairs, and how U obtains and uses her verinym.

4.3.1 Threshold Signatures

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 in the underlying threshold signatures: *unforgeability* and *uniqueness*. All secure signature schemes provide unforgeability [45]; uniqueness, on the other hand, is a property that is only possessed by certain signature schemes.

Definition 15 (Unique Signature Scheme [61])**.** 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 further detail.

Definition 16 ((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 [28] 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 in our case, since our nymble construction uses an RSA number with unknown factorization to protect against a malicious PE. 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.* [41]. In our case we will require that the public key n is chosen such that $N = 4n + 1$ is a prime;[1] this can be accomplished by repeatedly executing the protocol of [41] until a suitable n has been found. The prime number theorem [64, 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: $H(z, \xi) = (z^2 + (z \bmod \xi)) \bmod n$ in place of a hash. We choose this function to prevent U from exploiting the homomorphic properties of RSA encryption.

4.3.2 Initializing the VIs

Let $\mathbf{VI} = \{\mathrm{VI}_1, \mathrm{VI}_2, \ldots, \mathrm{VI}_s\}$ be a set of s VIs. The 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 each prime, by using the 'Distributed Computation of N' protocol from [41, §10] with the modification suggested in [28, §5].

[1]The protocol of [41] with the modification suggested in [28, §5] produces an RSA modulus $n = p_n q_n$ such that $\gcd((p_n - 1)/2, s!) = \gcd((q_n - 1)/2, s!) = 1$; thus, it is easy to see that $2n + 1$ is always divisible by 3; this is why we use $N = 4n + 1$.

Choosing a set of public keys. Suppose that there are K_{max} VVPs per linkability window and n will be retired after L_{max} linkability windows. The VIs agree on a prime $\eta > s$ such that $\lceil \log_2(\eta) \rceil + \mathit{hamming\_weight}(\eta)$ is the smallest possible. (If $s < 17$ then $\eta = 17$ is a good choice.) They also agree on a product of L_{max} distinct primes (and $\eta^{K_{\text{max}}-1}$), $E = \eta^{K_{\text{max}}-1} \cdot \prod_{L=0}^{L_{\text{max}}-1} e_L$, such that $e_L > s$ for all $0 \leq L < L_{\text{max}}$ 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 $\lceil \log_2(e_L) \rceil + \mathit{hamming\_weight}(e_L)$ 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 [41, §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 $(n, e_{L^*}, v, \eta, K_{\text{max}})$, where v is a verification value; the private key is $d_{L^*} = D \cdot \prod_{L \neq L^*} e_L = D \cdot E/(e_{L^*} \cdot \eta^{K_{\text{max}}-1})$. Each VI$_i \in$ **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 chapter, 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, 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, $0 \leq K < K_{\text{max}}$, 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 $\eta^{-1} \bmod \varphi(n)$.

4.3.3 Verinym Acquisition Protocol

We model the **Verinym Acquisition Protocol** after the signing protocol from [28, §3]. However, we make two significant changes in our version: 1) the VIs produce a threshold signature on the value $Y = (y^2 + (y \bmod \xi)) \bmod n$ 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 protocol is initiated by U and executed with at least t different VIs. Let z be U's IP address and set $y = \mathit{hash}(z)$. 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 $K_{\text{cur}} + K_{\text{lim}} \leq K_{\text{max}}$. Because U runs the Verinym Acquisition Protocol with at least t different VIs, it is possible for two or more VIs to compute shares of U's verinym that expire in different 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 last VVP for which all shares are valid) before constructing her verinym. The protocol works as follows:

U

1. U chooses a random size-t subset of $\mathbf{VI}$, say $S = \{\text{VI}_{i_1}, \ldots, \text{VI}_{i_t}\} \subseteq_R \mathbf{VI}$.
2. U connects directly to each $\text{VI}_{i_j} \in S$ and requests a verinym.

VI$_{i_j}$

3. VI_{i_j} receives the request for a verinym from U, with IP address z. It notes the current VVP K_{cur} and computes $y = hash(z)$, $Y = (y^2 + (y \bmod \xi)) \bmod n$ and $K_{i_j} = \min\{K_{\text{max}} - 1, K_{\text{cur}} + K_{\text{lim}}\}$.
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 [28, 40, 75]), 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

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
$$c_{i_j} \overset{?}{=} hash(v, \tilde{y}, v_{i_j}, X_{i_j}^2, v^{g_{i_j}(s!)^2} \cdot v_{i_j}^{-c_{i_j} \cdot (E/(e \,\cdot\, \eta^{K_{i_j}}))}, \tilde{y}^{g_{i_j}} \cdot X_{i_j}^{-2c_{i_j}}).$$
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 $1 \leq j \leq t$, if $K_{i_j} > K_{\text{exp}}$, then U replaces X_{i_j} by $X'_{i_j} = X_{i_j}^{\eta^{(K_{i_j} - K_{\text{exp}})}}$; 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

$$\omega = \prod_{\text{VI}_{i_j} \in S} (X'_{i_j})^{2\lambda_{i_j}} = Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}$$

    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:

$$\begin{aligned}
x_{K_{\text{exp}}} &= \omega^a Y^b \\
&= \left(Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}\right)^a Y^b \\
&= \left(Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}\right)^a \left(Y^{e \cdot d}\right)^b \\
&= \left(Y^{4(s!)^5 \cdot d/(\eta^{K_{\text{exp}}})}\right)^a \left(Y^{(e \cdot \eta^{K_{\text{exp}}}) \cdot d/(\eta^{K_{\text{exp}}})}\right)^b \\
&= \left(Y^{a \cdot 4(s!)^5 + b \cdot e \cdot \eta^{K_{\text{exp}}}}\right)^{d/(\eta^{K_{\text{exp}}})} \\
&= (Y)^{d/(\eta^{K_{\text{exp}}})} \\
&= (y^2 + (y \bmod \xi))^{d/(\eta^{K_{\text{exp}}})} \bmod n.
\end{aligned}$$

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* [35]. Thus, the VI 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 relay operator who wishes to connect to an SP using Nymbler for anonymous authentication. We outline the additional proof required from E below:

E

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

VI$_{i_j}$

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

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$_{i_j}$.

VI$_{i_j}$

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.

Verinym acquisition for censored users

Access to Tor is restricted in several countries due to government censorship (for example, the 'Great Firewall of China' [53]). To solve this problem, the Tor network uses *bridges* [33]. A bridge relay 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 bridges, 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 even a government-level 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 Nymbler 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 list of bridge relays and the list of IVs equivalent. Alternatively, the list of bridge relays and IVs could be linked in some other way so as to make the task of obtaining large portions of the two lists equivalent. However, we leave further development of these considerations to future work.

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

U

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.

VI$_{i_j}$

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

U

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

IV

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

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

VI_{i_j}

6. VI_{i_j} receives ψ_z and checks the signature. If ψ_z 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. Provided at least one of the IVs chosen by the VIs is honest, the attack will fail.

4.3.4 Verinym Showing Protocol

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 that)

$$\big(y^2 + (y \bmod \xi)\big) \equiv \left(x_{K_{\text{exp}}}^{\eta^{K_{\text{exp}}}}\right)^e \bmod n.$$

Figure 4.1 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 must prove that her verinym is valid for the VVP that contains the time period associated with her J^{th} nymble, i.e., for VVP $K^* = \min\{\lfloor (K_{\max} \cdot (T_{\text{cur}} + J))/T_{\max} \rfloor, K_{\text{exp}}\}$. 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.

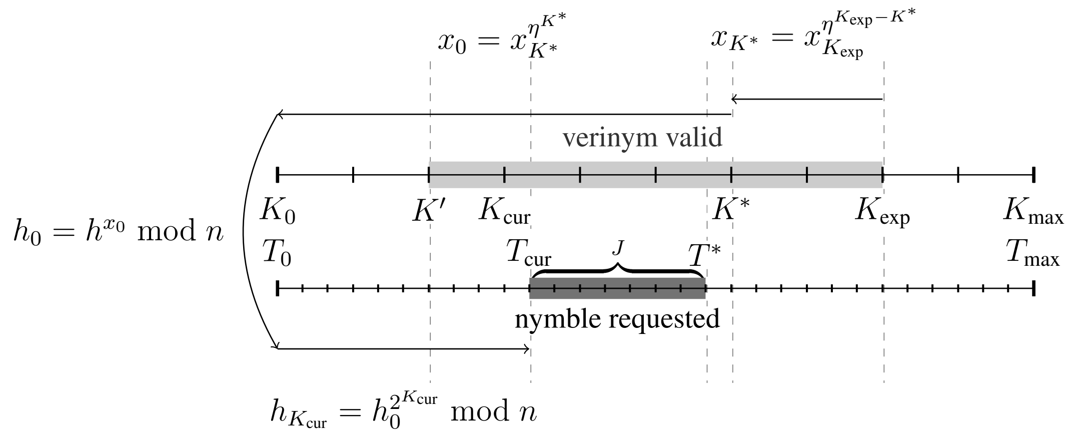

Figure 4.1: **Verinym showing procedure:** This figure outlines our threshold verinym construction. The upper timeline displays VVPs while the lower displays time periods. The **light grey** bar is the range of VVPs for which the verinym is valid. The **dark grey** bar is the range of time periods for which U is requesting nymbles.

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

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 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

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

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

NI

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

4.4 Nymble Construction

The nymble construction procedure in Nymbler is quite similar to the original Nymble. The primary difference with our approach is that we replace the two HMACs and one symmetric-key encryption from the original construction with different functions. We do this in order to enable U to compute her own Nymbles and efficiently prove in zero-knowledge that she has done so correctly. Proving knowledge of the preimage of an HMAC or of the plaintext associated with a symmetric-key ciphertext are both prohibitively expensive operations; further, U can only compute these values if she knows the associated key. Clearly, if the key used for symmetric-key encryption is made public, this enables any SP to deanonymize all users at will.

The first function we employ is Rabin's function: $f(h) = h^2 \bmod n$. No single party knows the factorization of n; thus, inverting f is equivalent to factoring n [64, Chapter 3]. We replace symmetric-key encryption by exponentiation in a trapdoor discrete logarithm group (see §4.5.3). This allows the PE to extract U's SP-specific pseudonym from one of her nymbles with a (tuneably) moderate amount of effort. We replace the final HMAC (used for authentication) with our very own **verifier-efficient restricted blind signatures** (VERBS). Figure 4.2 illustrates Nymbler's nymble construction (compare with Figure 1.1 in §1.2, page 5).

We now briefly introduce our VERBS protocols and then give full details of the Nymble Acquisition Protocol and the Nymble Showing Protocol. We also briefly mention how contracts can be encoded in our nymble construction to enable contract-based revocation.

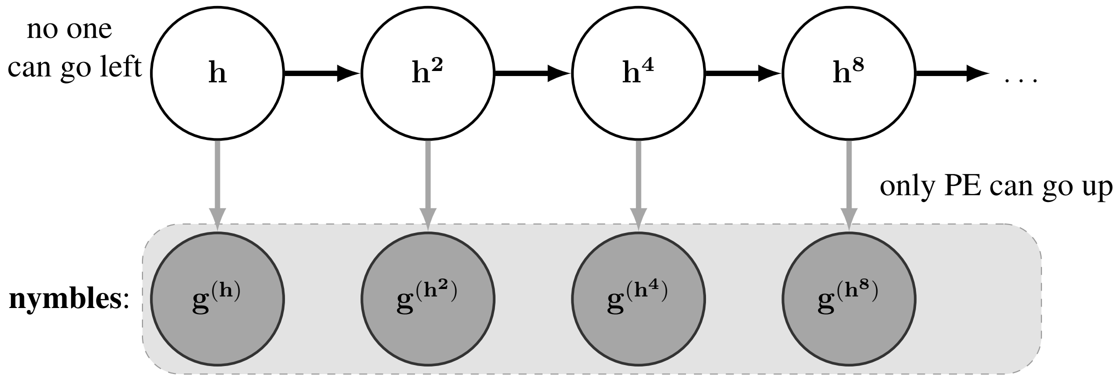

Figure 4.2: **Nymble construction in Nymbler:** In Nymbler, **black** arrows are implemented with Rabin's function ($f(z) = z^2 \bmod n$) and **grey** arrows are implemented with exponentiation in a trapdoor discrete log group ($f'(z) = g^z \bmod \zeta$). The NI issues a VERBS on each nymble (not illustrated) to provide authentication for the SP.

4.4.1 VERBS

We now briefly describe each of the protocols in VERBS. The full algorithms are located in Appendix B. We state each algorithm in its noninteractive zero-knowledge form (such as by using the Fiat-Shamir method [39]); 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 $m = p_m q_m$ whose factorization is known only to the signer; in our case, the signer is the NI. The **VERBS-Blind** algorithm is executed by U. The algorithm takes as input a group element g, a commitment C_x (either a discrete log or Pedersen commitment) to a secret value x, the secret value x itself (plus γ in the case of a Pedersen commitment), and a *context element* ξ that encodes meta-information about the signature. The role of this algorithm, much like its Chaumian counterpart, is to produce the blinded message $S_x = H(\nu, \xi) \cdot \theta^3 \bmod m$, where $\nu = g^x$ and the random blinding factor θ are hidden from the signer, ξ is an ℓ_{hash}-bit number, and $H(z, \xi) = (z^2 + (z \bmod \xi)) \bmod m$ is a one-way function. (It is one-way since the factorization of the modulus m is unknown to everyone but the signer.) In practice, ξ is a hash output and information about the context in which the signature should be accepted as valid are encoded in the preimage of the hash. The algorithm also outputs a ZKP Π_x that convinces the signer that S_x was computed correctly.

The **VERBS-Sign** protocol is run by the NI. It takes the tuple (S_x, p_m, q_m, Π_x) as input: $S_x \in \mathbb{Z}_m^*$ is a blinding of the message x; p_m and q_m are the factors of m; and Π_x is a ZKP that S_x was correctly formed using the x committed to in C_x. It outputs the blinded signature $\sigma' = (S_x)^{\frac{1}{3}} \bmod m$ if all proofs in Π_x are valid; otherwise, it outputs $\perp$. Note that σ' is essentially just a Chaum blind signature.

The **VERBS-Unblind** protocol is run by U. The algorithm takes the tuple (σ', θ) as input; σ' is a blinded signature and θ is the blinding factor used to blind that signature. It outputs $\sigma = \sigma' \cdot \theta^{-1} \bmod m$, which is the unblinded signature.

VERBS-Verify is run by the SP (and U). It takes the tuple (ν, σ, ξ) as input; ν is the message that was signed, σ is the (unblinded) signature, and ξ is the context element. It outputs `true` iff $\sigma^3 \bmod m \overset{?}{=} (\nu^2 + (\nu \bmod \xi)) \bmod m$. Note that the cost of VERBS-Verify is essentially just three modular multiplications.[2]

4.4.2 Nymble Acquisition Protocol

Let x_0 be U's verinym and suppose x_0 expires after VVP K_{exp}. For the remainder of this chapter, we will consider the case when U wishes to connect to an SP with the canonical name '`name`' during current linkability window L_{cur}. Let T_{exp} be the last time period in K_{exp} and let T_{lim} be the number of consecutive time periods, starting from T_{cur}, for which a user may request nymbles. U's first step in communicating with the SP is to invoke the **Nymble Acquisition Protocol** to obtain nymbles from the NI.

This protocol begins with U computing her sequence of nymble seed values. The first nymble seed is computed via exponentiation modulo n, and all subsequent values are computed via repeated squaring modulo n. A publicly known generator of the order-ζ subgroup modulo Z is then raised to the power of each nymble seed to obtain U's nymbles. Finally, the nymbles are blinded using the VERBS-Blind protocol and sent (along with ZKPs) to the NI for signing. The protocol works as follows:

U

1. U computes $h = hash(k \| L_{\text{cur}} \| \texttt{name})$, where k is the smallest 4-byte value that makes h a quadratic residue modulo n, and $T^* = \min\{T_{\text{cur}} + T_{\text{lim}}, T_{\text{exp}}\}$; she sends $(\texttt{name}, T^*)$ to the NI. This indicates to the NI that U is requesting nymbles beginning at time periods T_{cur} through T^*. (This is a total of $J = T^* - T_{\text{cur}} + 1$ nymbles.)
2. U verifiably computes her first *nymble seed* $h_{\text{cur}} = h^{x_0 2^{T_{\text{cur}}}}$ as follows:
   (a) U computes the Pedersen commitments (to x_0 and h_{cur}) $\overline{x_0} = \alpha_n^{x_0} \beta_n^{\gamma_1} \bmod N$ and

[2]We assume here that ℓ_{hash} is much smaller than n (say, 256-bit versus 1536-bit), and so the cost of $\nu \bmod \xi$ is insignificant compared to the cost of the multiplications.

$Y_{\text{cur}} = \alpha_n^{h^{x_0 2^{T_{\text{cur}}}}} \beta_n^{\gamma_2} \bmod N$, and the ZKP:

$$\Pi_0 = PK \left\{ (x_0, \gamma_1, \gamma_2) \; : \; \begin{array}{ll} \overline{x_0} = \alpha_n^{x_0} \beta_n^{\gamma_1} & \bmod N \\ \wedge \;\; Y_{\text{cur}} = \alpha_n^{h^{x_0 2^{T_{\text{cur}}}}} \beta_n^{\gamma_2} & \bmod N \\ \wedge \;\; 0 \le x_0 < n & \end{array} \right\}.$$

3. U computes her remaining nymble seeds—using h_{cur} as a starting point—as follows:
   (a) U computes $(h_{\text{cur}+i})_{i=2}^{T^*-1}$, where $h_{\text{cur}+i} = \left(h_{\text{cur}+(i-1)}\right)^2 = {h_{\text{cur}}}^{2^i} \bmod n$;
   (b) U computes $J-1$ Pedersen commitments $Y_{\text{cur}+i} = \alpha_n^{h_{\text{cur}+i}} \beta_n^{\gamma_i} \bmod N$ ($\gamma_i \in_R \mathbb{Z}_n^*$) and the ZKP

$$\Pi_2 = PK \left\{ \left(\begin{array}{c} h_{\text{cur}+i-1}, \gamma_{i-1}, \\ h_{\text{cur}+i}, \gamma_i \\ \text{for } 1 \le i \le J \end{array} \right) \; : \; \begin{array}{lll} & Y_{\text{cur}+i-1} = \alpha_n^{h_{\text{cur}+i-1}} \beta_n^{\gamma_{\text{cur}+i-1}} & \bmod N \\ \wedge & h_{\text{cur}+i} = (h_{\text{cur}+i-1})^2 & \bmod n \\ \wedge & Y_{\text{cur}+i} = \alpha_n^{h_{\text{cur}+i}} \beta_n^{\gamma_{\text{cur}+i}} & \bmod N \\ & \text{for } 1 \le i \le J & \end{array} \right\}.$$

4. U computes her nymbles $\nu_{\text{cur}+i} = g^{h_{\text{cur}+i}} \bmod \zeta$ for $0 \le i \le J$. (Here, the exponent is just taken as an integer in $[1, n)$.)
5. U computes $\xi^{\text{SP}}_{L_{\text{cur}},(T_{\text{cur}}+i)} = hash(L_{\text{cur}} \| (T_{\text{cur}}+i) \| \text{SP})$ and invokes VERBS-Blind to compute $(\gamma_i', S_i, \Pi_i) = \text{VERBS-Blind}(g, Y_{\text{cur}+i}, \nu_{\text{cur}+i}, \gamma_i', \xi^{\text{SP}}_{L_{\text{cur}},(T_{\text{cur}}+i)})$ for $0 \le i \le J$.
6. U sends all commitments, ZKPs and VERBS-Blind outputs to the NI. U additionally invokes the Verinym Showing Protocol to prove that the value committed to in $\overline{x_0}$ is a valid verinym for K_{exp}, and performs the Non-membership Proof Protocol (see Section 4.5.7) as necessary.

NI

7. The NI receives the commitments, proofs and blinded nymbles from U. It verifies that $J \le T_{\text{lim}}$ and $T_{\text{cur}} + J \le T_{\text{max}}$. The NI also computes (or looks up) $h = hash(k \| L_{\text{cur}} \| \texttt{name})$, which will be needed to verify the ZKPs.
8. The NI verifies each of the ZKPs sent by U. If any proof fails, the NI aborts.
9. The NI computes (or looks up) $\xi^{\text{SP}}_{L_{\text{cur}},(T_{\text{cur}}+i)} = hash(L_{\text{cur}} \| (T_{\text{cur}} + i) \| \text{SP})$ for $0 \le i \le J$. It invokes VERBS-Sign J times to issue blind VERBS on each of U's blinded nymbles. If any call to VERBS-Sign fails due to an incorrect ZKP, the NI aborts.
10. The NI transmits each blinded VERBS to U.

U

11. U receives the blinded VERBS from the NI. She invokes $\textsf{VERBS-Unblind}$ on each blinded VERBS $\sigma'_{\text{cur}+i}$ to obtain an unblinded VERBS $\sigma_{\text{cur}+i}$ on $\nu_{\text{cur}+i}$.
12. U computes and records her SP-specific pseudonym $h_{T_{\text{max}}} = g^{(h^{x_0 2^{T_{\text{max}}}} \bmod n)} \bmod \zeta$.

If U wishes to authenticate with the SP during time period T^* she sends the tuple $(\nu_{T^*}, \sigma_{T^*})$. U can first verify the validity of her nymble by checking if $\textsf{VERBS-Verify}(\nu_{T^*}, \sigma_{T^*}, \xi^{\text{SP}}_{L_{\text{cur}},T^*})$ returns `true`. (Note that U is susceptible to tagging attacks if she does not do this.)

Contract-based Nymble Acquisition

Lin and Hopper [57] describe an objective blacklisting extension for Jack. Their approach uses the *label* field in Camenisch and Shoup's [20] verifiable encryption scheme to force the PE to include a contract in its trapdoor computation. The idea is that if the provided contract is incorrect, then the trapdoor computation will fail. A similar mechanism can easily be incorporated into Nymbler's nymble construction as follows: replace Rabin's function $f(z) = z^2 \bmod n$ with $f'(z, c) = c'z^2 \bmod n$ where c' is a hash of the contract 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 with Lin and Hopper's approach, our approach does not leak information to the PE or SP about whether the given contract is actually enforced on U. This means that, for example, with our approach, different users may have different rights in their contracts, without partitioning the anonymity set. In this case, the SP would just complain about U, even though she may have a "special" contract that allows her to behave "badly", but the resulting nymbles would not actually result in her being blocked.

4.4.3 Nymble Showing Protocol

The **Nymble Showing Protocol** is quite simple; U sends her nymble to the SP, who then confirms that it is valid for, and does not appear on the linking list of, the current time period T_{cur} and linkability window L_{cur}. U initiates this protocol with the SP when she wishes to authenticate. It works as follows:

U

1. U invokes the Revocation Audit Protocol (see Section 4.5.5) for the SP to ensure that she is not presently on the blacklist or on the revocation queue. If U is on one of these lists, she aborts immediately (and does not attempt to authenticate until the next linkability window).
2. U sends $(\nu_{\text{cur}}, \sigma_{\text{cur}})$ to the SP.

SP

3. The SP receives $(\nu_{\text{cur}}, \sigma_{\text{cur}})$ from U. It verifies that $\mathsf{VERBS\text{-}Verify}(\nu_{\text{cur}}, \sigma_{\text{cur}}, \xi^{\text{SP}}_{L_{\text{cur}},T_{\text{cur}}})$ is `true`. If not, the SP aborts.
4. The SP checks if ν_{cur} appears on its linking list $\mathcal{L}_{(\text{SP},L_{\text{cur}},T_{\text{cur}})}$ for the current time period. If so, the SP aborts.
5. The SP grants U access.

4.5 Revocation Mechanisms

We now present Nymbler's revocation mechanisms. We first describe the format of the blacklist, linking lists and revocation queue before describing the Revocation Audit Protocol and the Revocation Protocol. We then introduce trapdoor discrete logarithm groups and outline the Pseudonym Extraction Protocol. Finally, we present the Non-membership Proof Protocol that U executes with the NI during the Nymble Acquisition Protocol.

4.5.1 Revocation Lists

There are a total of $T_{\text{max}} + 2$ different revocation lists per SP and linkability window L_{cur}: the blacklist $\mathcal{B}_{\text{SP}}$, the revocation queue $\mathcal{R}_{\text{SP}}$, and T_{max} linking lists $\mathcal{L}_{(\text{SP},L_{\text{cur}},T)}$ for $0 \le T < T_{\text{max}}$.

U must download $\mathcal{B}_{\text{SP}}$ and $\mathcal{R}_{\text{SP}}$ each time she authenticates; naively implemented, a blacklist of Λ nymbles is about $(1536 \cdot \Lambda)$-bits of download (assuming $\ell_\zeta = 1536$, as we suggest) versus only $(256 \cdot \Lambda)$-bits in the original Nymble, a factor of six increase. We solve this by reducing each entry on $\mathcal{B}_{\text{SP}}$ modulo an ℓ_ρ-bit prime. We reduce nymbles modulo a prime instead of using, say, a cryptographic hash function, because this allows us to later prove statements about the reduced nymbles. 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 $\mathcal{B}_{\text{SP}}$ are reduced modulo ρ. The Non-membership Proof Protocol in §4.5.7 constructs ZKPs in the order-ρ subgroup modulo P. We suggest $\ell_\rho = 256$ (indeed, $\rho = 2^{256} - 183$ is a good choice) and $\ell_P = 1536$ as reasonable values for these parameters.

$\mathcal{B}_{\text{SP}}$ and each $\mathcal{L}_{(\text{SP},L_{\text{cur}},T)}$ are implemented as a hash map (which is maintained and distributed by the PE). This enables the SP and U to query these lists in constant amortized time.

Blacklist freshness

In order for U to verify the *integrity* and *freshness* of $\mathcal{B}_{\text{SP}}$, we adopt the idea of a **daisy chain** (or simply a **daisy**), which was originally proposed by Tsang *et al.* [89] in the journal version of their Nymble paper. The idea works as follows: when the PE places a new entry on the blacklist (during time period T^*), it also generates a random **daisy seed** $\delta_{T_{\text{max}}}$. The PE 'grows the daisy' from this seed by computing the hash chain $\delta_{T_{\text{max}}}, \delta_{T_{\text{max}}-1}, \ldots, \delta_{T^*}$. (Each element in the chain is a hash of the preceding element.) It also issues a signature $\sigma_{\mathcal{B}_{\text{SP}}}$ on $hash(\delta_{T^*} \| T^* \| L_{\text{cur}} \| \mathcal{B}_{\text{SP}})$ and sends the tuple $(\mathcal{B}_{\text{SP}}, \delta_{T^*}, \sigma_{\mathcal{B}_{\text{SP}}})$ to the SP. U now confirms that the blacklist is up-to-date (freshness) and that it has not been modified by an unauthorized party (integrity) by downloading $\mathcal{B}_{\text{SP}}$, computing $hash(\delta_{T^*} \| T^* \| L_{\text{cur}} \| \mathcal{B}_{\text{SP}})$, and checking that the signature is valid for this value. If there are no changes to the blacklist for the next $i - 1$ time periods, then on the i^{th} time period the PE sends δ_{T^*+i} to the SP. U now checks the integrity and freshness of $\mathcal{B}_{\text{SP}}$ by computing $\delta = hash^i(\delta_{T^*+i})$ and $hash(\delta \| T^* \| L_{\text{cur}} \| \mathcal{B}_{\text{SP}})$, and then checking that the signature is valid. This strategy avoids the need for the PE to recertify the blacklist at each time period even when no updates have occurred. Instead, the PE need only reveal a single value (the preimage of a hash) to the SP at the start of the time period.

Except for when it is empty, the revocation queue will typically not remain unchanged between two time periods. Therefore, we do not use daisies for the revocation queue. Instead, the PE just publishes a signature on $hash(T_{\text{cur}} \| L_{\text{cur}} \| \text{SP} \| \mathcal{R}_{\text{SP}})$ at the start of each time period.

4.5.2 Revocation Protocol

Suppose that U misbehaves in time period T^* and her misbehaviour is discovered in time period T' of the same linkability window. The SP connects to the PE to invoke the **Revocation Protocol**, which works as follows:

SP

1. The SP connects to the PE and sends $(\nu_{T^*}, \sigma_{T^*})$.

PE

2. The PE receives $(\nu_{T^*}, \sigma_{T^*})$ from the SP. It notes the current time T' and checks that $\textsf{VERBS-Verify}(\nu_{T^*}, \sigma_{T^*}, \xi^{\text{SP}}_{L_{\text{cur}}, T^*})$ is `true`; if not, it aborts.
3. The PE pushes $\nu_{T^*} \bmod \rho$ onto $\mathcal{R}_{\text{SP}}$ and waits for it to reach the front of the queue. Meanwhile, at the end of each time period $T' + i$, the PE computes a signature $\sigma_{\mathcal{R}_{\text{SP}}}$ on $hash(T_{\text{cur}} \| L_{\text{cur}} \| \text{SP} \| \mathcal{R}_{\text{SP}})$.

4. The PE invokes the Pseudonym Extraction Protocol to have $\mathcal{B}_{\text{SP}}$ and $\mathcal{L}_{(\text{SP},L_{\text{cur}},T)}$ updated ($T' < T < T_{\max}$).
5. The PE pops $\nu_{T^*} \bmod \rho$ from the front of $\mathcal{R}_{\text{SP}}$.
6. The PE constructs a daisy for $\mathcal{B}_{\text{SP}}$ as outlined in §4.5.1.
7. The PE sends $(\mathcal{B}_{\text{SP}}, \mathcal{R}_{\text{SP}}, \mathcal{L}_{(\text{SP},L_{\text{cur}},T'+1)}, \ldots, \mathcal{L}_{(\text{SP},L_{\text{cur}},T_{\max})}, \delta_{T_{\text{cur}}}, \sigma_{\mathcal{B}_{\text{SP}}}, \sigma_{\mathcal{R}_{\text{SP}}})$ to the SP.

4.5.3 Trapdoor Discrete Logarithms

The PE's public key is an ℓ_ζ-bit product $\zeta = p_\zeta q_\zeta$ of two B-smooth primes p_ζ and q_ζ (that is, $p_\zeta - 1$ and $q_\zeta - 1$ are products of ℓ_B-bit primes), such that $Z = 4\zeta + 1$ is a prime.[3] It is required that $\zeta > n$, but being just barely larger is sufficient. A different ζ could be used in conjunction with each SP and linkability window, but for brevity we will treat ζ as a fixed value in our discussion.

ℓ_B is chosen so that computing discrete logarithms modulo ζ in subgroups of order $\approx 2^{\ell_B}$ is costly but feasible. In other words, given knowledge of the factorization of $p_\zeta - 1$ and $q_\zeta - 1$, computing discrete logarithms modulo p_ζ and q_ζ (and hence, modulo ζ) is feasible using a lot of parallelism and an efficient algorithm like van Oorschot and Wiener's parallel ρ algorithm [91]. g is a generator of the group of quadratic residues modulo ζ, and α_ζ and β_ζ are generators of the order-ζ subgroup of $\mathbb{Z}_Z^*$ such that $\log_{\alpha_\zeta}(\beta_\zeta)$ is unknown. The PE's private key is (p_ζ, q_ζ) and the factorization of $\varphi(\zeta)$ (into ℓ_B-bit primes). Thus, the PE can compute discrete logarithms modulo ζ using its private key, but for everyone else this is infeasible.

Proving statements about the cost of the trapdoor. We mentioned in §1.2 that the PE should be able to prove in zero-knowledge that random instances of the trapdoor problem require a certain amount, t_{dl}, of wall-clock time to solve (on average) with the trapdoor. Using a straightforward generalization of the zero-knowledge proof that a number is a product of two safe primes, due to Camenisch and Michels [19], the PE can prove in zero-knowledge that ζ is B-smooth. Given a number of cores available and the speed of each of these cores, we can easily come up with an estimate for how long a trapdoor discrete logarithm computation will take (see §5.3.1); this computation depends on the bit length of the public modulus, ℓ_ζ, and the bit length of the prime factors of $\varphi(\zeta)$. In other words, given some assumptions about the computational power of the PE, the PE can prove in zero-knowledge that the average time t_{dl} needed to solve a trapdoor discrete logarithm has a particular value. While the validity of this proof relies on assumptions about the computational capacity of the adversary, it *does* give a reliable estimate of the computational—thus, economic—cost for a malicious PE to link the actions of a large number of users.

[3]We use $Z = 4\zeta + 1$ because it is easy to see that p_ζ and q_ζ must be congruent to $2 \bmod 3$, and so $2\zeta + 1$ must be divisible by 3.

4.5.4 Pseudonym Extraction Protocol

Suppose U misbehaves in time period T^* and a complaint is filed in time period T'. When U's nymble reaches the front of the revocation queue, the PE invokes the **Pseudonym Extraction Protocol** to perform a trapdoor discrete logarithm and add an entry to the SP's blacklist and linking lists. The protocol proceeds as follows:

PE

1. The PE looks up the most recent blacklist and linking lists for the SP. (It already has a local copy.)
2. The PE uses knowledge of the factorization of ζ to recover $h_{T^*} = h^{x_0 2^{T^*}}$ from $(\nu_{T^*} = g^{h_{T^*}} \bmod \zeta, \sigma_{T^*})$.
3. The PE computes $h_{T'} = h^{x_0 2^{T^* - T'}} \bmod n$ by repeated squaring of h_{T^*}.
4. The PE computes $h_{T'+1}, \ldots, h_{T_{\text{max}}}$ by repeated squaring of $h_{T'}$. It uses these to compute $\nu_{T'+1}, \ldots, \nu_{T_{\text{max}}}$ via exponentiation.
5. For $T' < T < T_{\text{max}}$, the PE puts g^{h_T} into $\mathcal{L}_{(\text{SP}, L_{\text{cur}}, T)}$. It also adds $g^{h_{T_{\text{max}}}}$ to $\mathcal{B}_{\text{SP}}$.

4.5.5 Revocation Audit Protocol

U invokes the **Revocation Audit Protocol** with either the SP or the PE in order to find out her revocation status at the SP. As the name implies, this protocol is what brings revocation auditability to Nymbler. It is important for the user to run this protocol each time she wishes to authenticate with an SP; otherwise, a malicious SP could accept nymbles from U *after* she is revoked and thus trace her actions without her knowledge. The protocol works as follows:

U

1. U connects (anonymously) to the SP (or the PE) and requests the latest copy of tuple $(\mathcal{B}_{\text{SP}}, \mathcal{R}_{\text{SP}}, \sigma_{\mathcal{B}_{\text{SP}}}, \sigma_{\mathcal{R}_{\text{SP}}}, \delta_{T_{\text{cur}}}, T^*)$.
2. When U receives this tuple, she then verifies that $\sigma_{\mathcal{R}_{\text{SP}}}$ is a valid signature on

$$hash(T_{\text{cur}} \| L_{\text{cur}} \| \text{SP} \| \mathcal{R}_{\text{SP}})$$

   and that $\sigma_{\mathcal{B}}$ is a valid signature on

$$hash(\delta \| T_{\text{cur}} \| L_{\text{cur}} \| \mathcal{B}_{\text{SP}}),$$

   where $\delta = hash^{(T_{\text{cur}} - T^*)}(\delta_{T_{\text{cur}}})$. If either verification fails, then U aborts.

3. U checks $\mathcal{B}_{\text{SP}}$ to ensure that it does not contain her SP-specific pseudonym $\nu_{T_{\max}}$; if it does contain her SP-specific pseudonym, then she aborts.
4. U checks $\mathcal{R}_{\text{SP}}$ to see if any of her recently used nymbles are listed; if so, she aborts.
5. U concludes that she is not revoked from SP and proceeds with the Nymble Acquisition Protocol as usual.

4.5.6 Supporting Inter-window Revocation

Supporting inter-window revocations requires each SP to maintain a blacklist for each prior linkability window from which some user is still blocked. When U wishes to obtain a set of nymbles for an SP, she first proves that her SP-specific pseudonyms from past linkability windows are not on the associated blacklists. This is reminiscent of the BLAC [83–87] and EPID [13,14] approach to blacklisting, which, as we discussed earlier, and has been discussed elsewhere [50, 88, 89], raises performance and scalability concerns. However, five important distinctions with our approach warrant mention:

1. Since most IP address bans are short term [94], most revoked users will not appear on an inter-window blacklist. This significantly reduces the expected size of the blacklist against which U must generate a proof.

2. U forms a proof only 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 blacklist entries are all reduced modulo ρ, and we can therefore work in the order-ρ subgroup modulo P.

4. Brands *et al.*'s [10, 11] square-root-time non-membership proof allows U to prove non-membership on a blacklist, and the NI to verify this proof, using a number of exponentiations proportional to $\sqrt{\Lambda}$, where Λ is the size of the blacklist against which the proof is generated. For large blacklists, this method dramatically outperforms the linear exponentiations approach used in both BLAC [83–87] and EPID [13, 14].

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. We argue that SPs are likely to discover a majority of misbehaviours—particularly those serious enough to warrant inter-window revocation—during the linkability window in which they occur; thus, long-term blacklists are likely to remain relatively static as compared to the short-term blacklist. It is this largely static nature of long-term blacklists that makes this approach effective.

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

Remark. Observations 3 through 5 may be useful for reducing the computation and communication costs of some other anonymous blacklisting systems in the literature.

4.5.7 Non-membership Proof Protocol

Let $\mathcal{B}_{(\mathrm{SP},L^*)} = \{\nu_1 \bmod \rho, \ldots, \nu_\Lambda \bmod \rho\}$ 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 $|\mathcal{B}_{(\mathrm{SP},L^*)}| = \Lambda$ is a perfect square and let $\lambda = \sqrt{\Lambda}$. Also, let α_ρ, β_ρ be generators of the order-ρ subgroup modulo P; α_ρ and β_ρ are publicly known and $\psi_\rho = \log_{\alpha_\rho}(\beta_\rho) \bmod P$ is known only to the NI. (See §5.3.2.)

Thus, to prove 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 the following technique, due to Brands *et al.* [10, 11], to implement this proof.

Let $j = (i-1)\cdot\lambda$ and define the polynomial

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

for $1 \le i \le \lambda$. Also, let $h_{L^*} = h^{2^{T_{\max}}}$, where $h = hash(k\|L^*\|\mathtt{name})$ and k is the smallest 4-byte value making h a quadratic residue modulo n.

U invokes the **Non-membership Proof Protocol** to prove to the NI that she is not revoked from an SP. This protocol works as follows:

U

1. U computes a Pedersen commitment to her verinym, $\overline{x_0} = \alpha_\rho^{x_0}\beta_\rho^{\gamma_1} \bmod P$ and uses the Verinym Showing Protocol to prove it is valid.
2. U computes her SP-specific pseudonym for L^*, $\nu_{L^*} = (h_{L^*}^{x_0} \bmod N) \bmod \rho$, a commitment $\overline{\nu_{L^*}} = \alpha_\rho^{\nu_{L^*}}\beta_\rho^{\gamma_2} \bmod P$ and a ZKP of correctness

$$\Pi_1 = PK\left\{ (x_0, \gamma_1, \gamma_2) \ : \ \begin{array}{rll} \overline{x_0} & = \alpha_\rho^{x_0}\beta_\rho^{\gamma_1} & \bmod P \\ \wedge\ \overline{\nu_{L^*}} & = \alpha_\rho^{(h_{L^*}^{x_0} \bmod \rho)}\beta_\rho^{\gamma_2} & \bmod P \end{array} \right\}$$

3. U chooses $r_1, \ldots, r_\lambda \in_R \mathbb{Z}_\rho$ and, for $1 \le i \le \lambda$, computes
   (a) $o_i = p_i(\nu_{L^*}) \bmod \rho$, the evaluation of p_i at ν_{L^*};
   (b) $\varsigma_i = a_{i,\lambda}r_\lambda + \ldots + a_{i,1}r_1 \bmod \rho$;

(c) $C_i = \alpha_\rho^{(\nu_{L^*})^i} \beta_\rho^{r_i} \bmod P$; and,

(d) $C_{o_i} = \alpha_\rho^{o_i} \beta_\rho^{\varsigma_i} \bmod P$.

4. U sends each commitment $\overline{\nu_{L^*}}$, C_i and C_{o_i} to the NI, together with Π_1 and the following ZKP:

$$\Pi_2 = PK\left\{\begin{pmatrix} \gamma_2, \\ \nu_{L^*}, \\ r_i, \\ \varsigma_i, \\ o_i \end{pmatrix} : \begin{array}{lll} & \overline{\nu_{L^*}} = \alpha_\rho^{\nu_{L^*}} \beta_\rho^{\gamma_2} & \bmod P \\ \wedge & C_i = \alpha_\rho^{\nu_{L^*}^i} \beta_\rho^{r_i} & \bmod P, \\ \wedge & C_{o_i} = \alpha_\rho^{o_i} \beta_\rho^{\varsigma_i} & \bmod P, \\ \wedge & o_i \not\equiv 0 & \bmod \rho, \\ & \text{for all } 1 \le i \le \lambda & \end{array}\right\}, \tag{4.1}$$

which proves that: 1) the commitments C_i hide consecutive powers of ν_{L^*}; and, 2) the commitments C_{o_i} each hide nonzero values. Note that, combined with Equation 4.2, this proves to the NI that C_{o_i} is a commitment to $p_i(\nu_{L^*})$, and that this evaluation is nonzero.

NI

5. The NI verifies that, for each $1 \le i \le \lambda$,

$$C_{o_i} \overset{?}{\equiv} (C_\lambda)^{a_{i,\lambda}} (C_{\lambda-1})^{a_{i,\lambda-1}} \cdots (C_1)^{a_{i,1}} \alpha_\rho^{a_{i,0}} \bmod P \tag{4.2}$$

If any of these equivalences fails, then the NI aborts.

6. The NI verifies the ZKPs Π_1 and Π_2. If either verification fails, the VI aborts.

Note Step 5 seems to require $\lambda(\lambda + 1) = \Lambda + \lambda$ modular exponentiations; however, collapsing all λ of these verifications into a single batch verification using techniques of Bellare *et al.* [4] reduces this to just $2\lambda + 1$ modular exponentiations. To do this, the NI chooses random $s_1, \ldots, s_\lambda \in_R \{1, \ldots, \kappa_2\}$, where κ_2 is a security parameter, and checks if

$$\prod_{i=1}^{\lambda} C_{o_i}^{s_i} \overset{?}{\equiv} \alpha_\rho^{\sum_{i=1}^{\lambda} a_{i,0} \cdot s_i} \cdot \prod_{i=1}^{\lambda} C_i^{\sum_{j=1}^{\lambda} a_{j,i} \cdot s_j} \bmod P. \tag{4.3}$$

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

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 $1 \le i \le \lambda$, the zeros of $p_i(\tau)$ are exactly those values appearing on the sublist of $\mathcal{B}_{(\text{SP},L^*)}$ defined by $\{\nu_{j+1}, \nu_{j+2}, \cdots, \nu_{j+\lambda}\}$, where $j = (i-1) \cdot \lambda$, and that these sublists cover the entire blacklist. Combined with Π_1, the first line of Π_2 proves that C_i hides the i^{th} power of ν_{L^*}; thus, if Equation 4.2 holds then it follows that C_{o_i} is a commitment to an evaluation of $p_i(\tau)$ at the point ν_{L^*}, since

$$\begin{aligned} C_{o_i} &\equiv (C_\lambda)^{a_{i,\lambda}} (C_{\lambda-1})^{a_{i,\lambda-1}} \cdots (C_1)^{a_{i,1}} \alpha_\rho^{a_{i,0}} \\ &\equiv \alpha_\rho^{(a_{i,\lambda}\nu_{L^*}^\lambda + \cdots + a_{i,1}\nu_{L^*} + a_{i,0})} \beta_\rho^{(a_{i,\lambda} r_\lambda + \cdots + a_{i,1} r_1)} \\ &\equiv \alpha_\rho^{p_i(\nu_{L^*})} \beta_\rho^{\varsigma_i} \bmod P \\ &\equiv \alpha_\rho^{o_i} \beta_\rho^{\varsigma_i} \bmod P. \end{aligned}$$

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

Blacklist transferability

Our approach to inter-window revocation also enables a limited form of blacklist transferability. 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 from 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 registration process.

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 blacklist. In other words, once U proves that her pseudonym ν_{L^*} is not on $\mathcal{B}_{(\text{SP},L^*)}$, 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 with the 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, the use of verfication tokens does leak some (albeit minimal) information to the VI about U's actions. 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 verification tokens are an opt-in feature; each user is free to make her own choice about whether this much information leakage is acceptable to them. Those users who are uncomfortable revealing this additional information about their actions can simply perform the entire non-membership proof during each invocation of the Nymble Acquisition Protocol.

Chapter 5

Implementation

We have implemented each of the key components of Nymbler. This chapter presents some details about this implementation and summarizes the performance measurements that we have obtained from it.

Experimental setup. Our implementations are written in C++ using GMP [42] and NTL [76] to handle multiprecision arithmetic. Except for trapdoor discrete logarithms, all performance measurements are obtained on a 2.83 GHz Intel Core 2 Quad Q9550 running Ubuntu 9.10 64-bit. All code that we run on this machine is single-threaded (our experiments use only a single Q9550 core). Trapdoor discrete logarithms are implemented in CUDA [65], utilizing NVIDIA Tesla GPUs to achieve massively parallel computing. Our trapdoor discrete logarithm experiments are performed on a 2.4 GHz Intel Xeon E5620 with two Tesla M2050s running Ubuntu 10.04 64-bit. This code is highly parallelized.

5.1 Threshold Verinyms

5.1.1 Verinym Acquisition Protocols

Performance measurements. Table 5.1 summarizes performance measurements for the Verinym Acquisition Protocol. We ran both U and the VIs on a single machine using a $(3, 7)$-threshold construction. 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

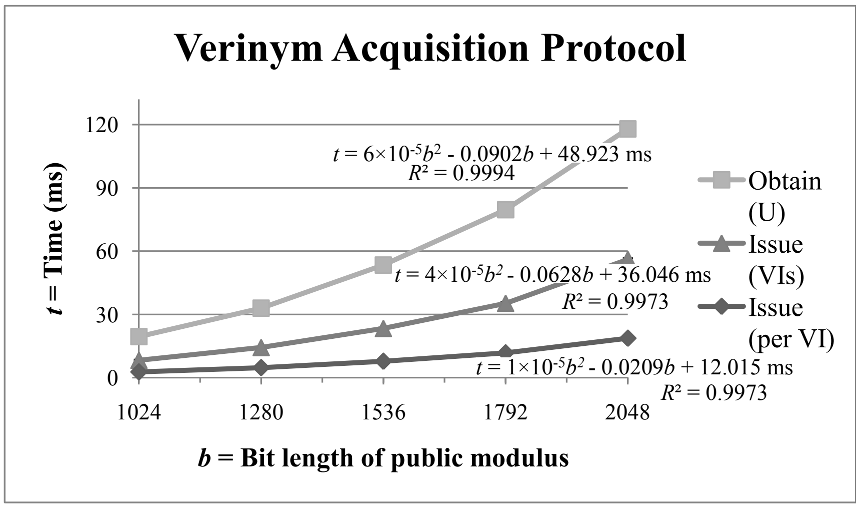

Figure 5.1: Performance measurements for U and the VI in the Verinym Acquisition Protocol. See also Table 5.1.

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 experiments use a value of $\kappa_1 = 30$ for the security parameter (as recommended by Fouque and Stern [40]). We choose $\eta = 17$ and each trial uses a random 12-bit prime e with hamming weight 3 for the public key. For each bit length, we repeated the experiment 100 times (with the VIs issuing, and U obtaining, 100 verinyms in each trial) and report here the mean execution time for issuing and obtaining a single verinym ($\pm$ the standard deviation) in milliseconds. Figure 5.1 is a graph of the data contained in Table 5.1.

Distributed key generation in our implementation has been simulated using the dealer-based version of the protocol from [28, §3]; thus, we 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 Nymbler's overall performance.

5.1.2 Verinym Showing Protocols

Performance measurements. Table 5.2 summarizes performance measurements for the Verinym Showing Protocol. The experimental setup is identical to §5.1.1. For each of the 100 experiments of the Verinym Acquisition Protocol, we used the last verinym produced to perform a trial of this experiment. The proof of knowledge of a valid verinym from U uses an expiration VVP of 5 for each trial. For each bit length, we repeated the experiment 100 times and report

Table 5.1: Performance measurements for U and the VI in the Verinym Acquisition Protocol.

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

Each experiment was repeated 100 times (with the VIs issuing, and U obtaining, 100 verinyms in each trial). The mean execution time for issuing and obtaining a single verinym ($\pm$ the standard deviation) in milliseconds across all trials is reported here. See also Figure 5.1.

here the mean execution time ($\pm$ the standard deviation) in milliseconds. Figure 5.2 is a graph of the data contained in Table 5.2. Note that the cost of the Verinym Showing Protocol is higher than the cost of the Verinym Acquisition Protocol; as a result, the the t-axis in Figure 5.2 uses a different scale than the t-axis in Figure 5.1.

Improving efficiency. The exponentiation proof (i.e., the fourth and fifth lines of $\Pi_{x_{K^*}}$, in which U proves that she knows x_{K^*} such that $(x_{K^*}^{\eta^{K^*}})^e = Y$) dominates 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 [48] 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 [4, 67].)

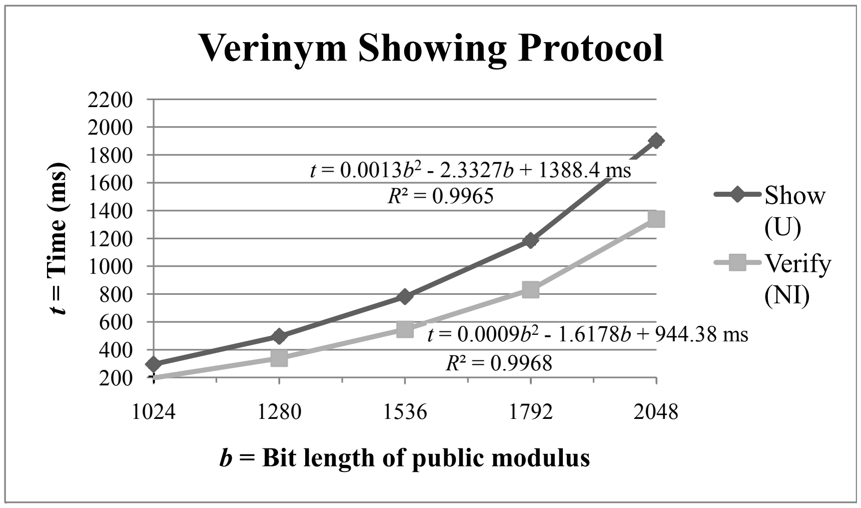

Figure 5.2: Performance measurements for U and the NI in the Verinym Showing Protocol. See also Table 5.2.

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.[1] 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 $j \cdot \lfloor \log_2(\eta) \rfloor + \lfloor \log_2(e) \rfloor$. Our measurements indicate that computing this value in $j+1$ stages reduces the cost of the Verinym Showing Protocol by a factor of about two. 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 [38] instead of Pedersen commitments. To maintain unlinkability, U chooses a random group element ϕ modulo N, computes $\Phi = \phi^4 \bmod N$ and sends (ϕ, Φ^{x_j}) along with a proof that x_j is the same value previously committed to. The remainder of the algorithm then runs as usual.

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

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

Table 5.2: 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 ± 68.58 ms
		1280	495.7 ms ± 113.89 ms
		1536	780.7 ms ± 168.05 ms
		1792	1184.2 ms ± 278.19 ms
		2048	1901.2 ms ± 483.36 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

Each experiment was repeated 100 times. The mean execution time (± the standard deviation) in milliseconds across all trials is reported here. See also Figure 5.2.

5.2 Nymble Construction

5.2.1 VERBS

Performance measurements. Table 5.3 summarizes performance measurements for VERBS verification. Note that the cost of VERBS verification dominates the cost of the Nymble Showing Protocol (the rest of the cost comes from a lookup in a hash map). For each bit length of the public modulus, we repeated the experiment 100 times and report here the mean execution time (± the standard deviation) in milliseconds. Figure 5.3 is a graph of the data contained in Table 5.3.

We omit performance measurements for the other VERBS protocols, but note that we do include indirect performance measurements for these protocols since they comprise a majority of the Nymble Acquisition Protocol.

5.2.2 Nymble Acquisition Protocol

Performance measurements. Table 5.4 summarizes performance measurements for the Nymble Acquisition Protocol. As in the previous experiments, we ran both U and the NI on a single

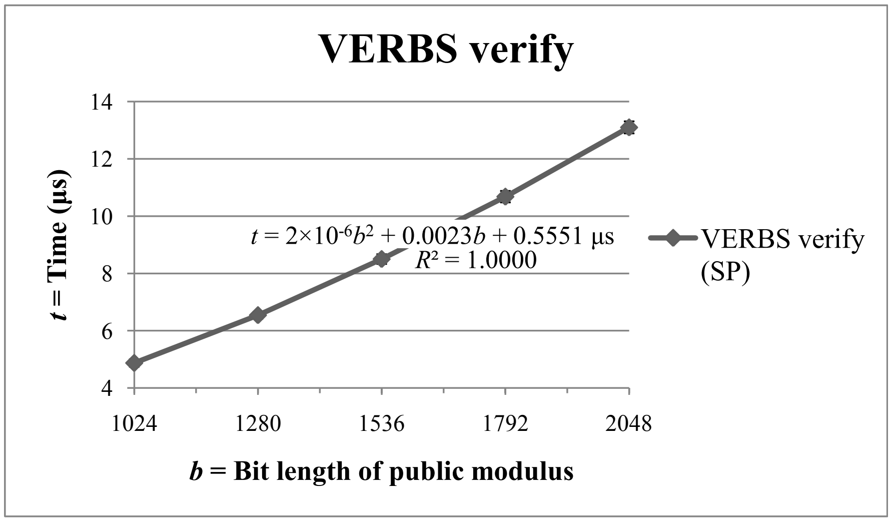

Figure 5.3: Performance measurements for VERBS verification. See also Table 5.3.

machine (so communication latency is not measured). Our implementation uses a bit-length of $\ell_c = 30$ for the challenge hash in the exponentiation proofs (see §5.3.1). A bit length of k bits indiates that n is $k - 2$ bits long and ζ is k-bits long. For each bit length of n and ζ, we repeated the experiment 100 times and report here the mean execution time ($\pm$ the standard deviation) in milliseconds. Figure 5.4 is a graph of the data for 1536-bit moduli contained in Table 5.4.

Improving efficiency. Our implementation uses the Fiat-Shamir heuristic to make each of the ZKPs noninteractive [39]; this results in an expensive computation in the nest proof (the most expensive part of the protocol). The cost of this computation scales linearly with the bit length of the hash function used for the challenge. We therefore introduce the security parameter κ_3 to specify the bit length of the challenge. In our implementation we use $\kappa_3 = 30$—which is relatively small, since U is only expected to do 2^{κ_3} work to forge a nymble—but we note that κ_3 can be safely reduced to, say, $\kappa_3 = 20$, without reducing security if the noninteractive protocol is replaced by an interactive one. This would result in about a one-third reduction in the computation time for the NI to verify the ZKP from U, at the cost of one additional round of interaction latency and the bandwidth required for the NI to send a single group element to U.

Table 5.3: Performance measurements for VERBS verification.

Operation	Host	Bit length of modulus	Mean execution time ± standard deviation (μs)
Verify VERBS	SP	1024	4.87 μs ± 0.13 μs
		1280	6.54 μs ± 0.12 μs
		1536	8.50 μs ± 0.17 μs
		1792	10.68 μs ± 0.20 μs
		2048	13.10 μs ± 0.21 μs

Each experiment was repeated 100 times. The mean execution time (± the standard deviation) in microseconds across all trials is reported here. See also Figure 5.3.

5.3 Revocation Mechanisms

5.3.1 Pseudonym Extraction Protocol

Performance measurements. Table 5.5 summarizes performance measurements for the trapdoor discrete logarithm computation in the Pseudonym Extraction Protocol. These experiments were performed on a 2.4GHz Intel Xeon E5620 with two Tesla M2050s running Ubuntu 10.04 64-bit. The experiment was performed 100 times for each value of ℓ_B. Each trial used a new pseudorandomly generated 1536-bit modulus ζ such that $\varphi(\zeta)$ is 2^{ℓ_B}-smooth. We report here the mean execution time (± the standard deviation) in milliseconds of this protocol. Figure 5.5 is a graph of the data contained in Table 5.5.

Parameter selection. A discrete logarithm computation takes about $c \cdot (\ell_\zeta/\ell_B) \cdot 2^{\ell_B/2}$ modular multiplications using van Oorschot and Wiener's parallel ρ algorithm [91], which are almost completely parallelizable, for some constant of proportionality c. If the PE has a parallelism factor of Ψ (i.e., Ψ is the number of cores available to the PE), this will be about $\frac{\ell_\zeta}{\ell_B} \cdot \frac{c \cdot 2^{\ell_B/2}}{\Psi \cdot \mu}$ minutes to compute a discrete log, where μ is the number of multiplications modulo an $(\ell_\zeta/2)$-bit modulus that can be computed by one core in one minute.

So we want to choose ℓ_B such that

$$\frac{2^{\ell_B/2}}{\ell_B} \approx \frac{t_{\text{dl}} \cdot \mu \cdot \Psi}{\ell_\zeta \cdot c}, \tag{5.1}$$

where t_{dl} is the desired wall-clock time (in minutes) for the PE to compute a trapdoor discrete logarithm.

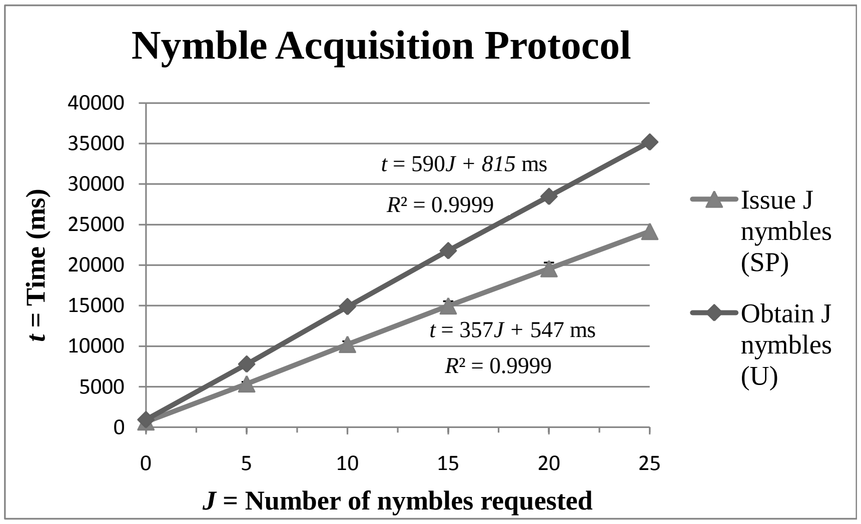

Figure 5.4: Performance measurements for U and the NI in the Nymble Acquisition Protocol with 1536-bit moduli. See also Table 5.4.

On the other hand, it takes at least about $\frac{3}{5} \cdot 2^{\ell_B}$ multiplications modulo an ℓ_ζ-bit modulus to factor ζ, taking advantage of its special form by using Pollard's $p-1$ factoring algorithm [68].[2] This algorithm is inherently sequential [12]; only a small speedup can be obtained, even with a very large degree of parallelism.[3] This means it will take about $\frac{3}{5} \cdot \frac{2^{\ell_B}}{\mu}$ minutes to factor ζ. Assuming $\mu = 23100000$ (as is the case on our Q9550 benchmark system, which has faster individual cores than the Tesla machine), then $\ell_B = 53$ yields over 445 years to factor ζ, and $\ell_B = 54$ yields over 890 years to factor ζ. (Remember again that this is *wall-clock* time, not CPU time.) Note also that a different ζ can be used for each SP and for each linkability window, thus reducing the value of expending even that much effort.

The reason we seem to be making the unusual claim that 2^{53} security is sufficient is twofold: first, these are counts of multiplications modulo an ℓ_ζ-bit modulus, each of which takes about $2^{12.8}$ cycles for our suggested $\ell_\zeta = 1536$; thus, we are really proposing about 2^{65} security here. More importantly, these are counts of *sequential operations*. When one typically speaks of 2^{80}

[2]Other factoring algorithms, such as the elliptic curve factorization method (ECM) [56] or the quadratic sieve algorithm (QS) [69], are highly parallelizable and can factor a general modulus with *sublinear* asymptotic complexity; however, the linear cost of Pollard's $p-1$ factoring algorithm is by far the most efficient method for factoring ζ, given its special form and our parameter selection. In other words, while both of the aforementioned algorithms have superior asymptotic complexity to Pollard's $p-1$ factoring algorithm (depending on how one asymptotically relates ℓ_B and ℓ_ζ), the actual position on the $O(2^{\ell_B})$ cost curve in our case is much smaller than the corresponding position on the cost curves for these asymptotically faster algorithms.

[3]Of course, with *arbitrarily* large parallelism, other algorithms can factor ζ more quickly *without* taking advantage of the special form of ζ; massively parallel trial division is an extreme example.

Table 5.4: Performance measurements for U and the NI in the Nymble Acquisition Protocol.

Operation	Host	Bit length of modulus	Mean execution time (ms) and correlation coefficient (R^2)
Issue nymble	NI	1024	$327J + 223$ ms $(R^2 = 0.9999)$
		1280	$592J + 435$ ms $(R^2 = 0.9999)$
		1536	$940J + 768$ ms $(R^2 = 0.9998)$
		1792	$1450J + 1068$ ms $(R^2 = 0.9999)$
		2048	$1445J + 638$ ms $(R^2 = 0.9999)$
Obtain nymble	U	1024	$477J + 315$ ms $(R^2 = 1.0000)$
		1280	$863J + 592$ ms $(R^2 = 1.0000)$
		1536	$1368J + 1102$ ms $(R^2 = 0.9998)$
		1792	$2113J + 1553$ ms $(R^2 = 0.9999)$
		2048	$1928J + 840$ ms $(R^2 = 0.9999)$

We performed experiments in which the client obtains between 5 and 25 nymbles at a time (in increments of 5). Each experiment was repeated 100 times. A line of best fit for the mean execution time in milliseconds across all trials is reported here. See also Figure 5.4.

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.

As noted by Maurer and Yacobi [63, §4], since the cost of factoring increases with 2^{ℓ_B}, while the cost of computing discrete logarithms increases with $2^{\ell_B/2}$, it follows that as cores get faster (μ increases) and more numerous (Ψ increases), the time to factor ζ only goes *up* with respect to the time to compute discrete logarithms. If μ increases by a factor of f, then this leads to a net security increase by a factor of f; if Ψ increases by a factor of g, this leads to a net security increase by a factor of g^2. These calculations suggest that the trapdoor discrete logarithm groups will get *more secure over time*.

5.3.2 Non-membership Proof Protocol

Performance measurements. Table 5.6 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

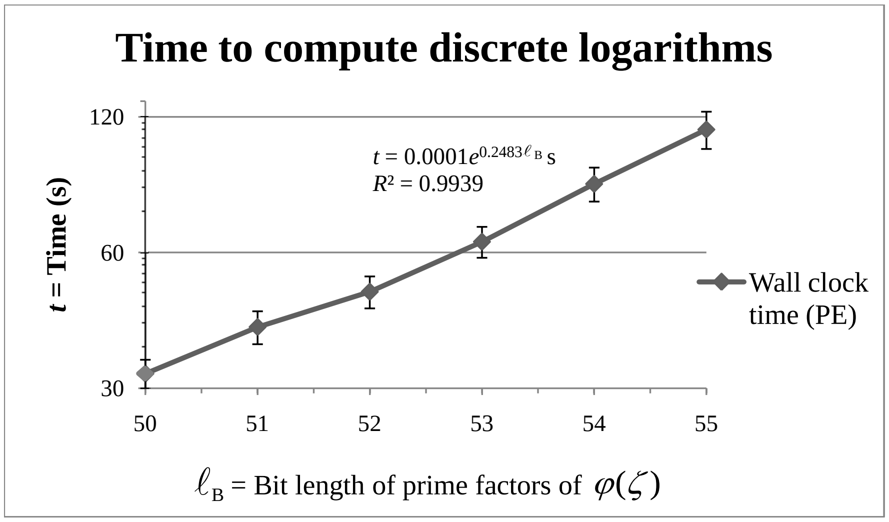

Figure 5.5: Performance measurements for the PE in trapdoor discrete logarithm computation in the Pseudonym Extraction Protocol. Note that the time axis of this plot uses a logarithmic scale. See also Table 5.5.

for the expected latency due to communication between the client 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 analyze below.) We offer some suggestions below about how these performance measurements could likely be improved. Figure 5.6 is a graph of the data contained in Table 5.6.

Comparison with other work. Our measurements indicate that our unoptimized implementation dramatically outperforms the approach taken by BLAC, EPID[4], and PEREA for even moderate-sized blacklists. Figures 5.7—5.9 compare the performance of our approach with the non-membership proofs in BLAC and PEREA. 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 U and the SP. Indeed, our approach is practical even for extremely large SPs such as Wikipedia and Slashdot.

Cost analysis. 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.

[4]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 [86].

Table 5.5: Performance measurements for PE in trapdoor discrete logarithm computation in the Pseudonym Extraction Protocol.

Operation	Host	ℓ_B	Mean execution time $\pm$ standard deviation (s)
Compute discrete logarithm	PE	50	32.3 s $\pm$ 2.4 s
		51	41.0 s $\pm$ 3.4 s
		52	49.1 s $\pm$ 4.0 s
		53	63.4 s $\pm$ 5.0 s
		54	85.2 s $\pm$ 7.4 s
		55	112.5 s $\pm$ 10.7 s

Each experiment was repeated 100 times. The mean execution time ($\pm$ the standard deviation) in seconds across all trials is reported here. See also Figure 5.5.

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-λ polynomials is monic, there are only λ 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 $\lambda \cdot \mu_\rho$ work using Horner's method; thus, the cost for this step is $\lambda \cdot \lambda \cdot \mu_\rho = \Lambda \cdot \mu_\rho$. Similarly, computing ς_i requires $\lambda \cdot \mu_\rho$ work; thus, the cost for this step is also $\lambda \cdot \lambda \cdot \mu_\rho = \Lambda \cdot \mu_\rho$. So far, this is $2\Lambda \cdot \mu_\rho$ 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 λ such commitments is then $2\lambda(\chi_\rho + \mu_P)$. The same analysis applies to computing each C_{o_i}, thus yielding a total of $4\lambda(\chi_\rho + \mu_P) + 2\Lambda \cdot \mu_\rho$ work for U. The bandwidth cost for U to upload each commitment C_i and C_{o_i} to the NI is just $2\lambda \cdot \ell_P$ bits. Together with the cost of U downloading the blacklist from the NI, this yields a total communications complexity of $\Lambda \cdot \ell_\rho$ bits download and $2\lambda \cdot \ell_P$ bits upload for U (and vice-versa for the NI).

The left-hand side of the batch proof, Equation 4.3, requires λ exponentiations with small exponents and $\lambda - 1$ multiplications modulo P; this is $\lambda \cdot \chi_{\kappa_2} + (\lambda - 1) \cdot \mu_P$ work. The right-hand side requires $\lambda(\lambda + 1)$ multiplications modulo ρ to compute the exponents, followed by $\lambda + 1$ exponentiations with exponents modulo ρ and λ multiplications modulo P; this is $\lambda \cdot \chi_\rho + \Lambda \cdot \mu_\rho + \lambda \cdot \mu_P$ work.

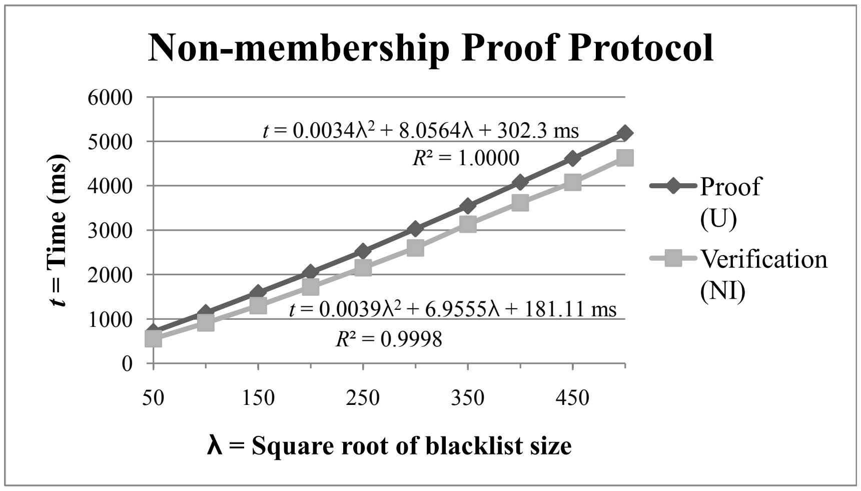

Figure 5.6: Performance measurements for U and the NI in the Non-membership Proof Protocol. See also Table 5.6.

The costs of the Verinym Showing Protocol and of the proof Π_1 that the SP-specific pseudonym is computed correctly are independent of the blacklist size Λ. Therefore, we omit them from our detailed analysis of the dependence of the protocol's cost on Λ. All that is left to be considered, then, is the cost of the large zero-knowledge proof, Π_2, in Equation 4.1.

The first line of Π_2 requires U to compute $4(\lambda-1)$ exponentiations with ℓ_ρ-bit exponents, 2λ multiplications modulo P and 3λ multiplications modulo ρ for a total cost of $4(\lambda-1)\cdot\chi_\rho + 2\lambda\cdot\mu_P + 3\lambda\cdot\mu_\rho$. U is also required to upload $2\cdot(\lambda-1)\cdot\ell_P + 3(\lambda-1)\cdot\ell_\rho$ 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 $2\lambda\cdot\chi_\rho$ work from the NI, using knowledge of $\psi_\rho = \log_{\alpha_\rho}(\beta_\rho)$ to reduce the number of exponentiations. The second and third lines of Π_2 require U to compute λ multiplicative inverses modulo P, λ multiplications modulo P, λ multiplicative inverses modulo ρ, 2λ multiplications modulo ρ and two exponentiations with ℓ_ρ-bit exponents. Thus the cost for U is $\lambda(\iota_P + \mu_P + \iota_\rho + 2\mu_\rho) + 2\chi_\rho$. Similarly, the NI can verify these proofs with $\lambda\chi_\rho + \lambda\iota_P$ work. This is done using Brands' NOT proof [9, §3]: to prove

$$PK\left\{(o_i, \varsigma_i)\ :\ C_{o_i} = \alpha_\rho^{o_i}\beta_\rho^{\varsigma_i} \bmod P \wedge (o_i \neq 0)\right\},$$

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

Table 5.6: Performance measurements for U and the NI in the Non-membership Proof Protocol.

Operation	Host	λ	$\Lambda = \lambda^2$	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

Each experiment was repeated 100 times. The mean execution time ($\pm$ the standard deviation) across all trials is reported here. See also Figures 5.6–5.9.

Thus, the overall computational cost of this protocol for U is $2(\Lambda + 5\lambda) = O(\Lambda)$ multiplications modulo ρ, $7\lambda = O(\lambda)$ multiplications modulo P, $\lambda = O(\lambda)$ multiplicative inverses modulo ρ, $\lambda = O(\lambda)$ multiplicative inverses modulo P, and $8\lambda - 2 = O(\lambda)$ exponentiations modulo P with ℓ_ρ-bit exponents. The cost for the NI is $\Lambda = O(\Lambda)$ multiplications modulo ρ, $2\lambda - 1 = O(\lambda)$ multiplications modulo P, $\lambda = O(\lambda)$ multiplicative inverses modulo P, $4\lambda = O(\lambda)$ exponentiations modulo P with ℓ_ρ-bit exponents, and $\lambda = O(\lambda)$ exponentiations modulo P with ℓ_{κ_2}-bit exponents. Communication costs are $\Lambda \cdot \ell_\rho$ bits download and $((7\lambda - 3) \cdot \ell_\rho + 2(\lambda - 1) \cdot \ell_P)$ bits upload for U, and vice versa for the NI. As noted in [51], 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.

Improving efficiency. Our implementation does not make use of the NI's knowledge of $\psi_\rho = \log_{\alpha_\rho}(\beta_\rho) \bmod P$ to improve the efficiency of the verification equations. Also, we do not implement Brands' error correcting factors technique [9], as suggested in [10, 11]. This would allow much of the computational cost of this algorithm to be converted into precomputation. We refer the reader to [10] or [9, §5.4.2] for details.

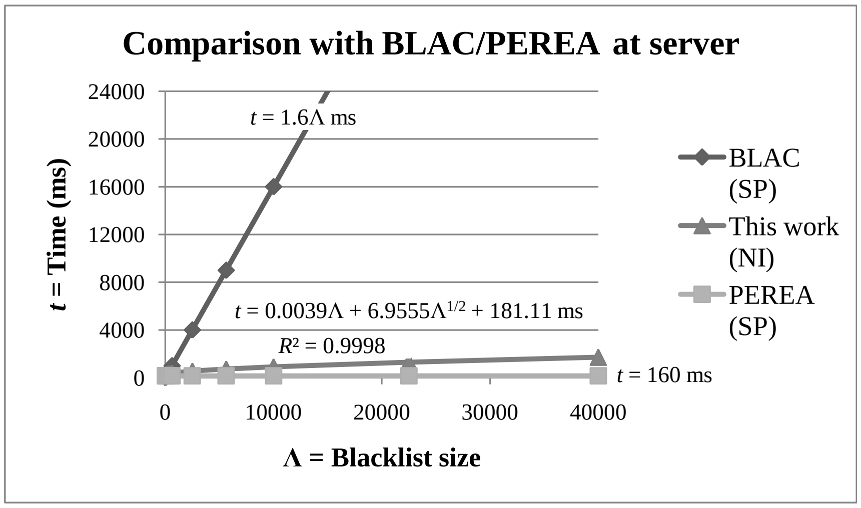

Figure 5.7: Comparison of non-membership proofs at the server for large blacklist sizes. See also Table 5.6.

Observe that, for small blacklist sizes, the naive linear non-membership proof employed by BLAC outperforms our more sophisticated square-root-time algorithm. In particular, the number of exponentiations in BLAC's approach scales linearly with the size of the blacklist (about 1.8 ms per entry at the client and 1.6 ms per entry at the server). Our algorithm has a very small linear component (a small number of multiplications modulo a small prime), but the number of exponentiations scales with the square root of the size of the blacklist. For blacklists of size at most 250000, our measurements indicate that the coefficient of the linear component is only about 3400 ns per entry and the coefficient of the square root term is about 8 ms. However, a significant constant component (about 302 ms) makes the linear algorithm outperform our approach for small blacklists; it would therefore be beneficial in these instances to employ the linear-time algorithm. 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 5.7, 5.8 and 5.9 compare the relative complexities of the different non-membership proofs for various blacklist sizes.

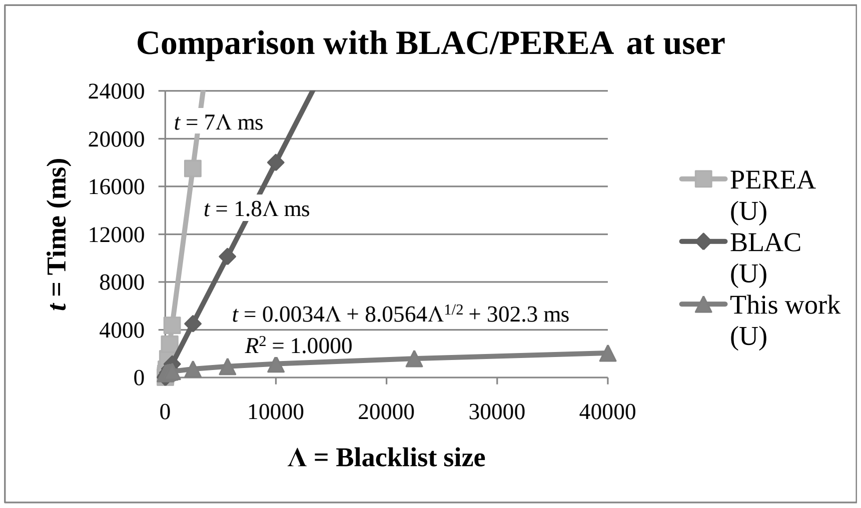

Figure 5.8: Comparison of non-membership proofs at the user for large blacklist sizes. See also Table 5.6.

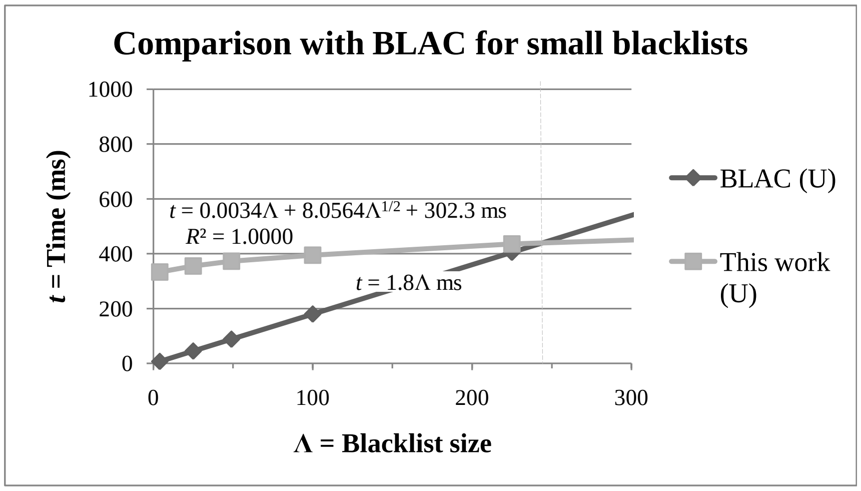

Figure 5.9: Comparison of non-membership proofs at the user for small blacklist sizes. See also Table 5.6.

Conclusion

The ability to communicate without fear of network surveillance makes it possible for users to research sensitive topics and share information and ideas that they may otherwise be unwilling to reveal for fear of persecution, punishment or simply embarrassment. Deployed anonymous communications networks, such as Tor, help users protect themselves from network surveillance and thereby provide an important public service. Unfortunately, some users abuse the anonymity provided by anonymous communications networks. In response, several major service providers no longer allow anonymous users to participate in their online communities. For example, Tor users are unable to edit articles on Wikipedia, post comments on Slashdot, or chat on most major IRC networks. This hurts the Tor network by discouraging casual use of Tor and thus reducing the size of each Tor user's anonymity set.

Anonymous blacklisting systems try to solve this by enabling users to authenticate anonymously with service providers, while giving service providers the ability to revoke access to their services from individual misbehaving anonymous users. Nymble is an efficient anonymous blacklisting system that implements a privacy-preserving form of IP address banning. However, Nymble makes extensive use of trusted third parties that can easily deanonymize users of the system. Other schemes can provide very strong security and privacy guarantees without using trusted third parties, but their computational costs make them impractical for real-world deployment.

We have thus proposed a new anonymous blacklisting system built in an extended Nymble framework. Our *Nymbler* scheme reduces Nymble's trust assumptions, improves on its security guarantees, and provides some new functionality. Nymbler maintains much of the original Nymble's efficiency and is highly practical even for use at extremely large service providers like Wikipedia and Slashdot. We employ several novel cryptographic constructions in our system, including our distributed threshold Verinym Issuer, our verifier-efficient restricted blind signature (VERBS) scheme, trapdoor discrete logarithms, and a highly efficient zero-knowledge proof of non-membership in a list. These tools enable us to solve several previously unsolved problems in the anonymous blacklisting systems literature.

One may pursue several directions to improve our system and we conclude this thesis by discussing some useful extensions that warrant future research. As first mentioned by Tsang *et* *al.* in the original Nymble paper [54], a useful feature would be to provide service providers with the ability to block *entire subnets* in addition to just individual IP addresses. While there exist straightforward modifications to our construction that would make this possible, these modifications would negatively affect user privacy; thus, we leave it to future work to develop a privacy-friendly solution to this problem. On the other hand, there are also situations in which it would be desirable to give *certain IP addresses* the ability to misbehave a threshold number of times before revoking access to the users behind these addresses. For example, large institutions (such as universities) often have many users who share a single IP address through NAT; in such cases, it might be useful to allow the institution to run its own VI (and perhaps NI) internally that issues verinyms (and nymbles) based on internal IP addresses. In this case, users that obtain verinyms (and nymbles) from the internal VI (and NI) could access Nymbler-enabled services concurrently and unlinkably. However, if more than some small threshold of internal users have their access revoked from a service provider, then this would result in an institution-wide revocation at the affected service provider.

Another useful extension would be to provide service providers with the ability to detect repeat offenders and revoke these users' access for longer durations of time. (For example, Wikipedia's blocking policy states that administrators should consider "the severity of the behavior; [and] whether the user has engaged in that behavior before" when deciding on the duration of a block [94, "Duration of blocks"].) Indeed, this is a trivial extension when the misbehaviours all occur in a single linkability window (i.e., before the user's access if revoked); however, detecting repeat offenders whose offenses occur in different linkability windows seems to require the ability for some party to link *all users'* actions across linkability windows, which is clearly undesirable.

Finally, while we have already implemented the key components of our system, our present implementation only serves as a proof of concept. In order to deploy our system in the wild, we must first expand this implementation into an industrial-grade software package. This would involve building both the infrastructure necessary to support the service, as well as the server-side and client-side tools needed to make it usable. In particular, it would be useful to develop both IRC and HTTP proxy servers, as well as web browser and IRC client support modules (likely in the form of plug-ins for existing clients).

APPENDICES

Appendix A

Security Games

A challenger $\mathcal{C}$ and probabilistic polynomial time adversary $\mathcal{A}$ compete in each security game in this appendix. If $\mathcal{A}$ controls a server, then $\mathcal{A}$ knows all of that server's secrets and can direct that server's actions. In particular, $\mathcal{A}$ may direct the server to deviate from the established protocols. If it is not explicitly stated that a particular server is under $\mathcal{A}$'s control, then it is implicitly assumed that this server behaves honestly. (Threshold entities, such as our threshold VI, are an exception to this rule: $\mathcal{A}$ always controls *at least one fewer than the threshold number* of any threshold entity; further, if it is explicitly stated that $\mathcal{A}$ controls a server that is a threshold entity, then $\mathcal{A}$ controls each of the servers that comprise this entity.) We also consider a set $\mathbf{U}$ of users. $\mathcal{C}$ controls each user in $\mathbf{U}$; however, the rules of each game permit $\mathcal{A}$ to compromise some subset of users in $\mathbf{U}$, learning all of those users' secrets and controlling their future actions. In each game, SP_1 is under the control of $\mathcal{C}$ and behaves honestly, while SP_2 is under the control of $\mathcal{A}$ and may arbitrarily deviate from the protocols.

Misauthentication resistance

Security Game A.1 (Misauthentication resistance) [See Definition 4, page 10]

Adversary: $\mathcal{A}$ controls SP_2 and the PE.

—

(**Probing phase**): $\mathcal{A}$ arbitrarily and adaptively compromises any subset of $\mathbf{U}$, obtains verinyms for these users from the VIs, and acquires nymbles for them for SP_1 and SP_2 from the NI. $\mathcal{A}$ may authenticate any compromised users with these SPs as desired, and may have any nymble revoked from either SP.

(**End phase**): $\mathcal{A}$ sends a nymble ν to SP_1. $\mathcal{A}$ wins the security game if and only if:

1. ν was not output by a correct execution of the *Nymble Acquisition Protocol*; and,
2. SP_1 accepts ν as valid.

Backward anonymity

Security Game A.2 (Backward anonymity) [See Definition 5, page 10]

Adversary: $\mathcal{A}$ controls SP_2, the NI, and the PE.

—

(**First probing phase**): $\mathcal{A}$ arbitrarily and adaptively compromises a subset of $\mathbf{U}$ of size at most $|\mathrm{U}| - 2$, obtains verinyms for these users from the VIs, and acquires nymbles for them for SP_1 and SP_2 from the NI. $\mathcal{A}$ may authenticate any compromised users with these SPs as desired, or have any nymble revoked from either SP.

(**Challenge phase**): For each uncompromised user $\mathrm{U}_i \in \mathbf{U}$, $\mathcal{C}$ obtains a verinym from the VIs and acquires nymbles for SP_2 from the NI. $\mathcal{C}$ then authenticates each uncompromised user with SP_2 (the order of authentications is random); U_i's nymble from this step is ν_i. Note that $\mathcal{A}$ learns each ν_i, but does not learn with which user each nymble is associated.

(**Second probing phase**): $\mathcal{A}$ may ask $\mathcal{C}$ to authenticate any user with SP_2 according to any strategy, and may have any nymble revoked from either SP.

(**End phase**): $\mathcal{A}$ chooses a tuple (U_i, ν_j). $\mathcal{A}$ wins the game if and only if $i = j$.

Unlinkability

Security Game A.3 (Unlinkability) [See Definition 6, page 10]

Adversary: $\mathcal{A}$ controls SP_2 and the NI.

—

(**First challenge phase**): For $\mathrm{U}_0, \mathrm{U}_1 \in \mathbf{U}$, $\mathcal{C}$ obtains a verinym from the VIs and acquires nymbles for SP_1 and SP_2 from the NI. $\mathcal{C}$ then authenticates both users with both SPs; the nymble used by U_i at SP_j is $\nu_{(i,j)}$. $\mathcal{C}$ reveals each $\nu_{(i,j)}$ to $\mathcal{A}$.

(**Probing phase**): $\mathcal{A}$ arbitrarily and adaptively compromises any subset of $\mathbf{U} - \{\mathrm{U}_0, \mathrm{U}_1\}$, obtains verinyms for these users from the VIs, and acquires nymbles for them for SP_1 and SP_2 from the NI. $\mathcal{A}$ may authenticate any compromised users with these SPs as desired, and may choose any compromised users' nymbles to be revoked from either SP.

(**Second challenge phase**): $\mathcal{C}$ flips two fair coins to obtain bits $a, b \in_R \{0, 1\}$. $\mathcal{C}$ authenticates U_a with SP_{b+1}.

(**End phase**): $\mathcal{A}$ chooses a nymble $\nu_{(i,j)}$. $\mathcal{A}$ wins the game if and only if $i = a$.

Revocability

Security Game A.4 (Revocability) [See Definition 7, page 11]

Adversary: $\mathcal{A}$ controls SP$_2$.

—

(**Probing phase**): $\mathcal{A}$ arbitrarily and adaptively compromises any subset of **U**, obtains verinyms for these users from the VIs, and acquires nymbles for them for SP$_1$ and SP$_2$ from the NI. $\mathcal{A}$ may authenticate any compromised users with these SPs as desired, and may choose any compromised users' nymbles to be revoked from either SP. By the end of this phase, $\mathcal{A}$ must authenticate each compromised user with SP$_1$ at least once.

(**Challenge phase**): $\mathcal{C}$ sends to the PE the last nymble used at SP$_1$ by each compromised user to have these users revoked from SP$_1$.

(**End phase**): $\mathcal{A}$ attempts to authenticate with SP$_1$ using some nymble ν. $\mathcal{A}$ wins the game if and only if SP$_1$ accepts ν.

Revocation auditability

Security Game A.5 (Revocation auditability) [See Definition 8, page 11]

Adversary: $\mathcal{A}$ controls SP$_2$ and the NI.

—

(**First challenge phase**): For $\mathrm{U}_0, \mathrm{U}_1 \in \mathbf{U}$, $\mathcal{C}$ obtains a verinym from the VIs and acquires nymbles for SP$_1$ and SP$_2$ from the NI. $\mathcal{C}$ then authenticates both users with both SPs; the nymble used by U$_i$ at SP$_j$ is $\nu_{(i,j)}$. $\mathcal{C}$ reveals each $\nu_{(i,j)}$ to $\mathcal{A}$.

(**Probing phase**): $\mathcal{A}$ arbitrarily and adaptively compromises any subset of $\mathbf{U} - \{\mathrm{U}_0, \mathrm{U}_1\}$. $\mathcal{A}$ may authenticate any compromised users with either SP as desired, may ask $\mathcal{C}$ to authenticate any uncompromised users with either SP according to any strategy, and may have any nymble revoked from either SP.

(**Second challenge phase**): $\mathcal{C}$ flips a fair coin to obtain a bit $a \in_R \{0, 1\}$. For U$_0$ and U$_1$, $\mathcal{C}$ then invokes the *Revocation Audit Protocol* to check that user's revocation status at SP$_2$. We denote U$_i$'s reported revocation status at SP$_2$ by $b_i \in \{\texttt{true}, \texttt{false}\}$.

(**End phase**): $\mathcal{A}$ chooses a nymble $\nu_{(i,j)}$ from the first challenge phase. $\mathcal{A}$ wins the game if and only if $i = a$ and $b_0 = b_1 = \texttt{true}$.

Non-frameability

Security Game A.6 (Non-frameability) [See Definition 9, page 11]

Adversary: $\mathcal{A}$ controls SP_2 and the NI.

—

(**First challenge phase**): For each $\mathrm{U}_i \in \mathbf{U}$, $\mathcal{C}$ obtains verinyms from the VIs and acquires nymbles for SP_1 and SP_2 from the NI. $\mathcal{C}$ then authenticates each user with both SPs; $\mathcal{C}$ reveals all nymbles from this phase to $\mathcal{A}$.

(**Probing phase**): $\mathcal{A}$ arbitrarily and adaptively compromises any proper subset of $\mathbf{U}$. $\mathcal{A}$ may authenticate any compromised users with either SP as desired, may ask $\mathcal{C}$ to authenticate any uncompromised users with either SP according to any strategy, and may have any compromised users' nymbles revoked from either SP.

(**Second challenge phase**): For each uncompromised user $\mathrm{U}_i \in \mathbf{U}$, $\mathcal{C}$ invokes the *Revocation Audit Protocol* to check that user's revocation status at SP_2, and then attempts to authenticate that user with SP_2 (regardless of the user's reported revocation status). We denote U_i's reported revocation status at SP_2 by $b_i \in \{\texttt{true}, \texttt{false}\}$, while $c_i \in \{\texttt{true}, \texttt{false}\}$ indicates if U_i successfully authenticated with SP_2.

(**End phase**): $\mathcal{A}$ wins the game if and only if there exists an uncompromised user U_i such that $b_i = \texttt{true}$ or $c_i = \texttt{false}$.

Appendix B

VERBS Protocols

VERBS-Blind

Protocol B.1 VERBS-Blind$(g, h, C_h, \gamma_1, \xi)$ [Run by U]

Input: g, h, C_h, γ_1, ξ [$g \in \mathbb{Z}_m^*$; $M = 4m + 1$ is prime]

Output: θ, ν', Π_ν

Compute: $\nu \leftarrow g^h \bmod \zeta$

Compute: $C_\nu \leftarrow \alpha_m^\nu \beta_m^{\gamma_2}$

Choose: $\theta \in_R \mathbb{Z}_m^*$ [Blinding factor]

Compute: $C_{\nu'} \leftarrow \alpha_m^{\theta^3\left(\nu^2+(\nu \bmod \xi)\right)} \beta_m^{\gamma_3}$ [Commit to blinded nymble]

Set: $\nu' \leftarrow \theta^3\left(\nu^2 + (\nu \bmod \xi)\right) \bmod m$ [Open commitment]

$$\textbf{Set: } \Pi_\nu \leftarrow PK\left\{\begin{pmatrix} h, \gamma_1, \\ \nu, \\ \gamma_2, \\ \\ \nu' \\ \theta, \gamma_3 \end{pmatrix} : \begin{array}{lll} & C_h = \alpha_n^h \beta_n^{\gamma_1} & \bmod N \\ \wedge & \nu = g^h & \bmod n \\ \wedge & C_\nu = \alpha_m^\nu \beta_m^{\gamma_2} & \bmod M \\ \wedge & 0 \le h < n & \\ \wedge & \nu' = \theta^3\left(\nu^2 + (\nu \bmod \xi)\right) & \bmod m \\ \wedge & C_{\nu'} = \alpha_m^{\nu'} \beta_m^{\gamma_3} & \bmod M \end{array}\right\}$$

Return: (θ, ν', Π_ν)

VERBS-Sign

Protocol B.2 VERBS-Sign($\nu', p_m, q_m, \xi, \Pi_\nu$) [Run by NI]

Input: ν', p_m, q_m, ξ, Π_ν
Output: σ' or $\perp$

```
if ($\Pi_\nu$ is correct) then
    Set: $\sigma' \leftarrow (\nu')^{\frac{1}{3}} \bmod m$                [Uses knowledge of $p_m$ and $q_m$]
else
    Set: $\sigma' \leftarrow \perp$
end if
```

Return: σ'

VERBS-Unblind

Protocol B.3 VERBS-Unblind(σ', θ) [Run by U]

Input: σ', θ
Output: σ

Set: $\sigma \leftarrow \sigma' \cdot \theta^{-1} \bmod m$

Return: σ

VERBS-Verify

Protocol B.4 VERBS-Verify(ν, σ, ξ) [Run by SP]

Input: ν, σ, ξ
Output: $b \in \{\texttt{true}, \texttt{false}\}$

Set: $b \leftarrow \left(\sigma^3 \overset{?}{\equiv} (\nu^2 + (\nu \bmod \xi)) \bmod m\right)$

Return: b

Bibliography

[1] Reed S. Abbott. CPG: Closed Pseudonymous Groups. Master's thesis, Brigham Young University Computer Science Department, Provo, Utah, USA, April 2008. (One citation on page 17.)

[2] Reed S. Abbott, Timothy W. van der Horst, and Kent E. Seamons. CPG: Closed Pseudonymous Groups. In Vijay Atluri and Marianne Winslett, editors, *Proceedings of WPES 2008*, pages 55–64. Association for Computing Machinery (ACM) Press, New York, NY, USA, October 2008. (One citation on page 17.)

[3] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and Noninteractive Anonymous Credentials. In Ran Canetti, editor, *Proceedings of TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer-Verlag, Berlin/Heidelberg, 2008. (One citation on page 22.)

[4] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In Kaisa Nyberg, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer-Verlag, Berlin/Heidelberg, May 1998. (3 citations on pages 21, 48, and 53.)

[5] Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Tom Heydt-Benjamin, Dieter Sommer, and Greg Zaverucha. Cryptographic Protocols of the Identity Mixer Library, v. 1.0. Research Report RZ3730, IBM Research GmbH, Zurich, Switzerland, March 2009. (One citation on page 22.)

[6] Fabrice Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In Bart Preneel, editor, *Advances in Cryptology: Proceedings of EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer-Verlag, Berlin/Heidelberg, 2000. (One citation on page 27.)

[7] Stefan Brands. Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). In Douglas R. Stinson, editor, *Advances in Cryptology: Proceedings of CRYPTO'93*,

volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer-Verlag, Berlin/Heidelberg, 1993. (One citation on page 22.)

[8] Stefan Brands. Restrictive Blinding of Secret-Key Certificates. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology: Proceedings of EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 231–247. Springer-Verlag, Berlin/Heidelberg, 1995. (One citation on page 21.)

[9] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. The MIT Press, Cambridge, Massachusetts, USA, first edition, August 2000. ISBN 978-0-262-02491-4. [Online] Available: `http://www.credentica.com/the_mit_pressbook.php`. (5 citations on pages 21, 22, 62, and 63.)

[10] Stefan A. Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. Technical Report CW472, Katholieke Universiteit Leuven, Department of Computer Science, Leuven, Belgium, December 2006. 22 pages. [Online] Available: `http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW472.pdf`. This is the extended version of [11]. (7 citations on pages 21, 46, 47, 63, and 76.)

[11] Stefan A. Brands, Liesje Demuynck, and Bart De Decker. A Practical System for Globally Revoking the Unlinkable Pseudonyms of Unknown Users. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Proceedings of ACISP 2007*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415. Springer-Verlag, Berlin/Heidelberg, 2007. An extended version of this paper is available [10]. (7 citations on pages 2, 21, 46, 47, 63, and 76.)

[12] Richard P. Brent. Parallel Algorithms for Integer Factorisation. In J. H. Loxton, editor, *Number Theory and Cryptography*, volume 154 of *London Mathematical Society Lecture Note Series*, pages 26–37. Cambridge University Press, Oxford, United Kingdom, 1990. (One citation on page 58.)

[13] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In Peng Ning and Ting Yu, editors, *Proceedings of WPES 2007*, pages 21–30. Association for Computing Machinery (ACM) Press, New York, NY, USA, October 2007. (5 citations on pages 2, 3, 21, and 46.)

[14] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID from Bilinear Pairing. Cryptology ePrint Archive, Report 2009/095, 2009. [Online] Available: `http://eprint.iacr.org/2009/095`. (5 citations on pages 2, 3, 21, and 46.)

[15] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to Win the Clone Wars: Efficient Periodic n-Times Anonymous Authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of CCS 2006*, pages 201–210. Association for Computing Machinery (ACM) Press, New York, NY, USA, November 2006. (One citation on page 14.)

[16] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology: Proceedings of EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer-Verlag, Berlin/Heidelberg, 2001. (4 citations on pages 17, 21, and 22.)

[17] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Proceedings of SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer-Verlag, Berlin/Heidelberg, 2002. (2 citations on page 22.)

[18] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In Matthew K. Franklin, editor, *Advances in Cryptology: Proceedings of CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer-Verlag, Berlin/Heidelberg, 2004. (One citation on page 22.)

[19] Jan Camenisch and Markus Michels. Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. In Jacques Stern, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer-Verlag, Berlin/Heidelberg, 1999. (3 citations on pages 27, 28, and 44.)

[20] Jan Camenisch and Victor Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In Dan Boneh, editor, *Advances in Cryptology: Proceedings of CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer-Verlag. 2003. (2 citations on pages 19 and 41.)

[21] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, Berlin/Heidelberg, 1997. (One citation on page 28.)

[22] David Chaum. Blind Signatures for Untraceable Payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO'82*, pages 199–203. Plenum Press, New York, NY, USA, 1982. (One citation on page 22.)

[23] David Chaum. Blind Signature System. In David Chaum, editor, *Advances in Cryptology: Proceedings of CRYPTO'83*, page 153. Plenum Press, New York, NY, USA, 1983. (One citation on page 22.)

[24] David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985. Association for Computing Machinery (ACM) Press, New York, NY, USA. ISSN 0001-0782. (One citation on page 16.)

[25] David Chaum and Jan-Hendrik Evertse. A Secure and Privacy-protecting Protocol for Transmitting Personal Information Between Organizations. In Andrew M. Odlyzko, editor, *Advances in Cryptology: Proceedings of CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 118–167. Springer-Verlag, Berlin/Heidelberg, 1986. (One citation on page 17.)

[26] David Chaum and Eugène van Heyst. Group Signatures. In Donald W. Davies, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, Berlin/Heidelberg, 1991. (One citation on page 18.)

[27] Lidong Chen. Access with Pseudonyms. In Ed Dawson and Jovan Dj. Golic, editors, *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, Berlin/Heidelberg, 1995. (One citation on page 17.)

[28] Ivan Damgård and Maciej Koprowski. Practical Threshold RSA Signatures without a Trusted Dealer. In Birgit Pfitzmann, editor, *Advances in Cryptology: Proceedings of EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 152–165. Springer-Verlag, Berlin/Heidelberg, 2001. (6 citations on pages 29, 30, 31, and 52.)

[29] Ivan Bjerre Damgård. Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals. In Shafi Goldwasser, editor, *Advances in Cryptology: Proceedings of CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 328–335. Springer-Verlag, Berlin/Heidelberg, 1988. (One citation on page 17.)

[30] S. Deering and R. Hinden. RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification, 1998. (One citation on page 19.)

[31] Roger Dingledine. Tor development roadmap, 2008–2011. Tor project technical report, The Tor Project Inc., December 2008. [Online] Available: `www.torproject.org/press/presskit/2008-12-19-roadmap-full.pdf`. (One citation on page 8.)

[32] Roger Dingledine. Personal communications, August 2010. (One citation on page 12.)

[33] Roger Dingledine and Nick Mathewson. Design of a blocking-resistant anonymity system. Tor project technical report, The Tor Project Inc., November 2006. [Online] Available: `https://svn.torproject.org/svn/projects/design-paper/blocking.html`. (One citation on page 33.)

[34] Roger Dingledine, Nick Mathewson, and Paul Syverson. Deploying Low-Latency Anonymity: Design Challenges and Social Factors. *IEEE Security and Privacy*, 5(5):83–87, 2007. IEEE Educational Activities Department, Piscataway, New Jersey, USA. ISSN 1540-7993. (One citation on page 8.)

[35] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of USENIX Security'04*, pages 303–320. The USENIX Association, Berkeley, California, USA, 2004. (2 citations on pages 1 and 32.)

[36] John R. Douceur. The Sybil Attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *Proceedings of IPTPS 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer-Verlag, Berlin/Heidelberg, 2002. (One citation on page 8.)

[37] R. Droms. RFC 2131 – Dynamic Host Configuration Protocol, 1997. (One citation on page 8.)

[38] Paul Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *Proceedings of FOCS 1987*, pages 427–437. The IEEE Computer Society. October 1987. (One citation on page 54.)

[39] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology: Proceedings of CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, Berlin/Heidelberg, 1986. (3 citations on pages 27, 38, and 56.)

[40] Pierre-Alain Fouque and Jacques Stern. Fully Distributed Threshold RSA under Standard Assumptions. In Colin Boyd, editor, *Advances in Cryptology: Proceedings of ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 310–330. Springer-Verlag, Berlin/Heidelberg, December 2001. (2 citations on pages 31 and 52.)

[41] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust Efficient Distributed RSA-Key Generation. In *Proceedings of STOC 1998*, pages 663–672. Association for Computing Machinery (ACM) Press, New York, NY, USA, 1998. (5 citations on pages 29 and 30.)

[42] Free Software Foundation. The GNU Multiple Precision (GMP) Arithmetic Library, Version 5.0.1, February 2010. [Online] Available: `http://gmplib.org/`. (One citation on page 51.)

[43] Geeknet Inc. Slashdot – News for nerds, stuff that matters. [Online] Available: http://slashdot.org/, Accessed: 11/08/2010. (One citation on page 2.)

[44] Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, University of California at Berkeley, Berkeley, California, USA, December 2000. (One citation on page 5.)

[45] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing (SICOMP)*, 17 (2):281–308, April 1988. Society for Industrial and Applied Mathematics (SIAM). ISSN 0097-5397. (One citation on page 28.)

[46] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(TR07-053), June 2007. ISSN 1433-809. Also available as [47]. (One citation on page 80.)

[47] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In Nigel P. Smart, editor, *Advances in Cryptology: Proceedings of EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, Berlin/Heidelberg, 2008. Also available as [46]. (2 citations on pages 22 and 80.)

[48] Ryan Henry. Pippenger's Multiproduct and Multiexponentiation Algorithms. Technical Report CACR 2010-26, University of Waterloo, Centre for Applied Cryptographic Research, Waterloo, Ontario, Canada, September 2010. [Online] Available: http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-26.pdf. (One citation on page 53.)

[49] Ryan Henry and Ian Goldberg. A Survey of Anonymous Blacklisting Systems. Technical Report CACR 2010-24, University of Waterloo, Centre for Applied Cryptographic Research, Waterloo, Ontario, Canada, September 2010. [Online] Available: http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-24.pdf. (One citation on page 8.)

[50] Ryan Henry, Kevin Henry, and Ian Goldberg. Making a Nymbler Nymble using VERBS. In Mikhail Atallah and Nick Hopper, editors, *Proceedings of PETS 2010*, volume 6205 of *Lecture Notes in Computer Science*, pages 110–129. Springer-Verlag, Berlin/Heidelberg, July 2010. An extended version of this paper is available [51]. (4 citations on pages 3, 18, 46, and 81.)

[51] Ryan Henry, Kevin Henry, and Ian Goldberg. Making a Nymbler Nymble using VERBS (Extended Version). Technical Report CACR 2010-05, University of Waterloo, Centre for Applied Cryptographic Research, Waterloo, Ontario, Canada, March 2010. 24 pages. [Online] Available: http://www.cacr.math.uwaterloo.ca/techreports/

2010/cacr2010-05.pdf. This is the extended version of [50]. (4 citations on pages 3, 18, 63, and 80.)

[52] Jason E. Holt and Kent E. Seamons. Nym: Practical Pseudonymity for Anonymous Networks. Technical Report 2006-4, Brigham Young University, Internet Security Research Lab, Provo, Utah, USA, June 2006. 12 pages. [Online] Available: http://isrl.cs.byu.edu/pubs/isrl-techreport-2006-4.pdf. (One citation on page 17.)

[53] Human Rights Watch. "Race to the Bottom": Corporate Complicity in Chinese Internet Censorship: II. How Censorship Works in China: A Brief Overview, March 2009. [Online] Available: http://www.hrw.org/reports/2006/china0806/3.htm, Accessed: 12/13/2010. (One citation on page 33.)

[54] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Nymble: Anonymous IP-Address Blocking. In Nikita Borisov and Philippe Golle, editors, *Proceedings of PETS 2007*, volume 4776 of *Lecture Notes in Computer Science*, pages 113–133. Springer-Verlag, Berlin/Heidelberg, June 2007. A work-in-progress version of this paper is available [90]. (7 citations on pages 2, 3, 4, 14, 18, 67, and 85.)

[55] Stefan Köpsell, Rolf Wendolsky, and Hannes Federrath. Revocable Anonymity. In Günter Müller, editor, *Proceedings of ETRICS 2006*, volume 3995 of *Lecture Notes in Computer Science*, pages 206–220. Springer-Verlag, Berlin/Heidelberg, 2006. (One citation on page 2.)

[56] Hendrik W. Lenstra. Factoring integers with elliptic curves. *The Annals of Mathematics*, 126(3):649–673, November 1987. (One citation on page 58.)

[57] Zi Lin and Nick Hopper. Jack: Scalable Accumulator-based Nymble System. In Keith Frikken, editor, *Proceedings of WPES 2010*, pages 53–62. Association for Computing Machinery (ACM) Press, New York, NY, USA, October 2010. (9 citations on pages 2, 3, 14, 15, 18, 19, 20, and 41.)

[58] Karsten Loesing. Measuring the Tor Network: Evaluation of Client Requests to the Directories. Technical report, The Tor Project, June 2009. [Online] Available: https://metrics.torproject.org/papers/directory-requests-2009-06-25.pdf. (One citation on page 1.)

[59] Peter Lofgren and Nicholas Hopper. BNymble (A short paper): More anonymous blacklisting at almost no cost. In *Proceedings of the Fifteenth International Conference on Financial Cryptography and Data Security*, 2011. to appear. (One citation on page 18.)

[60] Anna Lysyanskaya. Pseudonym Systems. Master's thesis, Massachusetts Institute of Technology (MIT), Department of Electrical Engineering and Computer Science, Cambridge, Massachusetts, USA, June 1999. (One citation on page 17.)

[61] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocols*. PhD thesis, Massachusetts Institute of Technology (MIT), Department of Electrical Engineering and Computer Science, Cambridge, Massachusetts, USA, September 2002. (One citation on page 28.)

[62] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. In Howard M. Heys and Carlisle M. Adams, editors, *Proceedings of SAC'99*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer-Verlag, Berlin/Heidelberg, 1999. (One citation on page 17.)

[63] Ueli M. Maurer and Yacov Yacobi. A non-interactive public-key distribution system. *Designs, Codes and Cryptography*, 9(3):305–316, 1996. (One citation on page 59.)

[64] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN 978-0-8493-8523-0. Fifth Printing (August 2001). [Online] Available: `http://www.cacr.math.uwaterloo.ca/hac/`. (2 citations on pages 29 and 37.)

[65] NVIDIA. CUDA Toolkit 3.2 RC 2, October 2010. [Online] Available: `http://developer.nvidia.com/object/cuda_3_2_toolkit_rc.html`. (One citation on page 51.)

[66] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology: Proceedings of CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, Berlin/Heidelberg, 1991. (One citation on page 27.)

[67] Kun Peng, Colin Boyd, and Ed Dawson. Batch Zero-Knowledge Proof and Verification and Its Applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(2), Article No. 6, May 2007. Association for Computing Machinery (ACM) Press, New York, NY, USA. ISSN 1094-9224. (One citation on page 53.)

[68] John M. Pollard. Theorems on factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*, 76(03):521, 1974. (One citation on page 58.)

[69] Carl Pomerance. The quadratic sieve factoring algorithm. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *Advances in Cryptology: Proceedings of EUROCRYPT 1984*, volume 209 of *Lecture Notes in Computer Science*, pages 169–182. Springer-Verlag, Berlin/Heidelberg, 1984. (One citation on page 58.)

[70] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. RFC 1918 – Address Allocation for Private Internets, 1996. (One citation on page 8.)

[71] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology: Proceedings of CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, Berlin/Heidelberg, 1989. (One citation on page 27.)

[72] Edward J. Schwartz. Contractual Anonymity. Master's thesis, Carnegie Mellon University, The Information Networking Institute, Pittsburgh, Pennsylvania, USA, May 2009. (2 citations on pages 13 and 18.)

[73] Edward J. Schwartz, David Brumley, and Jonathan M. McCune. Contractual Anonymity. Technical Report CMU-CS-09-144, Carnegie Melon University, School of Computer Science, Pittsburgh, Pennsylvania, USA, September 2009. 29 pages. [Online] Available: `http://reports-archive.adm.cs.cmu.edu/anon/2009/abstracts/09-144.html`. This is the extended version of [74]. (3 citations on pages 13, 18, and 83.)

[74] Edward J. Schwartz, David Brumley, and Jonathan M. McCune. A Contractual Anonymity System. In *Proceedings of NDSS 2010*. The Internet Society (ISOC). February 2010. An extended version of this paper is available [73]. (3 citations on pages 13, 18, and 83.)

[75] Victor Shoup. Practical Threshold Signatures. In Bart Preneel, editor, *Advances in Cryptology: Proceedings of EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer-Verlag, Berlin/Heidelberg, 2000. (One citation on page 31.)

[76] Victor Shoup. NTL: A Library for doing Number Theory, Version 5.5.2, August 2009. [Online] Available: `http://www.shoup.net/ntl/`. (One citation on page 51.)

[77] Stuart G. Stubblebine, Paul F. Syverson, and David M. Goldschlag. Unlinkable Serial Transactions: Protocols and Applications. *ACM Transactions on Information and System Security (TISSEC)*, 2(4):354–389, November 1999. Association for Computing Machinery (ACM) Press, New York, NY, USA. This is the journal version of [78]. (2 citations on pages 18 and 83.)

[78] Paul F. Syverson, Stuart G. Stubblebine, and David M. Goldschlag. Unlinkable Serial Transactions. In Rafael Hirschfeld, editor, *Proceedings of FC '97*, volume 1318 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag, Berlin/Heidelberg, February 1997. A journal version of this paper is available [77]. (2 citations on pages 18 and 83.)

[79] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-Times Anonymous Authentication (Extended Abstract). In Pil Joong Lee, editor, *Advances in Cryptology: Proceedings of ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, Berlin/Heidelberg, 2004. (One citation on page 14.)

[80] The Tor Project Inc. TheOnionRouter/BlockingIrc – Tor Bug Tracker & Wiki, November 2010. [Online] Available: `https://trac.torproject.org/projects/tor/wiki/TheOnionRouter/BlockingIrc`, Accessed: 11/08/2010. (One citation on page 2.)

[81] The Tor Project Inc. TorStatus - Tor Network Status, November 2010. [Online] Available: `http://torstatus.cyberphunk.org/`, Accessed: 11/08/2010. (One citation on page 1.)

[82] The Tor Project Inc. Who uses Tor?, November 2010. [Online] Available: `http://www.torproject.org/about/torusers.html.en`, Accessed: 11/08/2010. (One citation on page 1.)

[83] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable Anonymous Credentials: Blocking Misbehaving Users Without TTPs. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of CCS 2007*, pages 72–81. Association for Computing Machinery (ACM) Press, New York, NY, USA, October 2007. An extended version of this paper is available [84]. (9 citations on pages 2, 3, 14, 15, 21, 46, and 84.)

[84] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable Anonymous Credentials: Blocking Misbehaving Users Without TTPs (Extended Version). Technical Report TR2007-601, Dartmouth College, Computer Science Department, September 2007. 25 pages. [Online] Available: `http://www.cs.dartmouth.edu/reports/abstracts/TR2007-601/`. This is the extended version of [83]. (8 citations on pages 2, 3, 14, 15, 21, 46, and 84.)

[85] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. BLAC: Revoking Repeatedly Misbehaving Anonymous Users Without Relying on TTPs. Technical Report TR2008-635, Dartmouth College, Computer Science Department, Hanover, New Hampshire, USA, October 2008. 33 pages. [Online] Available: `http://www.cs.dartmouth.edu/reports/abstracts/TR2008-635/`. A journal version of this paper exists [87]. (8 citations on pages 2, 3, 14, 15, 21, 46, and 85.)

[86] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. PEREA: Towards Practical TTP-free Revocation in Anonymous Authentication. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of CCS 2008*, pages 333–344. Association for Computing

Machinery (ACM) Press, New York, NY, USA, October 2008. (9 citations on pages 2, 3, 14, 22, 46, and 60.)

[87] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. BLAC: Revoking Repeatedly Misbehaving Anonymous Users Without Relying on TTPs. *ACM Transactions on Information and System Security (TISSEC)*, 2010. Association for Computing Machinery (ACM) Press, New York, NY, USA. To appear. This is the journal version of [85]. (4 citations on pages 3, 46, and 84.)

[88] Patrick P. Tsang, Apu Kapadia, Cory Cornelius, and Sean W. Smith. Nymble: Blocking Misbehaving Users in Anonymizing Networks. Technical Report TR2008-637, Dartmouth College, Computer Science Department, Hanover, New Hampshire, USA, December 2008. 41 pages. [Online] Available: `http://www.cs.dartmouth.edu/reports/abstracts/TR2008-637/`. This is the extended version of [89]. (6 citations on pages 2, 3, 14, 18, 46, and 85.)

[89] Patrick P. Tsang, Apu Kapadia, Cory Cornelius, and Sean W. Smith. Nymble: Blocking Misbehaving Users in Anonymizing Networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, September 2009. The IEEE Computer Society. To appear. An extended version of this paper is available [88]. (10 citations on pages 2, 3, 14, 18, 21, 25, 43, 46, and 85.)

[90] Patrick P. Tsang, Apu Kapadia, and Sean W. Smith. Anonymous IP-address Blocking in Tor with Trusted Computing (Short Paper: Work in Progress). In *In the Second Workshop on Advances in Trusted Computing (WATC 2006 Fall), 30 November – 1 December 2006, Tokyo, Japan*, 2006. This is a work-in-progress version of [54]. (3 citations on pages 2, 3, and 81.)

[91] Paul C. van Oorschot and Michael J. Wiener. Parallel Collision Search with Application to Hash Functions and Discrete Logarithms. In *ACM Conference on Computer and Communications Security*, pages 210–218, 1994. (2 citations on pages 44 and 57.)

[92] Wikimedia Foundation. [Online] Available: `http://www.wikimedia.org/`, Accessed: 11/08/2010. (One citation on page 15.)

[93] Wikimedia Foundation. Wikipedia, the free encyclopedia. [Online] Available: `http://www.wikipedia.org/`, Accessed: 11/08/2010. (3 citations on pages 2, 15, and 17.)

[94] Wikimedia Foundation. Wikipedia:Blocking policy — Wikipedia, the free encyclopedia. [Online] Available: `http://en.wikipedia.org/wiki/Wikipedia:Blocking_policy`, Accessed: 12/10/2010. (5 citations on pages 4, 15, 19, 46, and 67.)

[95] Wikimedia Foundation. Wikiquote. [Online] Available: `http://www.wikiquote.org/`, Accessed: 11/08/2010. (One citation on page 15.)

[96] Wikimedia Foundation. Wiktionary. [Online] Available: `http://www.wiktionary.org/`, Accessed: 11/08/2010. (One citation on page 15.)