On Combating Adverse Selection in Anonymity Networks

by

Jeremy Clark

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

Anonymity networks allow users to hide their Internet Protocol (IP) address while performing actions online; a service that enhances internet privacy but also allows unlawful actions to be committed anonymously. This thesis proposes methods for combating adverse selection, and its effects, in anonymity networks. We first consider the problem of servers in an anonymity network being confiscated or held legally liable as a result of unlawful data being sent through them. We propose two cryptographic protocols that solve this problem by providing proof that the anonymous data originated from a different IP address than the server. We propose another cryptographic protocol that allows users to be anonymously and provably banned from an anonymity network if they conduct themselves in an unlawful manner. Finally we identify usability issues with the deployability of Tor, a popular anonymity network, by conducting a formal cognitive walkthough and propose improvements to make Tor more accessible.

Acknowledgements

I would like to thank foremost my adviser, Carlisle Adams, for guidance and inspiration of many ideas contained herein. I am grateful for the contributions of Paul Van Oorschot to [33], which is the basis of Chapter 5. I would like to acknowledge my colleague Aleks Essex for the many late-night blackboard sessions that helped flesh out both ideas presented here and in our other publications on cryptographic voting [30, 29, 21, 40]. I have loving appreciation for Neha Chugh and her support throughout my studies. And finally, I would like to acknowledge Wail Gueaieb and Stephen M. Carr for creation of the LaTeX template used here to comply with the University of Ottawa's manuscript requirements.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Motivating Problem

In early September 2006, German law enforcement conducted a sting operation to thwart the distribution of child pornography, resulting in several servers being confiscated [58]. Some of these servers were part of an anonymity network, called Tor, which anonymises internet traffic on behalf of its users by laundering it through a chain of specialized proxy servers. By having the sanitized traffic leaving the network appear to have originated from the final server in the chain, and not from the original sender, anonymity can be achieved. However this process could potentially leave the server liable for data it did not originate. While no charges to date have been laid against the operators of these servers, the legal consequences of forwarding illicit data remain ambiguous with respect to many countries' laws. As well, the mere threat of confiscation creates a strong negative incentive against volunteering to operate an exit server in an anonymity network like Tor.

This issue provides a glimpse into the dark side of anonymity. Criminals or mischievous users have a strong incentive to remain anonymous, and we expect that they are likely to surmount the configuration difficulties and bandwidth latencies which plague anonymity networks like Tor, problems that could preclude honest citizens from using it. When a service is most attractive to the individuals it does not want to attract, this situation is referred to as adverse selection.[1] In their current form, anonymity networks

[1] As an example of adverse selection, consider medical insurance [15]. An individual with a family history of hereditary disease, who engages in risky behaviour, is nearing old-age, or knows she is terminally ill will have a high incentive to pay for insurance. If the company has no way of distinguishing these individuals, insurance policies will be issued to individuals with a probability of needing to make

have a natural inclination toward adverse selection. This thesis will examine measures that can be implemented to attract honest users and server operators, while deterring misbehaving or criminal users. Specifically, we propose technical improvements to exonerate innocent servers and ban malicious users, and recommendations for improving the usability of configuration to attract a wider user base.

The salience of the issues we will examine is in how they are situated at the intersection of a variety of disciplines. For example, a seemly simple solution to the problem of servers being confiscated would be for the final node to reveal who it received the data from, and law enforcement could iteratively trace the data back to the original sender. There are three main problems with the viability of this solution. The first is a technical problem. Providing the ability to revoke a user's anonymity would require the volunteer servers to store logs, which would amount to storing large amounts of data. Furthermore, server logs have no inherent integrity and could be modified or forged. Implementing a secure, unforgeable method of storing the data would be a large computational burden. The second problem is the legal issue of jurisdiction. Most anonymity networks stretch, deliberately in the case of Tor, across different countries and continents, which would require an international effort to subpoena the data required to trace a message. The third and final problem is the political viability of requiring traceability. For example, when the US Government introduced a cryptographic chip for voice transmission in the early 1990s, the protocol included holding a decryption key in escrow that could be accessed by law enforcement officials if authorized. The political backlash against the technology ensured it was never adopted, and its foreseeable that compelling anonymity networks to provide traceability would face a similar political resistance.

There are also ethical considerations. Most individuals strongly condemn, for example, the creation and distribution of child pornography and would prefer measures to be taken to catch these criminals; the author of this thesis is no exception. However there could also be the competing concern that provisions created to facilitate the prosecution of the most heinous criminals will be gradually relaxed until they are used for civil actions, like copyright infringement [53], instead of criminal actions; or alternatively that they are used for anti-democratic purposes, like the restriction of free speech, in prohibitive nation-states. Online anonymity also offers privacy protection for

claims in excess of their premiums. In this simplified model, the insurance provider will have to raise prices to compensate. A price rise will decrease demand for insurance, and the provider will lose low-risk customers with the least suspicion of needing insurance, creating a negative feedback loop.

whistle-blowers, abused women, soldiers, individuals seeking information that might be embarrassing, *etc.* It has been argued that honest citizens have 'nothing to hide.' These examples illustrate otherwise. We also refer the reader to the extensive critique of the 'nothing to hide' argument in [66].

The ethical position of this thesis is neutral. Our approach will be to preserve the status quo of providing unrevocable anonymity for all users, while affording provisions to ease the legal threat facing anonymity networks. Each contribution is engineered to either create a positive incentive for honest users and operators, or to create a negative incentive for criminal users. The totality of our contributions is intended to change the inherent incentive structure in current anonymity networks. We do not purport to have completely solved the adverse selection issue but our research offers significant improvements.

1.2 Privacy as a Subset of Security

In the legal arena, privacy is commonly conceived of as the right to be left alone and is an important protected liberty in many Western democracies. A concise definition of privacy offered by the courts in the United States is the right to 'live life free from unwarranted publicity' [3]. In Canada, the *Privacy Act* serves the purpose of extending 'the present laws of Canada that protect the privacy of individuals and that provide individuals with a right of access to personal information about themselves' [5]. These public sector protections are augmented with private sector protections, in the *Personal Information Protection and Electronic Documents Act (PIPEDA)*, to 'support and promote electronic commerce by protecting personal information that is collected, used or disclosed in certain circumstances, by providing for the use of electronic means to communicate or record information or transactions' [4]. Oversight of citizen privacy in Canada is mandated to the independent Office of the Privacy Commissioner of Canada and the corresponding provincial privacy commissioners.

The Privacy Act presupposes a definition of privacy in its summary, as quoted above, while PIPEDA attempts to describe its purpose without reference to any specific notion of privacy (indeed, the word does not appear in its title), opting instead to address the more concrete concept of 'personal information protection.' In this light, privacy affords control to a person over how her personal information is disseminated, both in transactions involving her own participation and *ex post* transactions involving those with

whom she has disclosed private information. We refer to the control over dissemination of personal information as informational privacy, and it is the guiding definition of privacy for this thesis.

In the field of information security and cryptography, there is no universal consensus on the relationship between privacy and security. One common view considers them related but distinct. Often cited as evidence for this position is the importance placed upon non-repudiation in the framework of security. Non-repudiation is 'a service that prevents an entity from denying previous commitments or actions' [54], and it is clearly antithetical to privacy. However non-repudiation is not considered an objective of security *per se*, but rather a means to some other security objective (*e.g.* authentication or authorization). Thus its value in relation to information security is instrumental and not intrinsic. However when privacy is thought of as the right to keep private information confidential, it clearly aligns itself as a security objective. Thus we argue that informational privacy is a subset of security.

1.3 A Threat Model for Privacy

Like privacy, security is difficult to define and for the same reason. This reason is that both are *via negativa* concepts. That is to say, security claims take the form of negative statements instead of positive statements. For example, security claims about a piece of software may include the following: the software is protected against buffer overflows, the handling of strings prevents SQL injection attacks, measures have been taken to thwart decompilation, *etc.* These are all assertions about the way the software does *not* behave; namely, it is not susceptible to known attack vectors. But because there is always the possibility of unknown vectors of attack, it is fallicious to positively state that a piece of software is secure. Security is falsifiable: we can positively claim the insecurity of something. However, at best, we are confident but technically agnostic in our claims of positive security.

Similarly, the privacy claims in preceding paragraphs focused on negative assertions like being free *from* intervention and publicity. Its our view that the best way to approach privacy is by defining what it protects against; that is, defining and categorizing the threats to privacy instead of the properties of privacy. This is the approach taken by Solove who first bemoans the ambiguity of privacy, calling it a 'concept in disarray' the meaning of which 'nobody can articulate,' and then proceeds to create a taxonomy of

privacy threats [65]. His taxonomy has four categories of threats: invasion, information collection, information processing, and information dissemination.

The latter three categories are concerned with information the subject voluntarily discloses under some expectation of privacy, in contrast to the former which addresses threats to information the subject has not disclosed. This first category, **invasion**, includes intrusions, incursions, or purposeful interference in the subject's private sphere. In information security, this represents active attacks against a subject and may include the use of exploits to gain control of a computer or the physical theft of data. These represent an important set of threats but are largely outside the subject of this work. Rather, the focus here is on addressing the latter three threats.

Consider the situation where Alice voluntarily discloses a piece of private information to Bob. **Information collection** includes the threat of eavesdropping, where the adversary Eve listens in on the communication channel and thus learns the private information. Solove also includes interrogation under information collection; its our belief that interrogation would be better relegated to the invasion category. The **information processing** category deals with privacy threats surrounding the security of the private information once it is in Bob's possession. For example, Bob can attempt to identify Alice if she has not disclosed her identity. He can aggregate information about Alice, connecting disparate pieces of information through data-mining and merging archived information that was collected for other purposes. He could also be careless about how the information is secured from external adversaries. **Information dissemination** threats concern Bob's disclosure of the private information to third parties. Bob could breech the confidentiality agreement he has with Alice or could simply threaten to do so through blackmail.

Many of these threats have been examined from the perspective of information security. Cryptography can be used to secure a communication channel from eavesdroppers. It can also be used to protect private data stored on a drive, usually in combination with access control policies to ensure that private data is available only to those with the proper credentials. Cryptography can also be used to make forgeable messages: Alice can disclose information to Bob in a trusted and authenticated manner but Bob is unable to prove to Eve that the information was from Alice and not created by himself or forged (*e.g.*, OTR [18]). Other researchers have focused on improving how online privacy policies are communicated to users (*e.g.*, P3P [34]) and how they can be enforced by third party certification agencies like TRUSTe (which have been shown to be currently

unsuccessful and offering an adverse selection of sites [39]).

1.4 Anonymity, Pseudonymity, and Veronymity

A specific form of privacy is anonymity. Anonymity can mean different things in different contexts, but it is generally considered to be the 'state of namelessness' [50]. Namelessness implies moving throughout society without an identifier—nothing to tie your actions to your identity. Information with the potential of linking actions to an individual is referred to as personally identifiable information (PII). Anonymity could thus be thought as performing actions without disclosing any PII.

Formally, anonymity requires two necessary conditions,

P1: the action is not linkable to the identity of the actor, and

P2: two actions performed by the same actor are not linkable to each other.

Note that **P2** implies **P1**,

$$\neg\mathbf{P1} \rightarrow \neg\mathbf{P2} \tag{1.1}$$

$$\mathbf{P2} \wedge (\neg\mathbf{P1} \rightarrow \neg\mathbf{P2}) \rightarrow \mathbf{P1} \quad (\textit{modus tollens}) \tag{1.2}$$

$$\vdash \mathbf{P2} \rightarrow \mathbf{P1} \tag{1.3}$$

If the proposition **P1** is false, actions are associated with the actor's identifier, and the identifier is considered 'veronymous' (a Latin *portmanteau* for 'true name' [12]). In this case, two disparate actions performed by the same actor would both be linked to the actor's identity and are thereby linkable to each other. This implies that proposition **P2** is false whenever **P1** is. Inversely, if proposition **P2** is true, then both actions cannot be linked to the actor's identity, rendering **P1** true with respect to at least one of the actions. Thus **P2** implies **P1**, and thereby **P2** is both a necessary and sufficient condition for anonymity.

If **P1** is true and **P2** is false, then actions can be linked to a common identifier that is not the actor's true identity. This is referred to as 'pseudonymity' ('alternate name'). **P1** is a necessary condition for pseudonymity. Table 1.1 summarizes the relationship between the two propositions and the concepts of anonymity, pseudonymity, and veronymity. We now consider these concepts in terms of the online world.

	P1 - True	P1 - False
P2 - True	Anonymity	*(Impossible)*
P2 - False	Pseudonymity	Veronymity

Table 1.1: Summary of Relationship between Propositions 1 and 2.

1.5 Anonymous Browsing

In the online world, pseudonymous identifiers are pervasive. A *self-volunteered* identifier is a digital pseudonym used to access features on a web service (i.e., a screen-name, user-name, or email address). A *server-assigned* identifier is a unique identifier used by a web service to monitor their users (*i.e.*, a cookie or spyware). The anonymity afforded by anonymity networks like Tor does not extend to either of these categories of identifiers. Rather, it deals with *protocol-based* identifiers; specifically internet protocol (IP) addresses. When a device is online, it is reachable through its unique IP address. An IP address does not necessarily correspond to a single computer; it could, for example, identify the gateway to a network of computers. At best, IP addresses tie actions from this device together; and therefore could be pseudonymous. However if a single user of an IP address is revealed, then the IP address becomes a veronymous identifier.

There are practical reasons to be concerned about the privacy of an IP address. An internet service provider (ISP) can link its addresses to the owner of the account paying for it—information that is obtainable by others for a jurisdictional variety of reasons. Also, based on the fact that IP addresses are distributed by ISPs, it is possible to determine which ISP an IP address belongs to, and thus determine a general locale for a user based on their IP address alone. This reduces privacy even when the IP address is pseudonymous. Another privacy concern facilitated by IP addresses is the ability of a web service to link distinct actions by the same user together. Thus when a user accesses a web service repeatedly, the web service has the potential to link all of these actions together. With the collaboration of other web services, the actions of the user on other sites can be linked in (failed internet companies are often purchased for their database of user information). Should the user reveal her true identity at any point, such as by making a transaction or logging in, then all past and future actions with the same IP address can be linked to her identity. Even if the user does not reveal her full identity,

an IP address can be augmented with other categories of pseudonymous identifiers or PII (*i.e.*, a search query for a relative or the revelation of a postal code). Aggregating enough information can be used to reduce the user's privacy and possibly uncover her true identity. Data-mining and geo-location are examples of this privacy threat [60, 55]. Anonymity networks, like Tor, unlink a user's actions from her IP address.

1.6 Summary of Contributions

The first contribution of this thesis is to provide a technical means for exit nodes in a Tor network to prove that the data they release originated from an internet protocol address other than their own. As a design goal, the anonymity of the user (*i.e.*, the true originator of the data) is preserved. As a result, this proof must be structured to not reveal the originating IP address itself, only that it satisfies an inequality equation with the server's IP address. The proximate cause of this contribution is to alleviate the legal risk of operating a server, and the ultimate result is that it should increase the ease of recruiting server operators in Tor and other anonymity networks. A side benefit of an increase in server operators is an increase in the throughput of the network.

Our second contribution is to provide a technical means for law enforcement to have users banned from using Tor and other anonymity networks. Once again, a design goal of this contribution is to preserve the anonymity of the user. This appears to be an unsolvable problem, as anonymity precludes the ability to identify a user to be banned; however, using distributed trust and a system of pseudonyms, this can be accomplished with the participation of a trusted third party. The ultimate result of this contribution will be the prevention of sustained criminal activity through the network originating from the same IP address. While this does not eliminate all criminal activity, it affords law enforcement some measure of access control. The maintainers of Tor can also use this architecture to ban users who do not comply with their terms of service. Both the first and second contributions make use of digital credentials, developed by Stefan Brands [19], but the application of digital credentials to anonymity networks and the protocols that deploy them are original contributions [31, 32].

Our final contribution is the first extensive usability study of Tor to be presented in the literature (prior to [33]). We compile a set of Tor-relevant usability evaluation guidelines from a variety of sources, eliminate the redundancies, and offer justifications—in some cases, based on research in cognitive psychology not yet applied to usable security

and privacy. Our guidelines build on the earlier guidelines proposed to date, including Whitten and Tygar [71] and others, however our guidelines are appropriately shifted in focus from usable security to usable privacy. Using our guidelines, we perform a cognitive walkthrough of the core tasks of installing, configuring, and running Tor. We examine manually configuring Firefox for use with Tor, Privoxy (a filtering proxy) [6], and Vidalia (a GUI for Tor) [10]. We also examine two Firefox extensions, Torbutton [8] and FoxyProxy [2], designed to assist the user in performing the key tasks. And finally we inspect Torpark [9]—a standalone Firefox variant with built-in Tor support. We uncover numerous usability issues with each deployment option and offer suggestions for improvement (some of which have been adapted since the original publication of this work [33]).

1.7 Organization of Thesis

In Chapter 2, we review some preliminaries of cryptography and provide an overview of the literature on anonymity networks. In particular, we trace the evolution of anonymous web-browsing from mix networks to onion routing and ultimately to Tor. We also introduce digital credentials and review the usable security literature for usability guidelines. In Chapter 3, we present our solution for exonerating server nodes from liability for illicit data. In Chapter 4, we propose a method for banning users from Tor. Chapter 5 examines the usability of Tor. Finally, Chapter 6 will offer some concluding remarks and propose directions for future work.

Chapter 2

An Overview of Anonymous Web-browsing

2.1 Introduction

This chapter will provide an overview of the literature that this work builds on. We will begin with the cryptographic primitives that will be utilized in subsequent chapters. We then consider the literature on the anonymous communication subset of the field of privacy enhancing technologies. We begin with David Chaum's seminal paper introducing the topic of mix networks. The mix network forms the basis of nearly every anonymous communications technology subsequently proposed. We will then examine onion routing, an extension of mix networks for web browsing and other applications. We finish this thread of discussion with Tor, a self-proclaimed "second-generation onion router" and the topic of the usability study in Chapter 5. We also introduce the concepts of digital credentials, and provide an overview of usability techniques.

2.2 Cryptographic Primitives

In this section, we define several cryptographic primitives that will be used in the subsequent chapters. The intention here is to be concise and to define notation. For full and nuanced descriptions of these primitives, we refer the reader to the first two chapters of the *Handbook of Applied Cryptography* [54].

2.2.1 Encryption and Decryption Functions

Encryption function E maps plaintext message m from the message space $\mathcal{M}$ into a ciphertext c from the ciphertext space $\mathcal{C}$ using encryption key e from the keyspace $\mathcal{K}$: $E_e(m) = c$. Decryption function D inverts this using decryption key d: $D_d(c) = m$. When $e \neq d$, it is referred to as asymmetric or public key cryptography. In this case, given a full specification of the encryption and decryption functions and e, it should be computationally infeasible to calculate d. For this reason, e can be published in a public directory as long as d is retained privately. We refer to such keys as the public and private key respectively. In the case where $e = d$, this symmetric key will be denoted k. Given many pairs of m and c, all encrypted with the same k, and the full specification of the encryption and decryption functions, it should be computationally infeasible to calculate k.

2.2.2 Hash Functions and MACs

A cryptographic hash function $H()$ maps a message x from a message space of arbitrary size into hash-value y from the hashspace $\mathcal{H}$: $H(x) = y$. Given y and a full specification of the hash function, it should be computationally infeasible to find any x such that $H(x) = y$. Furthermore it should be computationally infeasible to find any two preimages x_1 and $x_2 \neq x_1$ that hash to the same y: $H(x_1) = H(x_2) = y$. Future references to cryptographically secure hash functions presume the hash function has these two properties. Hash functions do not depend on a key, however a Message Authentication Code or MAC is a function that depends on a secret key k with the same properties as a hash: $MAC_k(x) = y$. A MAC differs from an encryption/decryption function in that a MAC cannot be inverted even with the knowledge of the key.

2.2.3 Number Theoretic Notation

The set $\mathbb{Z}_n$ denotes the set of integers from 0 to $n-1$: $\{0, 1, \ldots, n-1\}$. The set $\mathbb{Z}_n^*$ is the subset of integers from 0 to $n-1$ that are relatively prime to n: $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | gcd(a, n) = 1\}$, where $gcd(a, n)$ is the greatest common denominator between a and n. In the case when n is prime number p, the subset is the set of integers from 1 to $n-1$: $\mathbb{Z}_p^* = \{1, 2, \ldots, n-1\}$. Integer g is a generator or primitive root of Z_p^* if $\left\{g^x \mod p | x \in Z_p^*\right\} = Z_p^*$. That is to say that the set $\{a^1, a^2, \ldots, a^{p-1}\}$ has no repeating

elements for $a \in Z_p^*$.

2.2.4 Inverses and Discrete Logarithms

If $y = a \cdot x \pmod{p}$ then the inverse of x is denoted x^{-1} and is such that $a \equiv y \cdot x^{-1}$ and $x \cdot x^{-1} \equiv 1$. Calculating the inverse of an element is computationally feasible and can be calculated efficiently with the extended Euclidean algorithm. Note that the inverse of an element does not necessarily exist for all elements in $\mathbb{Z}_n^*$ when n is non-prime. By limiting ourselves to only prime n, we can ensure that an arbitrary element is invertible and that a generator in the set exists. These properties will be useful in constructing digital credentials.

If $y = g^x \pmod{p}$, where g is a generator in Z_p^*, then the discrete log of y to base g is $\log_g y \equiv x$. Calculating the discrete logarithm of an element to a base is computationally infeasible for sufficiently large p (*e.g.*, 1024 bits). This is referred to as the discrete logarithm problem. A variation is that given g^a and g^b, it is infeasible to calculate $g^{a \cdot b}$ (all mod p). In fact, if given g^a and g^b but not a or b, it should be computationally infeasible to distinguish $g^{a \cdot b}$ from random element in Z_p^*. This latter problem is known as the Decisional Diffie-Hellman (DDH) problem, and will form the basis of the security proofs in Chapter 3.

2.3 Online Anonymity

In this section, we present an overview of several privacy enhancing technologies that can be employed to provide online anonymity. Recall that by online anonymity, we refer to hiding protocol level identifiers; namely IP addresses. These technologies do not purport to anonymise the content of a message, should the content refer to the sender's identity or an identifier associated with the sender, nor do they prevent web services from installing tracking cookies or spyware on a sender's computer. For an extensive overview of the field, we refer those interested to the set of papers hosted on the Anonymity Bibliography [1].

2.3.1 Proxy Servers

An intuitive method for a sender to achieve anonymity with respect to her IP address is to simply remove her IP address from all the TCP/IP packets and replace it with

a random address. This is known as header forgery or IP spoofing. This can be an effective method if the user does not want a response from the server, since the response will be routed to the forged IP address, however this eliminates the vast majority of online activities a user may wish to participate in. Even sending an email, a seemingly one-way task, requires the sender to exchange several messages with the server as part of the STMP connection protocol before sending the email data itself. There are a small number of connectionless protocols, the most popular is UDP, but they are typically employed to send data from a web service to the user and not vice-versa.

The most effective approach to controlling the dissemination of IP addresses online is through the use of a proxy. A proxy is someone who acts on another entity's behalf. For example, in voting, if an individual is unable to visit a polling station, they could be permitted to send a representative to vote on their behalf—a process called 'voting by proxy.' In a similar way, web services can be accessed by proxy. A proxy server will forward data from the user to the web service, and when the web service responds, it will forward the returning traffic to the user. The web service only sees the IP address of the proxy, preserving the anonymity of the user from the web service. This is a improvement over IP spoofing because it allows for two-way communication.

A proxy server effectively segments the connection between the user and the web service into the user-proxy link and the proxy-service link. To an eavesdropper listening in on the proxy-service link (the service itself is such an eavesdropper, and most services keep logs), it will appear that the proxy is communicating with service. An eavesdropper on the user-proxy link (such as the user's ISP, which can monitor and log the activities of its customers) will see the user interacting with the proxy server. However if the eavesdropper were to open up the packets, the outbound packets will contain instructions for the proxy server about the final destination and the content of the inbound packets will likely betray the final destination as well. And so a simple proxy server protects the user's identity from the web service, but it does not protect it from her ISP or any other eavesdropper watching her actions.

An easy improvement to the simple proxy would be to encrypt the link between the user and the proxy. Since it is not the routing information itself that betrays the user's actions but rather the content of the packets, this would prevent an eavesdropper on the user-proxy link from examining the packets and learning the final destination. This improvement protects the user's IP address from being linked to her actions by her ISP and by the web service independently. However there are still three problems with this

model.

The first problem is that the proxy server itself can link the user to her actions. If the proxy server is untrustworthy or compromised, the user is not anonymous. The second problem is that the encrypted traffic returning to the user will have a certain form in terms of how many packets are sent over what time interval. If an eavesdropper on the user-proxy link suspected that the user was accessing a particular service, the eavesdropper could access that service herself and build up a profile of the number of packets received in regular time intervals. This method is known as fingerprinting [46]. To illustrate this point, consider an example where the user browses to the Google homepage. Google may send the text of the website first in a burst of packets followed by some data from Google Syndicate and then the Google logo last. The amount of data arriving over time will form a distinctive shape—an initial rise, a pause, another rise, a pause, and then a large rise for the logo—that will not be obscured by the encryption. This attack is probabilistic and requires the adversary to first hypothesize who the user may be communicating with, both of which limit its effectiveness.

The third problem considers the possibility that the eavesdropper were to exist on both links. This could be a single eavesdropper listening in on the proxy server's line, a collusion between the ISP and the web service, or a third party with access to both the ISP logs and the server logs. The eavesdropper would see a correlation between the times when incoming encrypted packets arrive at the proxy from the user and when outgoing packets leave the proxy for the web service and vice versa. Even if many users are using the proxy server simultaneously, the proxy will generally process the packets in the order it receives them in and so simple timing analysis should suffice to untangle the traffic. If not, more sophisticated methods could include looking at the number of packets or fingerprinting the web services.

2.3.2 Mix Networks

In 1981, David Chaum introduced the mix node—an anonymising proxy server that uses a random permutation to remove order-based correspondence between an input and output message set, and cryptography to remove content-based correspondence [25]. A wide variety of modifications and extensions to Chaum's basic mix node have been proposed in the literature, as well as a variety of typologies for organizing these nodes into a network. Anonymity networks have been proposed to anonymise email [35] and

web traffic [37, 16, 64]. We review the original proposal and demonstrate how it solves various problems with the simple proxy servers in the previous section.

A mix node operates in discrete time and processes a finite set of messages, called a batch, from distinct senders at a given time step. The messages are encrypted by their senders either with the node's public key, or a session key negotiated using the node's public key. For simplicity, we will assume the former in our notation. The messages also contain random padding so that each input ciphertext is of a fixed length. The node accepts input messages until the batch is full, and then processes the batch. First the node decrypts each message and removes the random padding. It then performs a random permutation, or shuffle, on the group of messages which results in the batch being randomly resequenced. Given that the decryption function is a random mapping[1] between ciphertext and plaintext, an efficient shuffling method is sorting the ciphertexts according to a simple rule (*e.g.*, smallest to largest).

The decryption and unpadding operations ensure that each input message is negligibly correlated to each of the output messages. In other words, an eavesdropper may observe the set of encrypted messages entering the mix node. Given an output plaintext that she is interested in backtracking, the corresponding ciphertext should be indistinguishable from the set of all input ciphertexts. This assumes the encryption scheme used is semantically secure [54]. The eavesdropper however has a second avenue of attack. By repeatedly authoring messages addressed to herself or an accomplice, she may observe the position of the message in the input queue and attempt to correlate it with its output position and time. By randomly resequencing the order of the messages, these measures should negate any correlation between the input and output sequences. This prevents an eavesdropper listening on both the user-proxy and proxy-service link from being able to do timing analysis attacks (to further frustrate timing analysis, the mix node can introduce random delays on certain messages as will be discussed below).

While sending a message through a single mix node is theoretically sufficient for anonymity, often mix nodes are chained together to form a network. This can provide better statistical properties and ensure that a finite number of compromised or malicious mix nodes in the chain does not compromise the sender's overall anonymity. Each node along the route only knows the source node it received the message from and the next destination node it is sending it to. In a mix network, all messages follow a predefined

[1]The mapping between the elements of the two sets is unstructured while being deterministic and bijective.

route that is chosen *a priori* by the sender.

Message transfer protocol

To formalize the mix network, consider Alice who wants to send an anonymous message m to Bob at IP_B. She will select a path of three mix nodes, $N1$, $N2$, and $N3$, to send her message through. She obtains the IP addresses of each node, IP_{Ni}, and the public keys, e_{Ni}. The following shows the message Alice prepares and traces its route to Bob (recall that each node is processing a batch of messages and performing a group permutation on the order of the output, which is not shown).

$$A \rightarrow \mathrm{IP}_{N1} \quad : \quad E_{e_{N1}}(\mathrm{IP}_{N2} \| R_0 \| E_{e_{N2}}(\mathrm{IP}_{N3} \| R_1 \| E_{e_{N3}}(\mathrm{IP}_B \| R_2 \| m))) \tag{2.1}$$

$$\mathrm{IP}_{N1} \rightarrow \mathrm{IP}_{N2} \quad : \quad E_{e_{N2}}(\mathrm{IP}_{N3} \| R_1 \| E_{e_{N3}}(\mathrm{IP}_B \| R_2 \| m)) \tag{2.2}$$

$$\mathrm{IP}_{N2} \rightarrow \mathrm{IP}_{N3} \quad : \quad E_{e_{N3}}(\mathrm{IP}_B \| R_2 \| m) \tag{2.3}$$

$$\mathrm{IP}_{N3} \rightarrow \mathrm{IP}_B \quad : \quad m \tag{2.4}$$

Essentially, Alice is creating nested ciphertexts beginning with $N3$ and working to $N1$. Each node i removes a layer of encryption revealing the address of the next node, N_{i+1}, in the network, random padding R, and a ciphertext it cannot decrypt but forwards. Note that the random padding is only necessary if the encryption scheme is deterministic; alternatively a randomized encryption function (*i.e.,* ElGamal) could be used. In this model, $N1$ knows A and $N2$ but does not know $N3$ or B. $N2$ only knows $N1$ and $N2$ and does not know A or B. And $N3$ knows $N2$ and B but not A or $N1$. To trace the message, a node would need to know A and B. No individual node knows both. No two colluding nodes know both either except $N1$ and $N3$. However if they collude, $N1$ only knows it supplied one of the messages to $N2$. $N2$ will be processing an entire batch of messages, leaving $N3$ with no way of determining if it was given Alice's message or a different message. For this reason, theoretically speaking, only one node needs to be honest to achieve unconditional anonymity.

When Bob receives message m from $N3$, he or an eavesdropper will know IP_{N3} and could look up e_{N3}. He could thus construct ciphertext $E_{e_{N3}}(\mathrm{IP}_B \| m)$ which is similar to what $N3$ received as input. This demonstrates the importance of the random padding: without it, each hop could recreate the ciphertext the preceding node received. This

could be used to backtrack the message to Alice if the adversary were listening in on each link in the network (such an adversary is referred to as a global adversary).

Return address transfer protocol

In order to facilitate return messages, Alice must generate a public key, e_A, for each node in the network and retain the corresponding private keys. However these encryption keys are not published in a directory or linked to Alice's identity as public keys traditionally are. Alice can also choose between generating a n-tuple of new key pairs for each message she sends, allowing her to remain anonymous, or using the same set for multiple messages. The latter may allow her messages to be linked together but the set of messages will not be linked to her true identity—*i.e.*, she would remain pseudonymous.

Alice creates the following message,

$$\tilde{m} = E_{e_{N3}}(\mathrm{IP}_{N2} \| e_{A1} \| E_{e_{N2}}(\mathrm{IP}_{N1} \| e_{A2} \| E_{e_{N1}}(\mathrm{IP}_A \| e_{A3}))) \tag{2.5}$$

This message is an encrypted form of Alice's address and public key, of the same form as 2.1 except that the nested encryptions are inverted, with $N3$ representing the outer layer. Alice can use the message transfer protocol to send $\tilde{m}$ to Bob. Alternatively, $N3$ can retain $\tilde{m}$ and wait for a reply from Bob. Either way, $\tilde{m}$ is sent along with Bob's message m_b.

$$B \to \mathrm{IP}_{N3} \;:\; E_{e_{N3}}(\mathrm{IP}_{N2} \| e_{A1} \| E_{e_{N2}}(\mathrm{IP}_{N1} \| e_{A2} \| E_{e_{N1}}(\mathrm{IP}_A \| e_{A3}))),\ m_b \tag{2.6}$$

$$\mathrm{IP}_{N3} \to \mathrm{IP}_{N2} \;:\; E_{e_{N2}}(\mathrm{IP}_{N1} \| e_{A2} \| E_{e_{N1}}(\mathrm{IP}_A \| e_{A3})),\ E_{e_{A1}}(m_b) \tag{2.7}$$

$$\mathrm{IP}_{N2} \to \mathrm{IP}_{N1} \;:\; E_{e_{N1}}(\mathrm{IP}_A \| e_{A3}),\ E_{e_{A2}}(E_{e_{A1}}(m_b)) \tag{2.8}$$

$$\mathrm{IP}_{N1} \to \mathrm{IP}_A \;:\; E_{e_{A3}}(E_{e_{A2}}(E_{e_{A1}}(m_b))) \tag{2.9}$$

At this point Alice can decrypt this with her corresponding set of decryption keys, $\langle d_{A3}, d_{A2}, d_{A1} \rangle$, and recover m_b. In this protocol, Alice's nested IP address is being decrypted at each step until $N1$ recovers it. Similarly, each layer of encryption includes a key to encrypt m_b. This is to prevent someone who knows m_b, like Bob or an eavesdropper, from tracing the message through by watching for m_b.

Variations on the Basic Mix Network

Of the variety of mix networks, there are a few categorizations that will be relevant to this work. Messages are typically sent through more than one mix node. In a decryption mix network, the sender encrypts her message once for each node as shown in 2.1 above. This requires the user to know the path that the message will take *a priori*—either a chosen free-route or a path selected from a set of fixed cascades [38]. In re-encryption mix networks, the message is re-encrypted at each node requiring the sender to know, at a minimum, the first node to send her message to [59]. The sender can still know the entire route, but re-encryption allows for the possibility of a randomly generated route that the sender cannot trace. A special case of re-encryption is universal re-encryption where the final decryption is performed collaboratively by a collection of mix servers [44]. Messages are typically processed in batches. This is referred to as synchronous mix networks. They can also be processed asynchronously. A variety of flushing techniques can also be employed (including delays, pools, et cetera) to complicate attacks. See [1] for a list of recent research.

2.3.3 Onion Routing

Chaum's paper provides the theoretical basis for anonymous technologies [25] and can be applied to any generic message. Many papers have followed, and still do to the time of writing, with specific implementations of these concepts applied to various types of communication. Two predominant applications are email (SMTP) and web browsing (HTTP), the latter being the subject at hand. Onion routing was an architecture for HTTP transfer developed by David Goldschlag, Michael Reed, and Paul Syverson [42]. 'Onion' is an allusion to the nested encryptions that are peeled back a layer at a time as the message is 'routed' through the network.

The nodes in an onion routing network are of two types: routing nodes and proxy nodes. Routing nodes are interior servers that simply forward traffic between nodes in the network, while proxy nodes are the gateway nodes between the network and the senders or receivers. If Alice wants to send a stream of HTTP packets to Bob, she first contacts a proxy node through a secure connection (the node is likely a local application running on her machine). The entrance proxy node then chooses a path through the network, ultimately to another proxy node which will provide the HTTP data to Bob. While Chaum's design hints at the idea of processing each message independently, the

number of packets in even a simple HTTP exchange is too large to warrant independently processing them. Onion routing proposes a more efficient method where a circuit through the onion routing network is established, and then many streams of HTTP traffic from Alice to various senders can be routed through it. The circuit will change periodically.

Onion routing nodes respond to three commands: create a circuit, destroy a circuit, and transfer data. To create a circuit, Alice's entrance node creates a dual part packet. It generates an ID number for the circuit and puts this in a header with the command CREATE. It is assumed that the nodes have secure links between them using a stream cipher and the header information is transfered through this secure channel. The header does not contain information about where to route the circuit creation request. This information is put into an onion. Each layer of the onion contains an expiration time, the next hop, two pairs of an encryption function and decryption key, and random padding.

The expiration time is used to prevent replay attacks. Consider an eavesdropper Eve who wants to know who Alice is communicating with. She sees Alice send a message to $N1$ but cannot read it. Furthermore, $N1$ outputs a batch of messages and Eve does not know which one is Alice's. However Eve could log Alice's message and all of the output messages. Then at a future point in time, Eve could send to $N1$ an exact copy of Alice's message, and if it were processed by $N1$, then Eve could compare her logged batch of output messages to the new output batch and find the common message—which is Alice's. To prevent replays in onion routing, the created circuits are set to expire and current active circuits are kept in memory. If a circuit creation request is initiated twice, the second command can be ignored. The alternative to this is keeping large log files of received messages, which is inefficient in terms of space.

The next hop field in the onion contains the destination IP address (and port number) of the next node in the circuit. If the field is blank, the node is last node. The onion specifies a symmetric-key encryption function (*i.e.*, a block or stream cipher) to be used for en/decrypting future onions routed through the circuit. There are two function-key pairs. One is for onions traveling from sender to receiver, and the other is for the response traffic. The nodes store these keys securely in memory, indexed by the circuit ID number and with the expiration time. Because the nodes along the circuit are removing data from the circuit creation onion at each hop, the onion would become smaller. An eavesdropper could thus determine the hop position of a node by simply observing the size of the onion. To prevent this, each mix node pads the onion back to its original size with random bits before forwarding it.

When a circuit is terminated by the sender, expires, or a node disconnects from the network, a destroy circuit command is issued in an onion. Similarly to the create circuit command, the destroy circuit contains a header with the circuit ID and the command DESTROY, and an onion with the next hop in each layer. When a node receives a DESTROY onion, it sends a confirmation back to the node before it, deletes the keys associated with the circuit from memory, and forwards the command to the next node. If a node does not confirm deletion, it may be because the node is down.

Once a circuit is established, Alice can anonymously send data to Bob. Her proxy node splits the data into packets, places the data into an onion using the keys it distributed to the nodes during the circuit creation and puts the circuit ID into the header. Each node decrypts a layer of the onion until it reaches the exit node, which forwards the message to Bob. When Bob responds, the exit proxy node will receive the message and use the response keys to add a layer of encryption to the message. Each node adds a layer until it reaches the entrance node, which then removes all the layers and forwards the response to Alice.

Further work by the same authors offers packet-level specification, performance measures, and preliminary threat modeling [68]. They then extend the model by introducing an application to sit between the user and the proxy [63]. This application takes data from a specific protocol (web browsing, email, virtual private networking, remote login, *etc.*), sanitizes it, and creates a generic application-independent onion. These extensions also introduce the concept of entrance and exit funnels, which handle directing these generic messages to the correct application.

2.3.4 Tor

In 2004, Roger Dingeldine, Nick Mathewson, and Paul Syverson introduced Tor as a proposed 'second generation onion router' [37]. Tor offers many improvements over onion routing and places a high utility on efficiency. The Tor application is free, open source software and is available for a variety of operating systems including Windows, Apple, and Linux. Since its release it has become popular with an estimated 250,000 users worldwide [41], making it the most widely deployed anonymity tool.

Instead of using an onion to establish a circuit, Tor uses a telescoping method of negotiating session keys with each successive node in the circuit independently. Once a secure session is established with the first node, it is used to negotiate a key exchange with

the next node. This is repeated until the entire circuit is secured. The authentication protocol has been found secure under reasonable security assumptions [41]. For the sake of efficiency, Tor forgoes any mixing of messages at the node. This is based on the observation that a global eavesdropper could likely compromise the system despite mixing in real world scenarios, and so the marginal improvement in terms of security is not worth the efficiency cost. Tor does not claim to protect against a global eavesdropper, and traffic analysis attacks against Tor have been proposed where an attacker only sees part of the network [56].

Tor uses a SOCKS proxy [52] to interface with applications. Whereas onion routing would require a separate application interface for each program the user wants to anonymise, Tor's universal interface is supported by many TCP-based programs including web browsers. It also allows the user to route their traffic through a traffic sanitizer, like Privoxy [6], *en route* to Tor. Among other things, Privoxy can filter out unnecessary information in packet headers, remove banner ads, and manage cookies. The use of a single application interface also allows all internet traffic to be multiplexed through a single circuit, whereas onion routing would require a separate circuit for each internet application.

A directory of Tor servers is distributed among trusted nodes in the Tor network. The directory is periodically updated, and downloaded by the user. Onion routing used a decentralized model where new servers would announce their presence to the network, which creates a large amount of overhead traffic. Tor also uses integrity checks on data allowing the Tor network to protect against malicious servers that might modify traffic (traffic modification forms the basis of certain types of attacks [62]).

2.4 Digital Credentials

Digital credentials were proposed by Stefan Brands for identity management [19, 20]. They are similar to a digital certificate in that they enclose attributes in a signed document. However they differ from traditional certificates in the fact that these attributes are individually blinded. To illustrate the properties of a digital credential, consider three participants: Alice, Bob, and an Issuing Authority. Suppose Alice wants a digital version of her driver's license. The issuing authority creates a credential in cooperation with Alice, and encodes several attributes into the credential: Alice's name, address, date of birth, license number, and an expiration date. Both Alice and the issuing authority use

private keys in this protocol. The authority uses its private key to sign the credential. Alice uses her private key to ensure that she will be the only person able to use the credential. When the credential has been created and issued to Alice, she can 'blind' the credential [23], a process that makes it unrecognizable to the Issuing Authority without destroying the integrity of the attributes in the credential or the Issuing Authority's signature.

If Bob requires Alice to identify herself, Alice can give her credential to Bob. Bob can check, using the Issuing Authority's public key, that the credential was issued by the authority and is intact. If the authority sees Alice's credential, it will not be able to determine that it is the same credential it had given to Alice because of the blinding process. Also with the credential alone, Bob cannot determine any of the attributes in it. For these two reasons, a digital credential is anonymous until the attributes inside it are revealed (and if the attributes are not PII, then they remain anonymous).

Once Bob has checked the integrity of the credential, Alice can selectively reveal attributes inside the credential. This means she can, for example, reveal her name without revealing her address. To reveal an attribute, Alice claims that the credential contains a certain value, and then proves it does by showing a mathematical relationship that depends on her private key and on a random challenge chosen by Bob. This proof is unforgeable by anyone without Alice's secret key, and since it is in response to a random challenge, the credential and proof cannot be reused together. Digital credentials also allow Alice to prove properties about an attribute in her credential without revealing the attribute itself. Of particular importance to this work, Alice could prove an attribute is not equal to a certain value.

A credential takes the form of $(g_1^{x_1} g_2^{x_2} ... g_l^{x_l} h^{\alpha})$ in Z_p where x_i is an attribute, g_i and h are publicly known generators in Z_p^* chosen by the issuing authority, and α is Alice's private key. Knowing α and x_i for all i, Alice can prove through a challenge-response protocol the value or a property of x_i for a given i without revealing the value of the remaining x_j. The protocols involved in the issuing and showing of a credential will be examined in the forthcoming chapters as needed.

2.5 Usable Security and Privacy

There is much that can be said from a theoretical perspective on how anonymity can be achieved and how it could be broken. Many anonymity networks have been proposed and

examined for their theoretical merits. However the actual deployment of an anonymity network is arguably as difficult as its theoretic design, and for it to be usable by ordinary citizens, attention must be given to the tasks it requires its users to perform. Formal methods for testing the usability of software have been proposed. One methodology, the cognitive walkthrough, was proposed by Wharton et al. [70] and is based on the observation that users tend to learn software by exploring its interface and trying to use it instead of reading large amounts of supporting documentation. It suggests that a double-expert, one familiar with both the psychology of users and how a piece of software is intended to be used, determines a set of tasks that a typical user would want to perform and then walks through these tasks while evaluating the ease of performing them against some formal criteria.

A cognitive walkthrough is performed by Whitten and Tygar [71] in evaluating PGP (encrypted email) against a set of usability guidelines developed for security software. The authors discover a number of security risks and usability issues which are confirmed in a 12 participant user study. Goode and Krenkelberg [45] perform a cognitive walkthrough of KaZaA (a filesharing application) based on usability guidelines they adapted for P2P filesharing applications. More recently, Chiasson et al. [22] expanded on Whitten and Tygar with two additional guidelines. These guidelines are used in a user study of two password managers. Cranor [34] provides advice for software developers in the area of privacy based on lessons she learned in evaluating the usability of P3P and Privacy Bird. In Chapter 5, we will outline these guidelines while tailoring them to anonymity networks, and we will perform a cognitive walkthrough of a variety of deployment options for Tor.

Roger Dingledine and Nick Mathewson discuss several usability issues with Tor and other anonymity networks [36]. In particular, they note the difficulty of the task of configuring Tor for unsophisticated users. They propose that the most important solutions to this problem are improving documentation, providing the user with solution-oriented warning messages, and having bundled Tor with the additional components it relies on. While their paper has many useful insights, it does not apply any formal usability testing to Tor.

2.6 Summary

In this chapter, we have reviewed the cryptographic primitives to be used in subsequent chapters. We have provided an overview of anonymity networks, from Chaum's mix

networks through to onion routing and finally to Tor. Our contributions in the next three chapters will be made with respect to Tor, and Chapter 6 will detail how these contributions can be applied to other anonymity networks. Chapters 3 and 4 will make use of digital credentials, which we have introduced in this chapter, within Tor to provide some useful properties for combating adverse selection. Chapter 5 will evaluate the usability of Tor using the methodology of a cognitive walkthrough, reviewed in this chapter, with the proximate goal of establishing guidelines for anonymity software and the ultimate goal of increasing Tor's affinity among novice users who do not have the added incentive of hiding malicious behaviour to motivate them to use Tor.

Chapter 3

Exit Node Repudiation

3.1 Introduction and Motivation

In Chapter 1, we referenced a recent case regarding German law enforcement confiscating Tor servers that were used in the distribution of child pornography [58]. This particular case, and the potential for similar action against Tor servers in other countries, is the motivating problem of this chapter. We propose a new method that allows exit nodes in a Tor network to prove that all traffic they forward on behalf of anonymous users does not originate from their IP address. In cryptography, repudiation means disclaiming responsibility for an action [54]. We term our solution *exit node repudiation* (ENR) and differentiate it from past proposals for solving the motivating problem.

Exit node repudiation has primary and secondary effects. The primary effect is to alleviate the legal liability of operating an exit node. If this is achieved, some secondary effects may emerge. It may be easier to recruit node operators in an anonymity network, which will increase the global bandwidth of the network (in Chapter 6, we note why this may not increase bandwidth for individual users). Alessandro Acquisti, Roger Dingledine, and Paul Syverson also observe that users may increase their own security by operating a mix node [11] since they trust themselves, but they may also be hesitant if the legal environment is hostile. We demonstrate the benefits of ENR in the context of this observation in a later section, but first we introduce the problem and proposed solutions.

As discussed, there is an underlying ethical debate about whether cyber-criminals should enjoy the anonymity provided by anonymity networks or if the ability to revoke

their anonymity should be implemented. Similar debates have been held about cryptography, which allows individuals to transfer confidential messages, and steganography, which allows individuals to embed undetectable messages into innocent looking cover works like pictures or video. All of these technologies empower individuals with greater security and privacy, but they also create the potential for harm. On cryptography, the oldest of the three, the current consensus appears to be that the benefits outweigh the danger. In the United States, two events during the Clinton administration illustrate a shift toward this consensus: the first was the reduction of export restrictions on cryptography in 1996, and the second was the negative public reaction to the proposed Clipper chip, an encryption device for voice transmission that would have allowed law enforcement to keep a decryption key in escrow. It is our expectation that public opinion on anonymity will converge to a position similar to that of cryptography: that the benefits outweigh the danger. By that expectation alone, we argue it is prudent to consider solutions to the motivating problem that work within the framework of anonymity regardless of where one stands on the debate.

3.2 Related Work

Research on the technical side of this debate has predictably forked between providing traceability for messages in an anonymity network, and providing measures that allow the anonymity network to prove certain useful properties without revoking the anonymity of any senders.

3.2.1 Selective Traceability

The main work on the traceability side of the debate is by Luis von Ahn, Andrew Bortz, Nicholas J. Hopper, and Kevin ONeill [13, 14]. They propose a method that can selectively trace a single message in an anonymity network without revealing the origin of other messages. Their work first proposes a generic solution that is applicable to a wide variety of anonymity networks, including Tor, and then two efficient solutions for two specific types of anonymity networks (called DC-nets [24]) that are significantly different from mix networks, onion routing, and Tor. We concentrate only on the former solution since our concern is with *deployed* anonymity networks—all of which are based on some variant of a mix network. In particular, we are interested in solutions that are adaptable

to Tor, since it is the only deployed anonymity network with a substantial user base.

Selective traceability empowers a set of trustees with the ability to trace a message. Who the trustees are and the threshold of votes needed for traceability to occur is open to design. The author's solution uses two cryptographic constructions: threshold encryption [54] and group signatures [27]. Threshold encryption schemes are based on public key cryptography: the encryption key is public and known to all, however the decryption key is broken into shares and distributed to a number of trustees. When a plurality of trustees (equal to the pre-specified threshold) combine their shares, an encrypted message may be decrypted. A group signature allows any authorized member of a group to individually and anonymously sign a document on behalf of the group. Anyone can verify that a signed document bares the signature of a group member, but which member cannot be determined from the signature alone. However there is a group manager who can reveal which member created a given signature. These constructions are combined so that the role of the group manager is distributed to a set of trustees, where a threshold of trustees can reveal the signer of a message.

Users of an anonymity network must join a group and some identifiable information, such as an IP address, is bound to their group identity during the joining protocol. When the user sends a message, they anonymously sign it and attach the group signature to the message. The exit node checks the validity of the group signature before releasing the message. If the message is nominated for tracing, the trustees can vote on whether or not to trace the message based on some pre-specified criteria. If a threshold agree to trace the message, the signature is opened to reveal the identity of the signer.

The authors suggest their solution is problematic from a game-theoretic standpoint. They assert that the exit node has no incentive to check the signature—we argue that they are mistaken on this point. If an exit node might be held liable for the content of the message, it does have a positive incentive to verify the signature. In doing this, they are complying with a procedure that could ultimately allow them to repudiate the message. If the sender herself runs a server, she may ignore the signature but this is equivalent to her sending a message directly from her server without using an anonymity network at all. Similarly, if she colludes with a server to allow an unsigned message through the network, this is equivalent to her convincing the server to send a message on her behalf. It is also worth noting that traceability will likely be implemented only if required by some form of regulation, and this regulation could extend the requirement that exit nodes check the signatures.

3.2.2 Robustness and Reputability

In Chapter 2, we briefly mentioned re-encryption mix networks. Re-encryption networks allow for a number of interesting properties including proofs of robustness. A robustness proof allows a batch of re-encrypted messages to be proven to be a random bijection of the input set without revealing the permutation [47]. These proofs could be issued by the entire network, proving that a mix node did not add a message into the mix. This solution is theoretically significant however it is largely impractical. Re-encryption mix networks themselves are slow and have not been deployed for message transfer,[1] and the robustness proofs themselves would be inefficient on an anonymity network the size of Tor.

Phillipe Golle offers a weaker but computationally feasible form of robustness which he calls reputability [43]. His paper includes three designs, two of which are only applicable to universal re-encryption mix networks [44]. Since we are interested in deployed anonymity networks like Tor, we will concentrate on only one of his solutions—the one which is applicable to an onion routing scheme. Golle's solution employs a cryptographic construction known as a blind signature [23]. If Alice wants a blind signature on a message, she first obfuscates the message, typically by adding in a random value in a specific manner. She then gives this 'blinded' message to the signer, who cannot recover the original message without knowing the random blinding factor, and cannot feasibly compute the blinding factor because it is encoded in a trapdoor function such as a discrete logarithm. The signer signs the message as normal and returns the blinded message with the blinded signature. Alice is then able to remove the random blinding factor from both the message and the signature. Essentially, this process lets Alice get a signature on a message without the signer being able to read the message.

Golle's solution is specific to mix networks, but we will generalize it to any onion routing scheme, including Tor, and offer a few improvements. A group of nodes are designated to be the signing authority, and they engage in a threshold encryption scheme to publish a public key and generate a shared private key (the threshold allows the process to continue if a signing node goes offline). These nodes also collaboratively generate a random nonce which is updated periodically. A sender concatenates the nonce to her message, blinds it, and submits it to the signing nodes. The nodes sign the

[1] Re-encryption mix networks have, however, been used in cryptographic voting systems—*i.e.*, Prêt à Voter [28].

blinded message by applying a decryption with their shared private key. The sender then unblinds the message and sends it with the signature through the anonymity network. The exit node can verify that the message was submitted during the current time frame by verifying that the nonce is correct, and it can verify that the signature is valid by applying an encryption to it using the signing node's public key and checking that it matches the concatenation of the message and nonce it received. If both are valid, the message is released from the anonymity network. However in the paper Golle suggests that the exit nodes are not required to perform these checks. The purpose of this protocol is to prove that no messages were originated and mixed into the traffic by a node in the network; that all messages 'came in by the front door.' This result is similar to that offered by a proof of robustness.

3.3 Defining Exit Node Repudiation

Golle's proposed solution meets a criterion he terms near-reputability:

Definition 3.3.1 An anonymity network is **near-reputable** for demarcation function f, batch output B, and set of players P_B if there exists a subset of the batch output $f(B) \subseteq B$ such that each message in $f(B)$ can be proven to have originated from some player $p \in P_B$ without revealing which one.

First note that Tor does not perform mixing and therefore does not use batches; however, we can define a batch to be the set of all output messages during a certain time interval—such as the interval that the nonce was a given value—thereby making this definition applicable to Tor. Second, note that we expect legal action to be levied against the exit nodes of an anonymity network and not the anonymity network as whole. As a result, we prefer a definition of a near-reputable exit *node*. Consider the following definition as a potential candidate,

Definition 3.3.2 An exit node is **near-reputable** if it operates in a near-reputable anonymity network as per Definition 3.3.1.

This definition implies that an exit node can prove it did not originate a message in $f(B)$ if it is not in P_B. Every anonymity network is near-reputable for some f (for example, if f is the function that maps a set to the empty set). To avoid arbitrary

definitions, its useful to make some assumptions about f. In Golle's system, a message is in $f(B)$ if it contains a proper nonce and a proper signature. If all the nodes behave correctly and the exit node is in P_B, then this definition will suffice; but these assumptions are too strict. First, a major incentive to operating a node is the ability to mix in your own traffic (this way, you can ensure yourself that one node in the network operates correctly) and so requiring the set of exit nodes to be disjoint from P_B is not ideal. Second, we expect some nodes will not behave correctly, whether maliciously or as a result of unintentional data corruption. Thus we must consider the case that a message is not in $f(B)$ (*i.e.,* is in $B - f(B)$). Such a message may have originated from the exit node in question, or it may have originated from any other node in the network. The situation is ambiguous and offers plausible deniability to all nodes.

While it is formally fallacious to *require* inverses of definitions to be true, we have chosen to tighten definition 3.3.2 so that the consequent can be affirmed,

Definition 3.3.3 An exit node is **g-reputable** for batch output B, demarcation function g, and subset of all players $g(P) \subset P$ if every message can be proven to have originated from a player in $g(P)$ without revealing which one. **Exit Node Repudiation (ENR)** is the further condition that the only player in $P - g(P)$ is the exit node itself.

Note that function g operates on a set of players, whereas function f in the previous definitions operated on a set of messages. ENR divides the set of all players into two subsets: the exit node in question, and everyone else. Our proposed solution will query an algorithm to determine if a message originated from the set of 'everyone else.' If the algorithm returns true, the message is proven to not have originated from the exit node. If the algorithm returns false, the message is proven to have originated from the mix node. This definition is perfectly precise and resolves any ambiguity over the exit node's actions. If accused of originating a message, the message is either repudiable or non-repudiable. This definition presumes that the anonymity network will only output messages if they properly conform to a protocol and drop everything else. It also excludes the exit node from serving as an exit node for its own anonymous messages; however, it can still originate anonymous messages and either serve as an entrance or intermediary node or send them through other nodes in the network.

3.4 A Game-Theoretic Perspective

The subject of game theory considers the strategic interaction of agents and how an agent makes decisions in order to achieve a goal. An anonymity network could be considered a multi-participant game, with participants of three publicly-known types: senders, operators, and both [11]. Each sender also has a characteristic that is known to them but not to the other players: each sender is either honest or dishonest. This is an example of asymmetrical information. For our purposes, we define dishonest sender is one that transmits illicit data through the mix network. Illicit data could be data that violates the terms of service for the anonymity network, or data that may have legal consequences for the exit node: illegal data such as child pornography or death threats, data in violation of copyright, computer exploits launched against users or websites, *etc.* Honest users use the anonymity network as it is intended. This differs from the discussion in [11] which considers the strategic decisions of agents with respect to becoming operators or 'freeloading' off of existing operators, however we do refer interested readers to this paper for a thorough discussion of an agent's payoff function.

Consider the player Fox. Fox wants to send an anonymous message using the mix network by adopting the role of a sender, and Fox's dominant strategy is to 'trust no one.' It could be argued that players with this strategy would be the type of players attracted to using anonymity networks in the real world. Fox will not betray his dominant strategy but he will adjust his behavior in any way that facilitates his goal of sending his message. We adopt the role of system designers, and seek to offer a series of Pareto improvements[2] to the trust model of an anonymity network such that Fox will reach a best response[3] of successfully submitting a message.

In an anonymity network, a message is typically sent through several nodes. Theoretically, only one well-behaved node is needed for anonymity, but adding more nodes to the path increases the probability of using a trustworthy server. Assume the number of nodes is fixed at some number (three is typical in the real world). Since Fox's dominant strategy is to 'trust no one,' Fox will refuse to send his message to any operator he does not trust, and since he cannot observe the trustworthiness of other participants, he will refuse to send a message. It could be argued that his mistrust is well-placed; anonymity networks like Tor are voluntary, and honest operators are not compensated for the band-

[2]Improvements that increase the benefit of the current player without decreasing the benefit of other players. In this case, we limit our consideration to the benefit of other honest players.

[3]The strategy with the most beneficial outcome for the current player.

width and computational power they dedicate to the task of anonymising traffic. On the other hand, corrupt operators are compensated by the potential to trace senders. In a two-participant game, with only senders and operators, the incentives are structured for an adverse selection of corrupt operators.

This game involves a third type of participant—someone who is both a sender and an operator. If Fox becomes an operator in addition to being a sender, he can send his message through himself whom he trusts. Because only one well-behaved node is necessary for anonymity, this is acceptable with respect to his strategy. So the first improvement Fox makes is to become an operator as well as a sender. This choice has positive, non-zero externalities for all honest players, as it increases the number of well-behaved operators, and therefore represents a Pareto improvement. However, while this allows Fox to avoid an important trust issue, it does so only by introducing a new one.

Now as an operator, Fox must trust the users sending messages through his server to not disseminate illicit data. Fox, of course, has no grounds to trust his users as he cannot observe their characteristic and limit his services to honest players. Furthermore, his own anonymity relies on mixing his message in with the messages of other players, and so he cannot act selfishly by limiting his services to the only player he trusts: himself. A potential solution to this problem would be the redesign of the anonymity network to one that provides near-reputability. However once again, this simply displaces the issue without solving it. Reputable mix networks only provide reputability when all mix operators are well-behaved. In other words, if the mix network as a whole behaves, everyone is exonerated. If it does not, no one is. Since Fox does not trust the other operators to be well-behaved, we are back to the original problem and Fox will not use the anonymity network to send his message.

However if a method for providing exit node repudiation exists, Fox can exonerate himself from the corrupt behavior of other players regardless of the behavior of the other operators. Thus ENR will allow Fox to rationally justify his participation in an anonymity network while only trusting himself. This solution, summarized in Figure 3.1, has important theoretical consequences for the incentive structure of mix networks. Without ENR, players who do not trust other players will be averse to becoming operators, leaving a higher proportion of dishonest operators. This is a negative externality that is costly for all players, and may have compounding effects on well-behaved players who will not use the service unless if they feel a good proportion of the operators are well-behaved. Furthermore, without the ability to become an operator, players like Fox

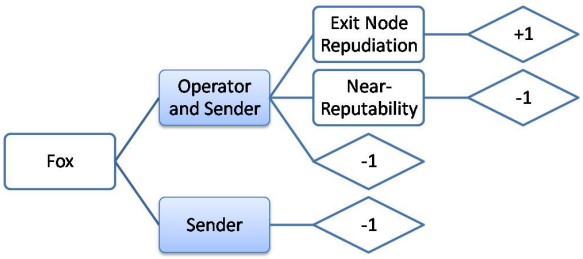

Table 3.1: Extensive form of strategic decisions (shaded) and mechanism design decisions (non-shaded) that allow Fox to reach a best response (+1). All other outcomes result in Fox not sending the message (-1).

will be left with no choice but to not use the anonymity network, lowering the proportion of honest senders. This illustrates how exit node repudiation can help combat adverse selection in anonymity networks.

3.5 Design Goals and Constraints

In our proposed solution for exit node repudiation, we will consider four players. The sender of the message is Alice, the entrance node is N_1, the exit node is N_3, and the recipient of the message is Bob (Note: the number of nodes in the route does not have to be three, this is simply the default in Tor). It is worth noting the following constraints on any solution:

1. N_1 sees Alice's IP address but cannot see the message.
2. N_3 sees the message but cannot know Alice's IP address.
3. N_1 and N_3 cannot mark the message, or anything associated with the message, in any way that will be recognizable to each other.
4. N_1 and N_3 cannot both see any parameter in the reputability proof that is unique to Alice.

Anonymity networks, by design, hide the path a message takes through the network from the nodes forwarding the message by restricting the nodes' view of the path. Each node only knows who they received the message from and who they should forward it to. This allows the network to provide anonymity even when only one node in the path is honest. Each of these constraints on our solution is a consequence of preserving the nodes' limited view of the path. If N_1 could see the message sent by Alice, it would recognize it in the output and could determine that Alice is communicating to Bob. Similarly, if N_3 knew an output message was sent from Alice's IP address, it could determine Alice is communicating with Bob. If N_1 and N_3 both see a common, unique element associated with the message (be it a mark placed by N_1 or a parameter in a protocol), then N_1 and N_3 could collude to link Alice's IP address to the message for Bob. This excludes N_2 from making any meaningful contribution to hiding the path of the message.

Our first design goal is to provide repudiation for individual mix nodes. While we could easily extend repudiation proofs to every node in the network, we will limit ourselves to providing it for only the exit nodes as they face the greatest liability.

The second goal is to base the repudiation proof on Alice's IP address instead of her message. Legal action has been instigated based on IP addresses, and so if court rulings are already worded in terms of IP addresses, an IP-based repudiation proof could conform to already existing legal precedents in various other cases that involved the liability of the owner of an IP address (eg. *Capitol Records v. Deborah Foster* in the United States). Note that using IP addresses is not a necessary constraint within our solution—for example, if there was widely deployed federated identity management system in place, those identifiers could be used instead. We concentrate on IP addresses because of their current prevalence.

The third design goal is to ensure that server logs are never needed, even in the case when the message did originate from a malicious node within the network. In Golle's near-reputable mix networks, if a message does not have valid signature, "the mixnet has no other option but to break the privacy of that message and trace it back either to an input, or to a malicious server who introduced the input into the batch fraudulently" [43]. Depending on how they are disclosed, server logs could break the anonymity of other innocent parties and should be avoided. Also, server logs are typically not secure and could be fabricated or selective in the connections they record. Furthermore, the servers logs could be distributed across many countries making them difficult or impossible to legally obtain. If tracing messages through server logs became common place, its likely

that mix nodes operating in countries hostile to foreign subpoenas would become the *de facto* servers used by criminals.

3.6 An Attempt at MAC-based ENR

As a proposal for ENR, consider cryptographically secure hash function H, Alice's IP address x, and N_3's IP address y. Alice can compute a hash of her IP address, $H(x)$, and concatenate it to her anonymous message m. Since hashes are non-invertable, there is no way for N_3 to recover x from $H(x)$. Furthermore, both N_3 and Bob know y and H. Either could compute $H(y)$ and compare it to $H(x)$. If they are different, it would appear that N_3 did not originate m. However there are a few problems with this proposal. The first is the integrity of $H(x)$. There is no way for Bob to know that N_3 did not replace $H(x)$ with a random value, and similarly there is no way for N_3 to know that Alice submitted a valid $H(x)$ instead of submitting a random value.

We can address this first problem with the observation that N_1 also knows Alice's IP address. We could thus require N_1 to generate $H(x)$ and sign it: $\sigma = \texttt{SIGN}(H(x))$. Alice now sends $m||H(x)||\sigma$ as her message. There are still a few problems with this scheme. First N_3 would have to know Alice's entrance node is N_1 to validate the signature. This could be prevented by having the entrance nodes compute group signatures. However, anyone interested in tracking m that had access to a list of all the potential senders' IP addresses could compute $\{H(\mathrm{IP}_1), H(\mathrm{IP}_2), \ldots, H(\mathrm{IP}_B)\}$ until they discover a hash equal to $H(x)$, at which point they will have identified the sender. This problem can be addressed by switching from hash functions to MAC functions. Since a MAC depends on a secret key, only N_1 could perform $\mathrm{MAC}_k(x)$. However this solution violates the fourth design constraint of the previous section: N_1 and N_3 cannot both see any parameter in the repudiation proof that is unique to Alice, and in this case they both see $\mathrm{MAC}_k(x)$. Furthermore if N_1 logged the value of x that corresponded to the $\mathrm{MAC}_k(x)$ it generated, then it could recognize $\mathrm{MAC}_k(x)$ in the output message and trace it to Alice without N_3's involvement. In response Alice could blind $\mathrm{MAC}_k(x)$ and the accompanying signature, but this leaves Bob with no way of comparing $\mathrm{MAC}_k(y)$ to Alice's $\texttt{BLIND}(\mathrm{MAC}_k(x))$. Furthermore it still requires N_3 to know N_1 in order to get $\mathrm{MAC}_k(y)$ generated by N_1's MAC key k. There appears to be no obvious way of using MACs to provide ENR.

3.7 ENR using Commutative Functions

For a more successful attempt at providing ENR, consider three commutative functions: f, g, and h (these have no relation to the demarcation functions f and g used in the definitions in section 3.3). Functions f and g are commutative if $f(g(x)) = g(f(x))$, and similarly with more than two functions. If each node in an anonymity network possesses a secret commutative function, then ENR could be achieved as follows. Alice establishes a circuit through N_1, N_2, and N_3. As in any anonymity network, each node only knows its immediate neighbours. N_3 sends its IP address y to N_2. N_2 applies its secret commutative function g to y and sends the result, $g(y)$, to N_1. Similarly N_1 computes $f(g(y))$ and sends this to Alice. Alice includes this with her message and sends her IP address forward through the chain so that it picks up a function at each hop. N_3 receives $g(f(x))$ and applies its own function to it to compute $h(g(f(x))$. It also applies h to $f(g(y))$: $h(f(g(y)))$. Since the functions are commutative, the order they are applied is not relevant. N_3 holds the equivalent of $(f \circ g \circ h)(x)$ and $(f \circ g \circ h)(y)$. If they are unequal, then $x \neq y$.

Addition and multiplication are commutative in $\mathbb{Z}_p$. If g_0 is a public primitive root in $\mathbb{Z}_p$ for a large prime p, then N_1, N_2, and N_3 could randomly select secrets $\{\alpha, \beta, \gamma \in \mathbb{Z}_p^*\}$ respectively. The composite function $(f \circ g \circ h)$ would apply a multiplication of $g_0^{\alpha} g_0^{\beta} g_0^{\gamma}$ or $g_0^{\alpha+\beta+\gamma}$ to the domain. This solution still allows N_1 and N_3 to see a common value: $f(g(y))$. The point of N_3 waiting to apply h to its own y is to prevent the final output from being recognizable to any of the other nodes. N_1 cannot link the final message to Alice, but the anonymity of the message is reduced to the trustworthiness of N_1 and N_3. To prevent this, N_2 can apply a second random function in the forward direction to the composite on y, and likewise apply both of its functions to the composite on x. As a final construction, N_1, N_2, and N_3 each randomly generate a (pair of) secret value(s) α, $\langle \beta_1, \beta_2 \rangle$, and γ respectively, all in $\mathbb{Z}_p^*$ for large prime p. Alice's IP address is x, N_3's IP address is y, and g_0 is a publicly known primitive root in $\mathbb{Z}_p^*$. The protocol is executed as follows,

$$\mathrm{N}_3 \rightarrow \mathrm{N}_2 \;:\; y \tag{3.1}$$

$$\mathrm{N}_2 \rightarrow \mathrm{N}_1 \;:\; y g_0^{\beta_1} \tag{3.2}$$

$$\mathrm{N}_1 \rightarrow \text{Alice} \;:\; y g_0^{\beta_1} g_0^{\alpha} \tag{3.3}$$

$$\text{Alice} \rightarrow \mathrm{N}_1 \;:\; \left\langle y g_0^{\beta_1} g_0^{\alpha}, x \right\rangle \tag{3.4}$$

$$\mathrm{N}_1 \rightarrow \mathrm{N}_2 \;:\; \left\langle y g_0^{\beta_1} g_0^{\alpha}, x g_0^{\alpha} \right\rangle \tag{3.5}$$

$$\mathrm{N}_2 \rightarrow \mathrm{N}_3 \;:\; \left\langle y g_0^{\beta_1} g_0^{\alpha} g_0^{\beta_2}, x g_0^{\alpha} g_0^{\beta_1} g_0^{\beta_2} \right\rangle \tag{3.6}$$

$$\mathrm{N}_3 \rightarrow \text{Bob} \;:\; \left\langle y g_0^{\beta_1} g_0^{\alpha} g_0^{\beta_2} g_0^{\gamma}, x g_0^{\alpha} g_0^{\beta_1} g_0^{\beta_2} g_0^{\gamma} \right\rangle \tag{3.7}$$

$$\text{Bob} \;:\; y g_0^{\alpha+\beta_1+\beta_2+\gamma} \stackrel{?}{=} x g_0^{\alpha+\beta_1+\beta_2+\gamma} \tag{3.8}$$

This ENR protocol uses the same distributed trust model as the anonymity network—as long as one node is trustworthy, Alice cannot be linked to her message. Consider one hop in the network: N_2 receives a value equal to $x g_0^{\alpha}$ from N_1. This equation has two unknowns: x and α, and thus if α is kept secret, x is unrecoverable. Furthermore, if N_2 has a list of all the possible x values (*i.e.*, a list of the potential senders and their IP addresses), there exists an α for every possible x such that $x g_0^{\alpha}$ is constant. Thus, every honest node unconditionally obscures the message it received from the previous node.

The motivating problem also requires a second element of trust. The beneficiary of the proof, law enforcement, must trust the integrity of the final two values in the inequality. Unlike Alice who simply needs one honest node, law enforcement must trust all of the nodes to correctly follow the protocol. If even a single malicious node does not apply the same function to both the composite on x and composite on y, then the final values could produce an inequality even when they are equal. More simply, N_3 could simply report a random number for either value. Since the functions are secret, there is no inherent integrity.

Integrity can be added to the ENR protocol with the addition of a cut-and-choose protocol [26]. We will demonstrate this for N_2, since its participation is the most complex, however the same principal can be extended to any of the other nodes. N_2 is associated with three input-output pairs: (1) it receives y and generates $y g_0^{\beta_1}$; (2) it receives $y g_0^{\beta_1+\alpha}$ and generates $y g_0^{\beta_1+\alpha+\beta_2}$; (3) it receives $x g_0^{\alpha}$ and generates $x g_0^{\alpha+\beta_1+\beta_2}$. It performs these three computations with the pair of secrets β_1 and β_2. Suppose that N_2 generates four

secrets instead: β_1, β_2, β_3, and β_4. First it performs the three calculations with $\langle \beta_1, \beta_2 \rangle$, and then it performs them again with $\langle \beta_3, \beta_4 \rangle$. The six input-output pairs (three with each key pair) are given to an independent auditor. The auditor flips a coin, and asks N_2 to reveal the values of either $\langle \beta_1, \beta_2 \rangle$ or $\langle \beta_3, \beta_4 \rangle$. The auditor then checks that three input-output pairs that were generated with this key pair were correctly formed, and the other input-output pairs are used in the ENR protocol. If the node cheats once, it faces a 50% probabily of being caught. This probability can be adjusted by having the node produce n key pairs, and auditing all but 3 of the $3n$ input-output pairs. The probability of catching the node cheating during a given audit is $1-1/n$. The probability of catching sustained cheating after d audits is $1 - 1/n^d$. For $n = 2$ and $d = 10$, the resulting probability is over 99.9%.

This protocol would be executed during the `CREATE` command in onion networks, and the proof would be issued once per circuit. In anonymity networks that do not use mixing, like Tor, the auditor clearly cannot audit all of the nodes for a single execution of the ENR protocol or it would be able to determine the full circuit and then trace Alice's messages. It would randomly choose a node and step in the protocol to audit, wait for the node to produce an output for that step, halt the protocol, and then request the node to produce a second functionally equivalent output for that step.[4] Any node caught cheating would be expelled from the network. In the next section, we introduce a second ENR protocol, and then debate the respective merits of the two protocols in the concluding remarks of this chapter.

3.8 ENR using Digital Credentials

We now consider a second construction of an ENR protocol. This protocol encodes Alice's IP address into a digital credential, and Alice offers proof that the IP address in the credential is not equal to the IP address of N_3, without revealing the actual value of her IP address. We illustrate the protocol assuming that the credential is issued by N_1, given that N_1 knows Alice's IP address, and we consider alternative architectures after demonstrating the protocol. The ENR protocol will be performed after the `CREATE` circuit command in Tor, and prior to Alice submitting a message. We assume that N_1

[4]Note that these functions are easily invertible as well as being commutative. Thus a node can change a function that it applied to a message several steps prior even if other nodes have applied their functions during the ensuing steps.

is a member of a group of entrance nodes, and members of this group are capable of performing a group signature. Alternatively, the private keys used to sign the credential, $s1$ and $s2$, could be shared by members in the group. How and by whom the secrets are generated is flexible and a matter of policy. The key generation protocol is shown in Algorithm 1. Note that all algorithms are derived from the work of Stefan Brands [19, 20]; associated security proofs can be found therein.

Algorithm 1: Key Generation

Input: Public parameter p.
Output: Private key $\langle s_1, s_2 \rangle$ and public key $\langle g_0, g, h \rangle$.
1 **N_1 should:**
2 Choose random secrets $s_1, s_2 \leftarrow_r \mathbb{Z}_p$.
3 Choose random generator $g_0 \leftarrow_r \mathbb{G}_p$.
4 Compute $g = g_0^{s_1}$ and $h = g_0^{s_2}$.
5 **end**

Public parameter p is a suitably large prime number (*e.g.*, 1024 bits), and g_0 is a primitive root in $\mathbb{Z}_p^*$. The public key of N_1 is $\langle g_0, g, h, p \rangle$, while $s1$ and $s2$ are retained as a private key. Note that $s1$ and $s2$ cannot be recovered from the knowledge of the parameters in the public key without computing a discrete logarithm, a problem we assume to be computationally infeasible.

3.8.1 The Issuing Protocol

The issuing protocol is shown in Algorithm 2. The key generation algorithm produces public parameters g and h, which are arranged by Alice into a credential of form $(g^x h)^\alpha$ where x is Alice's IP address. This credential can be thought of as having secret key alpha applied to an encrypted attribute x. For every value of x, there is a unique value of α that will produce the same value for the credential. Thus if α is unknown, the value of x is unconditionally secure. It is impossible to recover x from a credential without a method for distinguishing the correct value of x from the complete set of possible values x could take. If such a method existed, then the adversary would already know x.

In Algorithm 2, Alice creates the credential, I, and N_1 provides a signature certifying that x is correct. Note that N_1 never sees the value of the I itself and so it cannot recognize I when Alice uses it. The 'signature' on the credential, $\langle c, r \rangle$, is more properly a private key certificate [19] however we refer to it as a signature for convenience. Once

Algorithm 2: Issuing Protocol

Input: Public Key $\langle g_0, g, h, p \rangle$, A's IP address x, and (known only to N_1) Private Key $\langle s_1, s_2 \rangle$.

Output: Credential I and signature sig(I)= $\langle c, r \rangle$.

A **should:**
  Choose random secret $\alpha \leftarrow_r \mathbb{Z}_p^*$.
  Compute $I = g^x h^\alpha$.
end
$\mathbf{N}_1$ **should:**
  Choose random secret $w \leftarrow_r \mathbb{Z}_p$.
  Compute $z = g_0^w$ and send to A.
end
A **should:**
  Choose random secrets $\beta_1, \beta_2 \leftarrow_r \mathbb{Z}_p$.
  Compute $c = \mathcal{H}(I, (g^x h)^{\beta 1} \cdot g_0^{\beta_2} \cdot z)$.
  Blind c by computing $\tilde{c} = c + \beta_1$ and send to N_1.
end
$\mathbf{N}_1$ **should:**
  Compute $\tilde{r} = \tilde{c}(s_2 + x s_1) + w$ and send to A.
end
A **should:**
  Verify $z = g_0^{\tilde{r}} (g^x h)^{-\tilde{c}}$.
  Unblind r by computing $r = \tilde{r} + \beta_2 + c\alpha$.
end

again, N_1 does not see $\langle c, r \rangle$, only a blinded version of the values: $\langle \tilde{c}, \tilde{r} \rangle$. The protocol employs a hash function $\mathcal{H}$ which is assumed to be publicly known and cryptographically secure. Alice employs the hash to send a function of her credential, c, to N_1 who calculates a suitable response using the value of x. Should Alice's credential not contain the same value of x that N_1 uses in its response, the signature will not hold (the validation protocol for the signature is discussed below).

3.8.2 The Showing Protocol

Algorithm 3 demonstrates how Alice can generate a signed proof that the attribute in her credential x is not the same as another attribute y. In this case, x is her IP address and y is the IP address of the exit node. The IP address of the exit node must be known by Alice. While it is more efficient if she knows it *a priori*, it is possible for N_3 to send its IP address back through the mix network to Alice. In onion routing networks like Tor, Alice can chose her own exit node and thus knows its IP address. Alice would create a circuit to N_3 using the standard circuit creation functionality on an onion routing network, and then she would include the output of Algorithm 3 inside an onion routed to N_3 as the final step in the circuit's creation.

Algorithm 3: Signed Proof ($x \neq y$)

Input: $\langle g_0, g, h, p \rangle$, I, x, α, N_3's IP address y, and nonce n.
Output: $\langle a, r_2, r_3 \rangle$.

1 A **should:**
2 Choose random secrets $w_1, w_2 \leftarrow_r \mathbb{Z}_p$.
3 Compute $a = I^{-w_1} g^{y w_1} h^{w_2}$.
4 Compute $c_1 = \mathcal{H}(a, I, y, n)$.
5 Compute $\epsilon = y - x$.
6 Compute $\delta = \epsilon^{-1}$.
7 Compute $r_2 = c_1 \delta + w_1$.
8 Compute $r_3 = c_1 \alpha \delta + w_2$.
9 **end**

The showing protocol is based on a challenge-response, where the challenge requires nonce n. The nonce is used to ensure that the credential is not used by anyone other than Alice (*i.e.*, only by those who know the secret key α). If the protocol were not challenge-response, the credential and proof could be replayed together by someone who observed Alice using a credential. We suggest that the nonce be a hash of the message,

Bob's IP address which N_3 knows, and a large random number collaboratively generated by the nodes in the mix network—the latter being published with a timestamp and periodically updated. This does not completely prevent replay attacks but it severely limits them to the same message and same receiver in the same window of time. This small cost is outweighed by the benefit of a standardized public nonce: Alice can compute the value of the nonce *a priori* and can create her response without having to exchange any information with N_3.

The proof itself is based on the observation that if x (Alice's IP address) and y (N_3's IP address) are different, their difference is non-zero and thus invertible within an appropriate finite group such that $(x-y)(x-y)^{-1} \equiv 1$. If x and y are the same, the difference is zero which is non-invertible, leaving δ uncalculated (or zero if the inverse of zero is so defined). However in the case that $\delta = 0$, then r_2 and r_3 would equal w_1 and w_2 respectively, and the verification procedure in Algorithm 4 would fail.

Algorithm 4: Verification Algorithm

Input: $\langle g_0, g, h, p\rangle$, $\langle I, c, r, a, r_2, r_3\rangle$, y, n.
Output: TRUE or FALSE.
N_3 **should:**
  Verify $c = \mathcal{H}(I, g_0^{r_0}(Ih)^{-c})$.
  Compute $c_1 = \mathcal{H}(a, I, y, n)$.
  Verify $I^{r_2}a = g^{r_2 y - c_1}h^{r_3}$.
end

The complete package that Alice delivers to N_3 is $\langle I, c, r, a, r_2, r_3\rangle$. There are three distinct parts to this package: I is the credential, c and r are used to verify N_1's signature on the credential, and a, r_2, and r_3 are Alice's signed proof that x is not equal to y. This verification should be performed by N_3 before completing the circuit creation protocol. If either verification fails, the circuit should be destroyed. The package can also be forwarded to Bob, who also has all the information needed to verify the correctness of the credential as long as he trusts that the credential issued by N_1 contains a valid value of x. This is important because it allows law enforcement to satisfy themselves of ENR without requiring any *ex post* interaction with N_3.

Instead of having the credentials issued by the entrance nodes of the anonymity network, they could be issued by an independent third party. This third party does not have to be trusted by Alice—it never sees the credential or associated signature during the issuing protocol, and Alice can verify for herself that certification matches

the information inside her credential. Since we are proposing ENR as a legal solution, a more suitable architecture would have the credentials issued to Alice by a third party that is trusted by the courts—it could be governmental or law enforcement itself. The credentials can be independent of what anonymity network Alice wants to use or what message she will send; in fact, they could be used for any online purpose where Alice wants to prove some property about her IP address. Furthermore, Alice can be issued a large quantity of credentials in bulk, each unique but with the same attribute, at some time prior to using an anonymity service as long as the issuing authority's public parameters are still known when she uses the credential. This changes the efficiency of the issuing protocol from a marginal cost to a fixed overhead cost.

3.9 Concluding Remarks

One criticism of both proposed ENR protocols is the validity of x in the credential. For example, its possible for a credential to be issued to a user at one IP address and then used by the same user to send a message from a different IP address. It would also be possible to use a proxy server to interact with credential issuer, so that the proxy server's IP address is encoded into the credential instead of Alice's. In response to this criticism, we note several things. First, Alice has no incentive to try to obscure her IP address from the credential issuer. The only property of her IP address that will be revealed is that its not equal to N_3's and any further proofs about x or its properties requires the collusion of the mix nodes in the case of commutative-based ENR, or Alice's private key in the cased of credential-based ENR. Second, lending and borrowing credentials is the equivalent of using someone else's computer; something that is possible independent of whether an anonymity network is even used. Third, lists of known proxy servers have been compiled and could be checked. Fourth, the legal alternative to ENR is traceability and this problem applies equally to it. If a messaged is traced through an anonymity network to a supposed sender's IP address, there is no guarantee that the IP address is actually the sender's and not that of a proxy server or compromised machine.

We believe that credential-based ENR is superior to the proposed ENR protocol based on commutative functions. Both afford proofs of integrity to law enforcement, but the former allows the oversight to be arms-length and independent of the anonymity network. Commutative-based ENR requires a cut-and-choose protocol that is logistically challenging: the auditor would need to be privy to the creation of every circuit. Also the integrity

is only probabilistic and extra computations are required by every node in the circuit. The architecture of the credential-based ENR is elegant, and would be easier to explain in a court of law by abstracting the mathematics and talking about credentials in their high level form. Credential-based reputability also offers an additional property: sender revocable anonymity. If Alice wanted to claim authorship of an anonymous message at a future time, she simply would need to keep the private key of the credential she used to establish the circuit used for transmitting the message. For example, a whistle-blower may wish to remain anonymous while she is in fear of retribution for her actions but later when the problem has been resolved, she may want to take credit for having reported it. She can offer a signed proof that x in the credential is equal to her IP address (the protocol for equality proofs is given in the next chapter).

In conclusion, we note that in Canada, a computer may be seized only if it "will afford evidence with respect to the commission of an offence, or will reveal the whereabouts of a person who is believed to have committed an offence.[5]" If the exit node of an anonymity network is reasonably shown to not have been the originator of an unlawful message, it will be much harder to convince a judge that seizing the computer is reasonable, at least with regard to the commission of the offence. Exit node repudiation provides a method of retaining the anonymity of the sender while presenting a response to the pertinent legal question of liability for the exit node. We hope this innovation is helpful in preserving the legality of anonymity networks and decreasing the legal aversion to volunteer operators of servers in the network.

[5]Criminal Code of Canada: 487 (1) b)

Chapter 4

Revocable Access for Malicious Users

4.1 Introduction and Motivation

In the previous chapter, we introduced the concept of exit node repudiation (ENR) and two protocols that allow an exit node to prove a message did not originate from it. Furthermore the exit node is afforded the ability to check that the protocol has been followed by the sender (who will remain anonymous and thus does not have a negative incentive for following it) and can drop messages that do not conform. We propose ENR as a counter-measure to the legal liability an exit node may face for the messages transmitted through it, and argue that through open-design and verifiable protocols, an exit node can demonstrate *a priori* that it will not 'afford evidence' of the sender's identity; an important legal criterion for the authorization of search and seizure in Canada and other Western democracies. In isolation, this solution preserves the anonymity of all senders—legitimate users and criminals—and is premised on the ethical view that the right of legitimate users to unconditional anonymity outweighs the efficacy of criminal prosecution. However until a legal precedent rules in favour of either side, the legal permanency of anonymity networks is indeterminate.

Therefore it is incumbent upon anonymity networks to add weight to their side of the argument. This can be accomplished by increasing the incentives of honest users to use the service, while decreasing the incentives of malicious users. This approach requires a separating equilibrium—a signal that separates honest users from malicious users.

There are at least two such signals. The first, and perhaps unfortunate, signal is that malicious users have a greater incentive to use the service and therefore are more likely to surmount any deployability or usability problems that would deter other potential users. This signal offers only a loose correlation. Many honest users have high incentives as well, and possibly even greater ones. The avoided loss afforded by anonymity to criminals may be incarceration, but for political advocates in oppressive nation states or military operatives in unsafe environments, anonymity could be life preserving. Therefore we expect this signal to have a high false positive rate (with the null assumption being honesty).

A far better signal, in these situations, is past behaviour. And since, by definition, the categories of honest and malicious are dictated by behaviour and not some intrinsic property of the user, this is in fact a tautological signal for establishing a separating equilibrium. In the case of anonymity services, past behaviour should not be linkable to present or future behaviour by the very definition of anonymity. If the actions were linkable, users would be achieving some privacy—pseudonymity—but not anonymity. There is an intrinsic, fundamental tension between the concept of anonymity and the concept of reputation.

This chapter will propose three methods for synthesizing the reputability of a user with anonymity. The goal is to develop a protocol that allows malicious users to be banned from using Tor in the future. This admittedly does not prevent one-time abuse, but it does create a negative incentive for sustained malicious actions. The final proposed architecture extends existing work done on the subject [48]. Our solution differs in a few fundamental ways: we use credential-based challenge-response identifiers to prevent offline attacks, we allow for IP addresses to be preemptively banned, and our interest is in banning users from using anonymity networks themselves, not from web services accessed through the network.

4.2 Moral Hazard and Anonymity

The overarching theme of this thesis is how to combat adverse selection in anonymity networks—that is, creating positive incentives to attract honest users and server operators, and negative incentives to discourage malicious or unlawful users. We have argued that that anonymity is more attractive to criminals than to the ordinary citizens who have only a moderate preference for privacy and anonymity. We can model Tor as a

market where the server operators are providing a service, presumably for some personal benefit or altruistic purpose. The consumers of this service know whether or not they behave unlawfully but the operators cannot distinguish good users from bad users in advance. So in an anonymity network, information is not evenly distributed between consumers and providers—a result economists refer to as asymmetrical information. A group of economists who pioneered research on markets with asymmetric information, including George Akerlof [15], shared a Nobel prize in Economics in 2001 for their work. In addition to identifying adverse selection, Akerlof considers the problem of moral hazard: a consequence of asymmetric information and a sort of dynamic version of it.

In Chapter 1, we introduced the concept of adverse selection with the short analogy of unregulated medical insurance. Extending this analogy, consider a low-risk individual that, against the odds of adverse selection, does purchase medical insurance. However as a consequence of being insured, the person begins to engage in more risky activities. This outcome is called moral hazard by economists [15]. An actual example of moral hazard is seat belt legislation, which can cause a marginal increase in the risks taken by drivers. When aggregated across a large population, the product of this increased risk is significant [61].

Regarding anonymity, we should question if well-behaved users behave marginally worse as a consequence of anonymity. As evidence in favor of a moral hazard, consider anonymous edits to Wikipedia. Wikipedia is an open-access encyclopedia that anyone can edit, including users who have not registered with the site (this privilege may be temporarily revoked for specific articles if they are controversial or subject to repeated malicious edits). Such edits are called 'anonymous edits' although they are not actually anonymous: Wikipedia logs the IP address of the user who makes the edit. In August 2007, Virgil Griffith created a novel tool called WikiScanner that allows users to search for anonymous edits by IP address [1]. Since corporate entities and organizations tend to reserve a block of publicly known IP addresses, WikiScanner allows users to search for edits originating from a given entity's network. The tool gained notoriety for exposing questionable edits, such as an edit from Diebold's corporate network removing a criticism section from the Diebold article[2], or an edit originating from the office of Turkey's Undersecretariat of the Treasury removing a reference to the Armenian genocide[3]. There

[1] Offical Website as of publication: `http://wikiscanner.virgil.gr/`

[2] `http://en.wikipedia.org/w/index.php?diff=prev&oldid=28623375`

[3] `http://en.wikipedia.org/w/index.php?diff=prev&oldid=77155119`

are many others that were reported.

If we accept the reasonable hypothesis that the originators of these kinds of edits were unaware that the edits would be linked to their organizations, two conclusions can be drawn. The first is the apparent discrepancy between users' mental model of the internet and how it actually works. The implications of this with respect to identity and IP addresses is pursued in the next chapter. The second conclusion is that the illusion of anonymity does appear to cause a moral hazard among users. The effects are likely marginal—anonymity does not necessarily make criminals out of ordinary citizens—but there is no negative incentive preventing some shift in behaviour. This chapter will propose a method for revocable access which serves as an negative incentive against sustained mischievous behavior, whether adversely selected or the product of moral hazard.

4.3 Anonymity through Distributed Trust

In an anonymity network—be it a simple proxy server, a mix network, an onion routing network, or Tor—Alice essentially transfers another entity's identity onto her actions. If her identity is IP_A, she assumes the identity of IP_{N1} to submit an action. If Alice is the only user assuming this identity, she is not anonymous because her actions as IP_{N1} can be linked together. So a sufficient condition for anonymity is that she assumes an identity that is used by others. Let f be a secret function defined on $\mathcal{A}$, the set of all users, which maps to the set of all identities $\mathcal{B}$. User $a \in \mathcal{A}$ is anonymous if $|A_b| > 1$, where A_b is the set of all $a_i \in \mathcal{A}$ such that $f(a_i) = b$, where b is a single element in $\mathcal{B}$.

The set A_b is sometimes referred to as the anonymity set because actions performed by the elements of the set through a many-to-one f are indistinguishable, as long as f is unknown. However f in an anonymity network is known to the node applying it (and may also be determined through some forms of attack). Thus anonymity networks provide a composition of functions $\{f_1, f_2, f_3, \ldots\}$. Each function takes the same form: $\mathcal{A} \cup \mathcal{B} \to \mathcal{B}$. The final identity, $(\ldots \circ f_3 \circ f_2 \circ f_1)(a)$, requires knowledge of the complete set $\{f_1, f_2, f_3, \ldots\}$ with respect to a. In an anonymity network, each f_i is performed by a different node, resulting in distributed trust.

In short, the *secrecy* of mapping f provides pseudonymity for an otherwise identifiable user, and the *many-to-one* nature of mapping f provides anonymity. The former condition is necessary but not sufficient for anonymity, and the combination of the two is sufficient but not necessary. A second way to achieve anonymity from the pseudonymity

provided by a secret f is to use a different $b = f(a)$ for every action. The elements of $\mathcal{B}$ have been called *nyms* in the literature, and anonymity can be achieved by using a unique nym for each action. Furthermore, the mapping between an identity and a nym can be composite: an identity can be mapped to a nym, and this nym can be mapped to another nym, *etc.* In order to determine the corresponding identity of the final nym, every mapping must be known. In the same way that mix networks use distributed trust to provide anonymity, anonymous nyms can be created and deployed.

4.4 `BAN` Cells

In onion routing networks like Tor, the nodes recognize and respond to several commands. For instance, there are protocols for creating circuits, destroying circuits, and transferring data. Our first attempt at implementing a protocol to revoke a user's access to an anonymity service focuses on extending the already established protocols, instead of introducing architectural changes. Assume that a web server who has received illicit data is somehow granted authorization to have the originator of this data banned. In the same way that backward `RELAY` cells/onions are created, a similarly structured `BAN` cell could be constructed. A dispute resolution entity would exist to authorize the banning of users. It could signal a decision by digitally signing a `BAN` cell and maintain a list of banned IP addresses.

A `BAN` cell would have the `BAN` command in its header, along with the circuit ID. It would be routed back to the entrance node as any return data would, and like in a `DESTROY` circuit cell, each node would be required to confirm the receipt of the onion. The entrance node would initiate a `DESTROY` circuit command and provide the dispute resolution entity with the IP address to be added to the ban list. Future `CREATE` circuit commands would be initiated only after consulting with the dispute resolution entity to ensure the requesting IP address is not on the ban list.

This suggestion is problematic in a few illustrative ways. The first is that it requires action to be taken while the circuit is still established and not afterward. It also relies on the nodes to be trustworthy. If the malicious user were to know the nature of this process, she could route her traffic through a node she controls. When a she receives a `BAN` cell, she could ignore it or if she was the entrance node, she could provide a false IP address to be banned. These shortcomings provide us with information about how to proceed. First, the mapping between an action and identity must persist after

the user has disconnected from the network. Second, nodes in the network cannot be relied on to facilitate revoked access. Even if a malicious node's non-cooperation were identified, there is no way to compel it to reveal the IP address. This appears to require the participation of a trusted third party.

4.5 NYMBLE

The NYMBLE system, proposed by Peter Johnson, Apu Kapadia, Patrick Tsang, and Sean Smith, proposes the use of two non-colluding servers [48] to ban users from Tor at the request of a web service. When Alice connects to Tor, she must first register with the Pseudonym Manager (PM), and receive a pseudonym. The pseudonym is a unique identifier based on IP_A and generated deterministically with a one-way function so that the same IP will always yield the same pseudonym, but given a pseudonym, it is computationally infeasible to invert it and recover the IP address. Alice then connects anonymously (through the anonymity network) to the second server, the Nymble Manager (NM). Alice presents her pseudonym and an accompanying message authentication code that NM can use to validate that the pseudonym originated from PM. In return, NM issues Alice a series of nymble tickets. Each ticket is unique, and they are initially unlinkable by anyone other than NM. Alice uses these tickets when she is sending messages through the anonymity network. If a web service complains about her behavior, it provides NM with the malicious user's ticket and NM provides the web service with a method for identifying the rest of the user's unexpired tickets. NYMBLE also offers methods for time-splicing the validity period of tickets and pseudonyms, and offers functions to allow servers to check the validity of the tickets and for users to check if they are on the banned list. The NYMBLE system's main advantage is its speed. By concentrating on hash-based functions and symmetric key ciphers, NYMBLE minimizes public key operations.

The goal of NYMBLE is to allow web services to ban malicious users. For instance, Wikipedia may wish to ban users who deface it. However Wikipedia and other web services have no way of verifying that the users are actually being banned. PM could issue a different pseudonym to a banned user, and NM could generate tickets that are not actually based on the pseudonym. This emphasizes an important point about the use of trusted servers in this context: the trust is dual factor. Users must trust the servers to not break their anonymity and unlinkability, and web services must trust the servers that

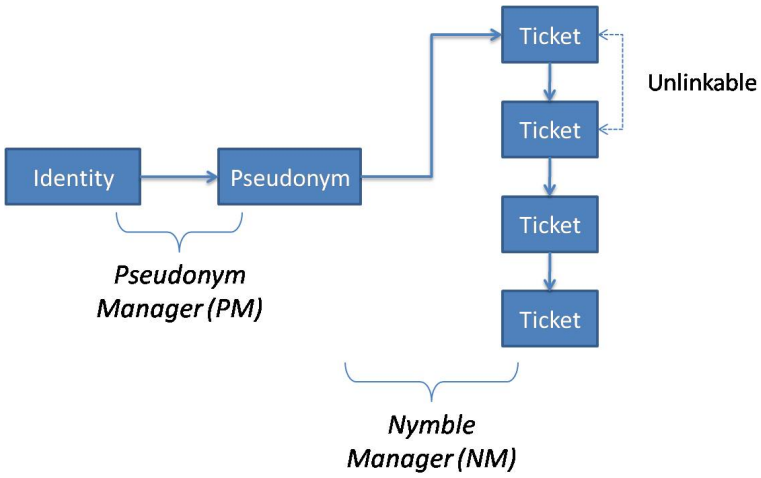

Figure 4.1: The NYMBLE Architecture.

they are indeed banning the specified users. Of course, there is little incentive for PM or NM to misbehave and if web services did not trust the servers, they could simply fetch a list of the anonymity network's servers and ban all the associated IP addresses. Thus for the application NYMBLE is addressing, it is a suitable and commendable solution.

4.6 Credential-based Nyms

The requirements for solving the motivating problem of this chapter require an increase in robustness over that offered by the NYMBLE architecture. If an anonymity network is going to make the legal claim that it has banned an IP address from using the service, it should be able to prove this to a court of law. It cannot trace a ticket to an IP address or it would break the anonymity of the user. However it should be able to respond to certain challenges with proof of action. Consider the following: law enforcement requests that a malicious user is banned from using Tor. After the request is made, a user of the anonymity network continues to perform malicious actions similar to the ones of the supposedly banned user. It is, of course, possible that it is the same user operating from a new IP address. But it is also possible that the anonymity network did not comply with the request. If supplied with two tickets, the anonymity network should be able to prove they did not originate from the same IP address without revealing the IP address of either. It should also be able to prove that an IP address inside a ticket is equal to

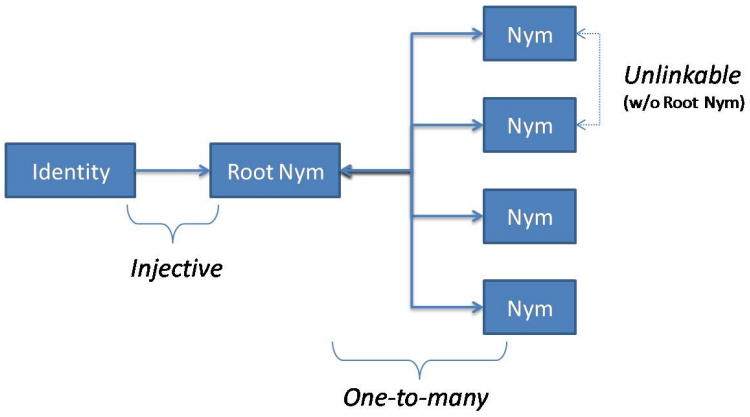

Figure 4.2: An Architecture for Anonymous Nyms.

that of an IP address on the ban list.

In the previous chapter, we introduced a method for distributing digital credentials containing Alice's IP address. Her signed credential is of the form $\langle I, c, r \rangle$. $I = g^x h^\alpha$ is the credential; g and h are public parameters, x is the attribute in the credential (in this case, Alice's IP address), and α is her private key. c and r are used to verify the issuer's signature on the credential. We also proposed that the issuer of this credential could be law enforcement itself. The credentials issued by this server are verified by Alice to be properly formed, and no information that the server sees can be used to link Alice's final credential and proof to any exchanges during the issuing protocol. For these two reasons, there are no reasonable grounds to object to receiving a credential from this server, even if the user does not trust it.

4.6.1 Nym Issuing Protocol

Figure 4.2 shows an alternative architecture for a nym issuing service. Since the user is already authenticating by IP address with law enforcement in the ENR protocol of Chapter 3, we suggest that law enforcement performs the first injective function as well. We will refer to this server as the Authentication Server (AS). AS generates $f(x)$, where f is a secret injective function—it could be a keyed hash or an encryption. In our terminology, we refer to the value of $f(x)$ for a given x as a Root Nym. The same x value will always produce the same root nym, and given a root nym, it is infeasible to invert f and recover x. The root nym will be used to generate many unique and

unlinkable nyms by a second server.

During the circuit creation protocol in Tor, Alice contacts AS and requests a root nym. AS gives Alice $\langle f(x), sig(f(x))\rangle$. Alice then contacts a second server, the Access Control Server (ACS) through the anonymity network so that ACS cannot see her IP address. ACS is run by someone affiliated with Tor, perhaps by the same server that offers the directory of server nodes in the network or by Tor's host, the Electronic Frontier Foundation. ACS is a trusted third party, and in particular it should be trusted not to collude with AS. Alice provides $\langle f(x), sig(f(x))\rangle$ to ACS, and requests a batch of nyms. ACS forwards $\langle f(x), sig(f(x))\rangle$ to AS through its own secure channel with AS. AS issues a batch of unique digital credentials containing $f(x)$ as its attribute to ACS using the Issuing Protocol in the previous chapter (replacing x with $f(x)$ so that $I = g^{f(x)}h^{\alpha}$). Note that in the previous chapter, the credentials were issued to Alice. Here they are being issued to ACS and so ACS is the holder of the credential and its corresponding secret key, and thus ACS is the only entity that can prove properties about the credential. These credentials are unblinded by ACS and given to Alice as her nyms. AS cannot trace a nym to a root nym because of the blinding factor in the issuing protocol, and ACS cannot trace a root nym to an IP address because it does not know the secret injective function f. As long as the two servers do not collude, Alice's anonymity is ensured. Alice can request that ACS engages in a showing protocol to prove to her that her nym is correctly formed before using it. For the sake of practicality, Alice could audit a random subset of the credentials or only audit them if she has a problem with a past credential. The showing protocol is given in Algorithm 5.

Algorithm 5: Signed Proof: $f(x) = y$

Input: $\langle g_0, g, h, p\rangle$, $\langle I, c, r\rangle$, $f(x)$, y, n.
Output: TRUE or FALSE.

ACS should:
- Choose random secret $w_1 \leftarrow_r \mathbb{Z}_p$.
- Compute $a = g^{w1}h^{w2}$.
- Compute $c_1 = \mathcal{H}(a, n)$.
- Compute $r_2 = c_1\alpha + w_2$.

end

Alice should:
- Compute $c_1 = \mathcal{H}(a, n)$.
- Verify $I^c a = g^{yc_1}h^{r_2}$.

end

This algorithm is quite similar to the showing protocol in the previous chapter, except that it is proving a different property of the attribute $f(x)$. N_3 retains a copy of the nym as a condition for the successful creation of the circuit, and forwards it with each new connection to the recipients of Alice's anonymous messages. When a user runs out of nyms, she can use her last nym to be issued more nyms instead of going through the verification process with AS.

If law enforcement determines that a malicious action was performed by a Tor user, it retains the nym associated with the action. Since AS is run by law enforcement, and AS issued the nym and signed it, it can be confident that the nym contains a valid root nym without having to trust Alice or ACS. Furthermore, since it calculated the root nym from the IP address, it can be confident that the nym is ultimately based on a valid IP address without trusting Alice or ACS. There is no way for ACS to cheat and give Alice a nym that does not correspond to her root nym as determined by AS. Therefore from the perspective of law enforcement, it has sole responsibility for the integrity of the nyms.

Law enforcement then provides the nym to ACS and requests that the IP address associated with the nym be banned from Tor. ACS can determine the root nym from the credential, although this requires it to keep a log of $\langle f(x), \alpha \rangle$ indexed by $I = g^{f(x)}h^{\alpha}$ for each nym it issues. It may choose to encrypt $\langle f(x), \alpha \rangle$, and give it to Alice as part of her nym. It can use a randomized encryption scheme, such as El-Gamel [54], which ensures that the same plaintext (*i.e.*, $f(x)$) produces a unique ciphertext every time it is encrypted. Under this method, ACS does not need to keep logs, only a secret decryption key. Once it has recovered $f(x)$, it creates a new credential $I_b = g^{f(x)}h^{\alpha_b}$ for the ban list. The ban list can be published, although only ACS can link a credential on the ban list to a nym. ACS can use the same α_b for every entry on the ban list.

If law enforcement requests that nym I_1 be banned and then has a reasonable belief that a second user with nym I_2 is the same user, it can provide AS with I_1 and I_2. ACS cannot show the root nym in either of the credentials or AS could trace the root nym to an IP address. However ACS can prove that the attributes are not equal without revealing them if it can place both attributes into a single credential. It can do this, in a way that AS can verify, by computing $I_1 \cdot I_2 = g_1^{x_1} g_2^{x_2} h^{\alpha_1} h^{\alpha_2} = g_1^{f(x_1)} g_2^{f(x_2)} h^{\alpha_1+\alpha_2}$. This new credential is essentially a credential with two attributes—the IP addresses of both parties—and a new secret key $\alpha = \alpha_1 + \alpha_2$. Given that ACS knows α_1 and α_2, it can easily calculate the new secret key. With knowledge of the secret key, it can then issue a

showing proof that the root nyms are not equal: $f(x1) \neq f(x2)$. The showing protocol is shown in Algorithm 6,

Algorithm 6: Signed Proof: $f(x_1) \neq f(x_2)$

Input: $\langle g_0, g, h, p \rangle$, $I_1 = g^{f(x_1)}h^{\alpha_1}$, $I_2 = g^{f(x_2)}h^{\alpha_2}$, $f(x_1)$, $f(x_2)$, n.
Output: TRUE or FALSE.

ACS should:
- Compute $\alpha = \alpha_1 + \alpha_2$
- Compute $I = g_1^{f(x_1)} g_2^{f(x_2)} h^{\alpha}$
- Choose random secrets $w_1, w_2 \leftarrow_r \mathbb{Z}_p$.
- Compute $a = I^{-w_1} g_1^{w_1} g_2^{w_1} h^{w_2}$.
- Compute $c_1 = \mathcal{H}(a, n)$.
- Compute $\epsilon = f(x_1) - f(x_2)$.
- Compute $\delta = \epsilon^{-1}$.
- Compute $r_2 = c_1\delta + w_1$.
- Compute $r_3 = c_1\alpha\delta + w_2$.

end
AS should:
- Compute $c_1 = \mathcal{H}(a, n)$.
- Verify $I^{r_2}a = g_1^{r_2+c_1} g_2^{r_2} h^{r_3}$.

end

After ACS puts $f(x)$ on the ban list, it can engage in a simliarly structured protocol to show AS that the root nym in the nym requested be banned is the same as the root nym in the credential it appended to the ban list. ACS simply takes the credential on the ban list and multiplies it with the nym requested by law enforcement to be banned, and issues a signed proof that $f(x_1) = f(x_b)$. This is shown in Algorithm 7.

4.7 Concluding Remarks

In Chapter 1, we defined anonymity, pseudonymity, and veronymity using the two propositions,

P1: an action is not linkable to the identity of the actor.

P2: two actions performed by the same actor are not linkable to each other.

In this chapter we introduce a method to ban users from Tor. Since banning a user inherently requires **P2** to be false, there is a tension between anonymity and reputability.

Algorithm 7: Signed Proof: $f(x_1) = f(x_b)$

Input: $\langle g_0, g, h, p\rangle$, $I_1 = g^{f(x_1)}h^{\alpha_1}$, $I_b = g^{f(x_b)}h^{\alpha_b}$, $f(x_1)$, $f(x_b)$, n.
Output: TRUE or FALSE.

ACS should:
- Compute $\alpha = \alpha_1 + \alpha_b$
- Compute $I = g_1^{f(x_1)} g_2^{f(x_b)} h^{\alpha}$
- Choose random secrets $w_1, w_2 \leftarrow_r \mathbb{Z}_p$.
- Compute $a = g_1^{w1} g_2^{w1} h^{w2}$.
- Compute $c_1 = \mathcal{H}(a, n)$.
- Compute $r_2 = c_1 x_b + w_1$.
- Compute $r_3 = c_1 \alpha + w_2$.

end

AS should:
- Compute $c_1 = \mathcal{H}(a, n)$.
- Verify $I^c a = g_1^{r_2} g_2^{r_2} h^{r_3}$.

end

	AS - Trustworthy	AS - Untrustworthy
ACS - Trustworthy	Anonymity	Anonymity
ACS - Untrustworthy	Pseudonymity	Veronymity

Table 4.1: Summary of Trust Model for Revocable Access.

However it is possible for these propositions to be true with respect to some individuals and false with respect to others. In our solution, **P2** is only false with respect to the trusted server ACS. With respect to all other entities, including AS, **P2** is true and Alice is anonymous. **P1** is true with respect to all other entities individually; however, it is false if the AS and ACS servers collude. Figure 4.1 summarizes the status of Alice's privacy with respect to the trustworthiness of these two servers.

The main improvement our solution offers over NYMBLE is a protocol for verifying the integrity of the process of having a user banned. NYMBLE has no inherent integrity, and requires the entities requesting the ban to trust the servers involved in the process. Our solution allows ACS to prove to AS that it is performing diligently. Specifically, it can prove that a nym is on the ban list, and it can prove that two separate nyms are not from the same root nym.

The goal of the work in this chapter is to provide a method for Tor to combat adverse selection and moral hazard by banning users who commit malicious acts from using Tor in the future. Since IP addresses change occasionally, a banning policy and dispute resolution policy should be developed to address the permanency of the ban. This method also allows Tor to ban users who violate its terms of service. For example, Tor strongly discourages users from downloading large media files through Tor due to the required bandwidth. This could be made a concrete clause in the terms of service, and then users in violation would be banned—freeing up bandwidth for other users.

Chapter 5

The Deployability and Usability of Tor

5.1 Introduction and Motivation

Tor is an important privacy tool that provides anonymous web-browsing capabilities by sending users' traffic through an anonymity network of specialized proxy servers designed to unlink the sender's identity from her traffic [37]. Like any application, Tor must be usable in order for it to be widely adopted. To this end, a number of tools have been developed to assist users in deploying and using Tor. By examining the usability of these tools and offering suggestions for their improvement, the motivation of this chapter is to help increase Tor's affability among novice users with hopes of expanding its user base. In this work, we examine the usability of Tor and evaluate how easy it is for novice users to install, configure, and use Tor to anonymise the Firefox web-browser.

As outlined in Chapter 2, anonymity networks mix the sender's internet traffic with traffic from other users so that the true sender of a particular message is indistinguishable within the set of all users. In other words, the sender is anonymous within a crowd [36]. As such, the sender's anonymity is contingent on other users being untraceable. If the other users commit errors that allow them to be traced, the number of users covering for the sender decreases, having a direct effect on the sender's own anonymity. It is thus in the sender's individualistic interest that not only she, but the other Tor users, can properly deploy and use the software—a factor that differentiates Tor from many other security applications and underscores the importance of examining the usability of Tor.

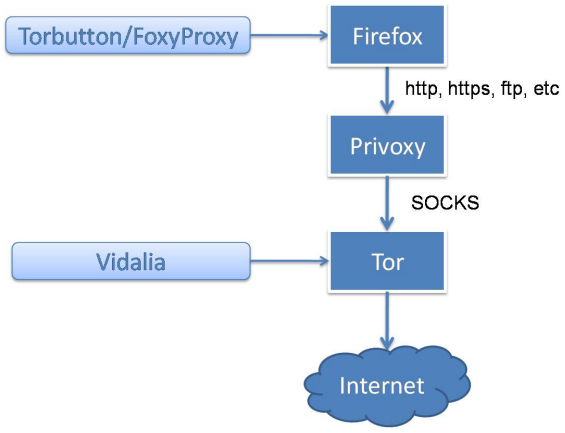

Figure 5.1: A typical Tor setup: Torbutton and FoxyProxy are extensions that can be installed within Firefox. Firefox is then configured to send traffic to Privoxy which forwards it to Tor. The user interacts with Tor through its graphical user interface Vidalia.

As critical as usability is to the successful and wide adoption of Tor, it appears that no extensive usability study has been presented in the literature (prior to [33]). Our first contribution is to compile a set of Tor-relevant usability evaluation guidelines from a variety of sources, eliminate the redundancies, and offer justifications—in some cases, based on research in cognitive psychology not yet applied to usable security and privacy. Our guidelines build on the earlier guidelines proposed to date, including Whitten and Tygar [71] and others, however our guidelines are appropriately shifted in focus from usable security to usable privacy. Using our guidelines, we perform a cognitive walkthrough of the core tasks of installing, configuring, and running Tor.

The Tor application only makes the onions. The application generating the traffic—in this case, Firefox—needs to be configured to direct its traffic into Tor, which is listening on a local port. Furthermore, this traffic is often first filtered through a different application called Privoxy [6] to ensure DNS lookups get captured and anonymised, as well as scrubbing various types of identifying data contributed to the packets from the higher layer protocols in the network stack (in particular, the application layer). Tor is a SOCKS proxy [52] and so a typical configuration will direct http, https, and DNS traffic from Firefox to Privoxy, and SOCKS traffic directly from Firefox to Tor. Privoxy then filters its received traffic and passes it into Tor through a SOCKS connection. We examine manually configuring Firefox for use with Tor [7], Privoxy (a filtering proxy)

[6], and Vidalia (a GUI for Tor) [10]. We also examine two Firefox extensions, Torbutton [8] and FoxyProxy [2], designed to assist the user in performing the key tasks. The relationship of these tools is shown in Figure 5.1. Finally we inspect Torpark [9]—a standalone Firefox variant with built-in Tor support[1]. We uncover numerous usability issues with each deployment option but find that the extensions and Torpark offer some improvement in important areas.

5.2 Usability and Adverse Selection

In July 2007, independent security consultant Dancho Danchev reported on his discovery of a step-by-step guide to deploying Tor. The guide was allegedly published by a Islamic extremist organization.[2] While there is no evidence that this organization has committed any unlawful behaviour using Tor (or in general), it highlights the problem of adverse selection within Tor. Furthermore, it demonstrates that those with incentives to use an anonymity network will overcome any barriers that might discourage typical users from using the software. Clearly the predominant barrier identified by this organization to deploying Tor is usability.

Adverse selection suggests that malicious users have greater incentives to overcome usability problems than average users. It follows that increasing usability will help Tor attract a wider base of users and balance out its selection of users. This is similar to lowering the costs of an insurance policies—we demonstrated how raising the price of premiums exacerbates the adverse selection in Chapter 1, and it follows that lowering prices will have the inverse effect. Decreasing the usability of Tor will eliminate those with the lowest incentives while keeping the malicious users. Thus increasing the usability of Tor will have the opposite effect, and will attract non-malicious users in a higher proportion than the natural selection of users.

[1]Since performing this usability study, Torpark has been renamed XeroBank. We preserve the name Torpark in this chpater given that our comments may be specific to the version of the browser we used in our usability study.

[2]`http://ddanchev.blogspot.com/2007/07/cyber-jihadists-and-tor.html`

5.3 Guidelines and Methodology

5.3.1 Evaluation Methodology

A common usability evaluation method is heuristic evaluation [57]. In heuristic evaluation, a double expert (someone who knows both the application domain and human-computer interaction principles) evaluates the user interface of the application against a set of heuristics. We will employ a related method: cognitive walkthrough [70], as has been used previously by others [71].

The premise behind a cognitive walkthrough is that users learn by exploring the software interface. They often do not sit down and learn the software interface *a priori* through a user manual or by systematically visiting every nook and cranny of the user interface. Rather they attempt to perform a task and rely on the interface to intuitively guide them through that task. The user is presumed to be pragmatic and she does the minimal amount of work to satisfy herself that the task is completed.

In a cognitive walkthrough, a set of core tasks is designated. The evaluator will perform these tasks and evaluate the usability of the application against a set of evaluation guidelines as these tasks are being performed. Thus it is similar in methodology to heuristic evaluation, but differs in the approach—instead of systematically exploring the user interface, the evaluator performs core tasks.

Cognitive walkthroughs are an important first step in evaluating the usability of a software tool. Conducting studies with users, either in a laboratory or in the field, is a common follow-up. However due to time constraints and limited budgets, user studies are typically constrained to a narrow set of tasks. By contrast, the power of the cognitive walkthrough is the breadth that it makes possible. In this work, we examine four different deployments of Tor, from installation to configuration to actual use, and perform comparative analysis. Such a wide comparison would be extremely challenging, if not impossible, in a typical user study. A user study is most appropriate when one or two predominant solutions emerge from a large solution set which, as we will see, is not the situation we currently have with Tor.

5.3.2 The Core Tasks

The core tasks to be performed in our cognitive walkthrough are as follows.

CT-1 Successfully install Tor and the components in question.

CT-2 Successfully configure the browser to work with Tor and the components.

CT-3 Confirm that the web-traffic is being anonymised.

CT-4 Successfully disable Tor and return to a direct connection.

The core tasks will be performed on the following four configurations:

1. Firefox with Tor [7], Vidalia [10], and Privoxy [6];
2. Firefox with Tor, Vidalia, Privoxy, and Torbutton [8] (a Firefox extension);
3. Firefox with Tor, Vidalia, and FoxyProxy [2] (a Firefox extension); and
4. Torpark [9] (a self-contained, anonymous browser).

There is considerable overlap between the first three configurations. Both Torbutton and FoxyProxy allow the user to shortcut a few of the steps in configuration 1, but most of the steps are common between them. Thus, the walkthrough of the first configuration will be significantly longer and more involved than that for the second and third.

5.3.3 Usability Guidelines

The set of guidelines that we use to evaluate each of the core tasks are as follows.

G1 Users should be aware of the steps they have to perform to complete a core task.

G2 Users should be able to determine how to perform these steps.

G3 Users should know when they have successfully completed a core task.

G4 Users should be able to recognize, diagnose, and recover from non-critical errors.

G5 Users should not make dangerous errors from which they cannot recover.

G6 Users should be comfortable with the terminology used in any interface dialogues or documentation.

G7 Users should be sufficiently comfortable with the interface to continue using it.

G8 Users should be aware of the application's status at all times.

These guidelines are drawn from a variety of sources [70, 71, 49, 34, 22] and are intended for evaluating Tor specifically. However they are suitably broad and may find application in other usable privacy walkthroughs. We now individually justify the inclusion of each.

G1: Users should be aware of the steps they have to perform to complete a core task.

This is a restatement of the first guideline of Whitten and Tygar [71]. Every user of a new application knows certain things before using the system and learns certain things during the use of the system. In the cognitive walkthroughs we carry out here, the presupposition is that the user knows enough to start the process for each core task—in the case of installation, the user can download the installation file and open it; in the case of configuration, the user can explore the user interface or follow cues. We are evaluating how the application cues the user to perform the intermediary steps between these broadly defined tasks.

G2: Users should be able to determine how to perform these steps.

Once the user is aware of what intermediary steps are necessary, she must be able to figure out how to perform these steps. This is the second guideline in [71]. It is assumed the user has a mental model of how the system works. It is thus important that the system model be harmonized with the user's mental model if the user is to be successful in performing the necessary steps required to complete each core task [70]. What is less obvious is why we cannot fully rely on the user to simply modify her mental model when given conflicting information.

A predominant reason is that humans have a stronger preference for confirming evidence than disconfirming evidence when evaluating their own hypotheses. This cognitive bias is well illustrated by Wason [69], who conducted a study where a set of subjects were given the following sequence of numbers: 2,4,6. The subjects were told that the numbers followed a rule, and their task was to determine the rule by proposing their own sequence of numbers, which would be declared as matching the rule or not. The rule was any ascending sequence of numbers. However most subjects derived a more complicated rule, such as every ascending sequence of numbers differing by two. The point of this test was that the subjects, on average, had a preconceived idea of what the rule was and

only proposed sequences to confirm that rule, instead of also proposing sequences that would falsify their perceived rule.

Confirmation bias is important in usability because it proposes that users are biased toward only seeking and accepting information that confirms their mental model, and thus may avoid or even ignore information that contradicts it. It cannot reasonably be expected that users will easily and quickly adapt their mental model to new information.

A second concern with how users perform these steps is that security is a secondary goal [71, 49]. If the user is given two paths to completing a core task—one that is fast but not secure, and one that is slower but more secure–it cannot be assumed that the user will take the latter approach. In fact, studies in behavioral economics demonstrate that humans often prefer smaller immediate payoffs to larger future payoffs, such as $50 today instead of $100 a year from today [51]. Using software securely has a greater (usually non-monetary) payoff for the user, but this utility has to be substantially higher than the alternative to justify the delay in achieving it.

G3: Users should know when they have successfully completed a core task.

In other words, users should be provided with ample feedback during the task to ensure they are aware of its successful completion. This principle has been proposed in the context of heuristic evaluation [70] and for a cognitive walkthrough [22]. It was also mentioned by Cranor [34]. With Tor it is critical that users be provided with a tool to determine that their traffic is actually going through the Tor network, as there is no easy way for them to determine this for themselves (short of running a packet sniffer).

G4: Users should be able to recognize, diagnose, and recover from non-critical errors.

Users will likely make errors in performing the core tasks and it is important for them to be able to recover from these errors [70]. It is important for users to be given concise error messages.

G5: Users should not make dangerous errors from which they cannot recover.

This guideline is from Whitten and Tygar [71]. Tor has a few dangerous errors associated with it. The first is a false sense of completion, where the user thinks that her traffic has been anonymised when it has not.

Second, in converting domain names into IP addresses, a web browser will often consult a domain name server (DNS) provided by the ISP. These DNS lookups do not happen over the typical port used for web-traffic and so configuring a browser to use Tor as its SOCKS proxy does not result in DNS lookups funneling through Tor. If the DNS look-ups are not anonymised, then the ISP can know when you visit each new site whose IP address is not cached locally in your hosts file if it monitors traffic to and from its DNS.

Finally, it has been shown (e.g. [55]) that allowing client-side execution of mobile code, such as downloading and executing a Java applet, could compromise the user's anonymity when browsing a website. An applet can be created to simply request the user's local IP address using a system call to the kernel, and to pass that information back to the website over regular http (or https if Privoxy is configured to filter out packets containing the user's local IP address). To avoid this dangerous error, users must disable Java. A similar exploit may be possible with ActiveX, Flash, or JavaScript as well (although Firefox does not come with ActiveX support).

G6: Users should be comfortable with the language used in any interface dialogues or documentation.

Wharton *et al.* emphasize that applications should use simple, natural, and familiar language [70]. This is also a central design goal for Privacy Bird [34], which takes natural language privacy policies and translates them from legalese into concise, human readable summaries. In the case of Tor, Privoxy, Torpark, *etc.*, we are looking at the understandability of dialogues and the documentation, and their reliance on technical language which users may have trouble understanding.

G7: Users should be comfortable with the interface.

This is the fourth principle of usable security of Whitten and Tygar [71], and is an essential part of the principal of psychological acceptability quoted by Bishop [17]. While Tor and Privoxy do not have an extensive user interface, we will be evaluating it in terms of how intuitive it is to use. We will also be analysing the interface of the Firefox extensions.

G8: Users should be aware of the system status at all times.

This principle was proposed in the context of heuristic evaluation [70] and cognitive walkthrough [22]. Cranor advocates the use of 'persistent indicators' that allow the user to see privacy information at a glance [34]. In terms of Tor, we are looking for indicators that show Tor is enabled as well as perhaps, for example, the current external IP address of the exit node, the number of users using the system, or other relevant system information.

5.4 Tor Installation

The Tor installation file may be downloaded from the Tor website [7]. The Tor download page may be confusing to a novice user unfamiliar with the open source software movement. Alongside the versions of Tor for other operating systems, there are three versions offered for Windows: a version marked only with its version number, a version marked by its version number and "alpha," and a link to "Windows packages for experts." Each installation file also has a digital signature file. At the top of the page, it is stated which version number is the "stable" release and which is a "development" release. This is a slight breach of G6 for using unfamiliar language. The intention is certainly for non-expert users to install the stable version. A statement at the top of the page designated for non-expert, Windows users and linking directly to the stable version installation file without mentioning version numbers would alleviate potential confusion over which version to download.

The installation process is a fairly straightforward wizard-style installation. The first dialogue screen informs the user that the installation bundle actually installs three distinct applications: Tor, Privoxy, and Vidalia. It is not clear from the installation dialogue what each application does or why the three are needed. The next dialogue alerts users with previously installed versions of any of these applications to exit them before proceeding. We confirmed this is not enforced and it is possible to proceed without exiting.

The next dialogue allows users to choose which components to install. It offers a tree-style list of components that can be expanded or collapsed, and each component has a checkmark that can be selected or deselected. The three top nodes are Tor, Vidalia, and Privoxy. It states that users may "position [their] mouse over a component to see

its description." Given that the user has not been cued as to what exactly Vidalia and Privoxy are, this would be an excellent place to indicate their purpose. However, the description for Vidalia is simply "Install Vidalia 0.0.7," and likewise for Tor and Privoxy with their respective version numbers. The default setting is a full installation, which installs all three components. However given that the user cannot be expected to understand what all three components are, she has the potential to make an error at this step. But she may recover from the error from simply rerunning the installer if she realizes she has not installed a required component. For this reason, it does not violate G4, although it does violate G1.

After specifying a directory for installation (a default is given: the Program Files directory in the root drive), the components will install. The final dialogue announces that installation is successful and informs the user, "Please see http://tor.eff.org/docs/tor-doc-win32.html to learn how to configure your applications to use Tor."

5.5 Tor, Vidalia, and Privoxy

The task of configuring Firefox to use Tor and Privoxy is simply impossible without using the aforementioned documentation page. While ideally, from an HCI perspective, it would be possible for users to execute core tasks from simply exploring the application and its interface, configuration is not confined to a single application. It requires steps to be performed within Firefox itself, as well as within Tor and Privoxy, and thus will require inter-application documentation. It cannot be accomplished merely with intra-application cues.

The configuration document is concise and broken into steps, and so it is reasonable that the user will use the documentation. The first step is downloading and installing the application, which is already completed. It does however inform the user of how Vidalia and Privoxy are different from Tor: Vidalia is a graphical user interface for Tor and Privoxy is a filtering web proxy. This information is essential for the user to have a successful installation. The fact that this information is divulged after the installation process violates G1. It should be included during the installation process, as mentioned previously.

Step two is concerned with configuring Tor. Given that Tor can be used to anonymise any web application, not just a browser like Firefox, the configuration page links to a separate page ("How to Torify") devoted to a list of applications and step-by-step

instructions for each. However the configuration page does offer some high-level advice. First it states, "using Privoxy is necessary because browsers leak your DNS requests when they use a SOCKS proxy directly, which is bad for your anonymity." This statement unfortunately uses unfamiliar language (G6), however its intention is to prevent the dangerous error of DNS leaks (G5). The novice user does not need to understand what DNS is or what ports and sockets are to prevent this dangerous error—so long as they understand that the steps they need to take are preventing a dangerous error (G2).

The configuration page also offers this advice: "to Torify ... applications that support HTTP proxies, just point them at Privoxy (that is, localhost port 8118)." For the expert user, this information is concise and succinct and may suffice for them to configure their application without referencing the 'How to Torify' page. Privoxy exists on local port 8118 and Tor exists on local port 9050. However Tor only accepts SOCKS connections and thus http (port 80), https (port 443), and DNS (port 53) traffic has to be filtered through Privoxy and into Tor.

With this information in hand, it is reasonable that the user will move to the 'How to Torify' page to learn how to configure Firefox. While the configuration page is concise and fairly non-technical, the 'How to Torify' page is dense, full of technical jargon, and includes information extraneous to the task of configuring the application for Tor. The Firefox entry includes two configuration methods: one that involves changing the connection options in Firefox, and a second that involves editing the actual configuration of Firefox (a daunting task) but this second method then alleviates the need for Privoxy. This information is technical, confusing, and the second method directly contradicts the configuration page that states Privoxy is necessary (with "necessary" in bold). Furthermore, the advanced method is presented first. The documentation violates G2 because its unclear how the user should proceed, and it violates G6 for all the technical jargon. This is the first point in CT-2 where we heavily suspect users will give up or start to make errors.

The intention appears to be that novice users will use the second method to configure Firefox. Assuming they can determine this is the step they need to take to complete the task, the instructions are rather straightforward. It gives the series of menus the users need to visit: "Tools → Options → General → Connection Settings → Manual proxy configuration" and instructs the user to "set HTTP Proxy 127.0.0.1 (or localhost), port 8118 and tick the box [X] Use for all protocols. Or you may explicitly set the Proxy information for SSL, FTP, and Gopher to localhost/8118 and then set the SOCKS

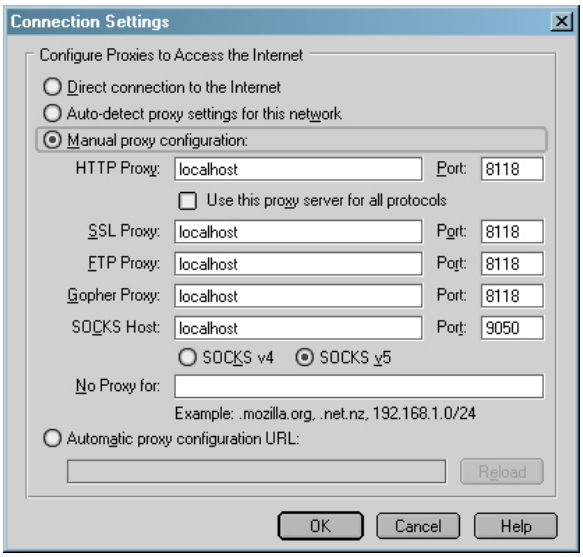

Figure 5.2: Connection Settings in Firefox.

Host information to localhost/9050, making sure to specify SOCKS v5." This appears complicated but referring to the configuration window in Figure 5.2, it is simply a matter of filling in the boxes. The problem with these instructions is that they give two options ("use the same proxy" box or individually configure) with no indication as to which the user should use. This violates G2. The options are not equivalent—the former uses Privoxy as the SOCKS host, the latter uses Tor—but will result in the same behaviour. However the user has no indication of which is better or which one they should use. They may choose the first option simply because it is given first, or they may choose the second because there is a graphic on the page illustrating what the correct settings look like—essentially the same as Figure 5.2. Note that Privoxy is needed either way since most data to be anonymized will be HTTP and SSL traffic.

For this configuration page to meet G2 and G6, it should pick one configuration method, place it at the top of the section, and simply give instructions on how to perform it without the technical language or extraneous information. The rest of the configuration information can be left as-is, but could be placed under an advanced configuration subheading within the section. Alternatively, the proxy settings could be encoded into a proxy auto-configuration (PAC) file. This would still require the user to find the Connection Settings in Figure 5.2, but the user would only have to fill in one box ('Automatic proxy configuration URL') which would result in a simpler set of instructions.

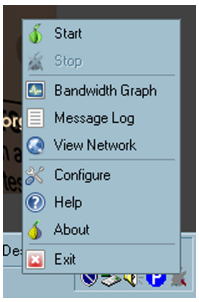

Figure 5.3: Vidalia Menu.

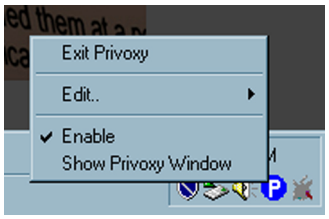

Figure 5.4: Privoxy Menu.

Presuming the user can configure Firefox, she can proceed to step three on the configuration page which is "making sure it's working." The first instruction is to ensure Privoxy and Vidalia are running. The correct configuration has these two applications running, however recall that the Tor application was also installed and it has an executable. It is not intuitive that Privoxy and Vidalia need to be run, but not Tor (Vidalia calls the Tor application). There are three ways to get these two applications running. By default, they both run after the installation process. Also by default, they both run when the computer boots up. If either of these defaults is changed, they can be run from the executables in their respective directories or in the start menu. By having the defaults set this way, less demand is placed on the user and so this is a commendable approach. However by default, Tor is not enabled while Privoxy is. The fact that the user needs to enable the one application to start and not the other is inconsistent. There is an option to start Tor when Vidalia starts, and this option should be made the default.

At this point, the user has Tor running but not enabled (Figure 5.3), Privoxy running and enabled (Figure 5.4), and the Firefox connection settings configured. To complete CT-2, the user simply needs select "Start" from the Vidalia menu (Figure 5.3) and begin browsing. If the user does not start Vidalia, or forgets to, the browser will display the error message in Figure 5.5 when the user attempts to visit a website. Given that Privoxy

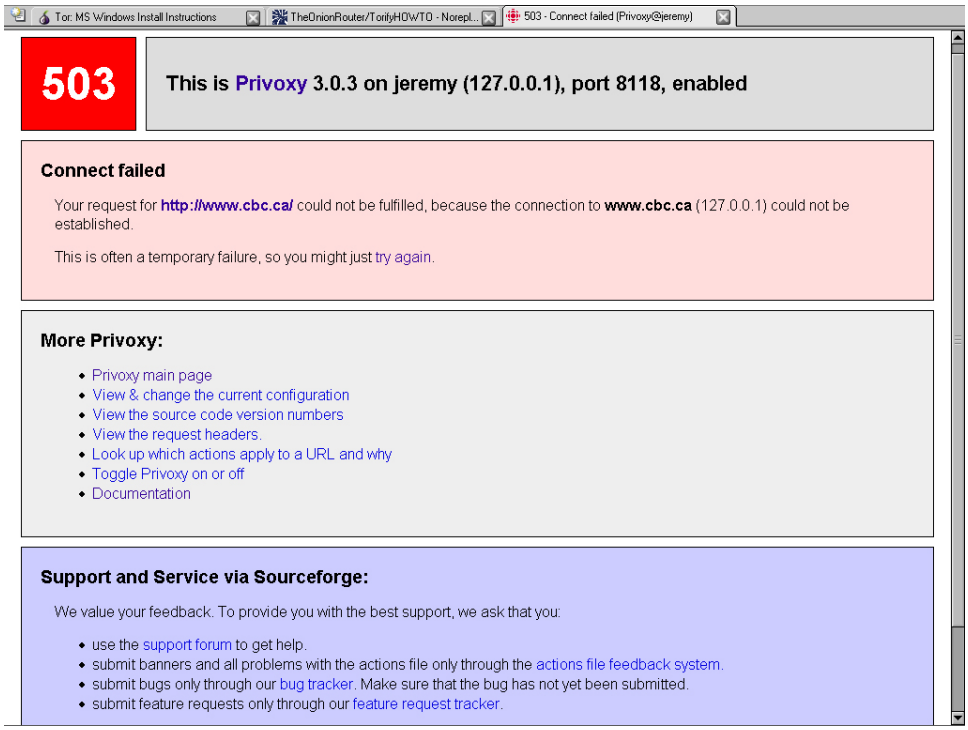

Figure 5.5: Privoxy Error Message.

is a standalone application that can be used independently of Tor, it will not give Tor related error messages. It is not clear from this error message how the user can recover; furthermore, the links to the documentation and support are all links to webpages, and given that the error is being generated because the internet is not accessible, clicking on them will simply generate the same error. This is not a dangerous error (G5), but it is one that does not provide sufficient information for the user to recover from and thus violates G4.

By selecting Start from the Vidalia menu, the Tor taskbar icon changes from a red onion with an X through it, to a yellow onion signifying that Tor is starting, to a green onion signifying that Tor is running properly. This two-factor visual cue provides feedback (G3) that would allow most users to determine the system status at all times with a quick glance at the task bar (G8). The colours used are consistent with traffic light colours, and for users with colour vision deficiencies, the placement of the X over the icon is suitably noticeable. Overall, it is a comfortable interface to use for this core task (G7). When webpages are being accessed, the Privoxy icon spins like a radar screen. This provides feedback to the user that Privoxy is filtering traffic, and meets G3.

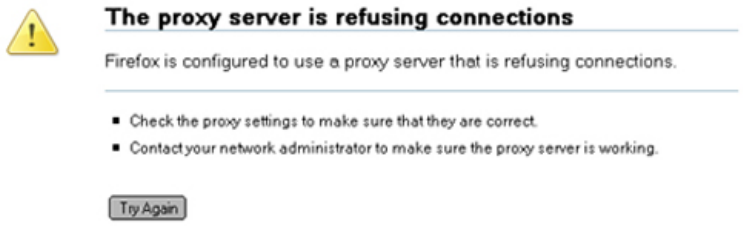

Figure 5.6: Firefox Error Message.

The third core task is to verify that traffic is being anonymised successfully. The configuration page offers a link to a "Tor detector" webpage, which checks the IP address of the packet it receives against a list of Tor servers. If it matches, it announces the configuration was successful and gives you the IP address and server name of the last mix proxy (called the exit node) in the network. This meets the feedback standard of G3 and does so without using unfamiliar language, meeting G6 as well. If Tor is not being used, the website tells the user they are not using Tor and refers them to the configuration website. This is a concise error message that provides enough information for the user to recover from the error, and thus meets G4.

While not a core task, a secondary but important task is determining the size of the anonymity set for the Tor servers the user is connected to, for reasons outlined in Section 2. This task is not possible. Vidalia does offer a View Network option (see Figure 5.3) that presents a dialogue screen which lists the servers, displays the server locations on a world map, and gives information about the server: server name, location, IP address, platform, contact information, bandwidth, and uptime. To allow users to successfully complete this secondary task, information about the number of users connected to the servers should be displayed as well.

The final core task is disabling Tor. Unfortunately, the configuration page and the 'How to Torify' page are both silent on this issue. The correct method is to change the connection settings in Firefox (Figure 5.2) back to "Direct connection to the Internet." However this information is not accessible in the Tor documentation, breaking G2. Intuitively, the user may try a few alternatives to accomplish this task. The first would be to disable Tor by using the Stop option in the Vidalia menu (Figure 5.3). This of course, will generate the same error discussed above and shown in Figure 5.5. The second alternative would be to disable Privoxy by unchecking enable in the Privoxy menu

Figure 5.7: Torbutton Icon in Firefox Status Bar.

(Figure 5.4). This generates the error shown in Figure 5.6 in the browser window, which is a generic Firefox error when Firefox cannot reach a proxy server. Neither provides enough information to allow the user to recover and thus violates G4. The browser is also completely useless in this stage, which will surely frustrate the user. The documentation should make the disabling process explicit, as users cannot be expected to know how to disable Tor once it is enabled.

5.6 Tor, Vidalia, Privoxy, and Torbutton

On the Tor configuration page,[3] under "Step two: Configure your applications to use Tor," the authors of the documentation offer an alternative to manually configuring Firefox for Tor (the configuration examined in the previous section). The page links to Torbutton [8], a Firefox extension, and instructs the user to visit the extension's page, install the extension, and restart Firefox. Extensions are small add-ons for Firefox that add functionality to the browser.

To perform the first core task, the user installs the Tor, Privoxy, and Vidalia bundle as in the previous section. However instead of configuring Firefox as described above, the user will click the link to Torbutton on the configuration page and be taken to the extension's listing in Mozilla's Firefox add-on directory. There is a predominant "Install now" button on the page and clicking it brings up a Firefox dialogue window warning the user to only install software from trusted sources. Clicking "install now" in this window brings up Firefox's list of installed extensions and informs the user to restart Firefox to use this extension. This process is straightforward, and conforms to G1, G2, and G6.

When Firefox is restarted, a cue is added to the status bar that states "Tor Disabled" in red letters, as shown in Figure 5.7. Many extensions install cues in this location, and so assuming the user is comfortable with Firefox, the user will likely notice the cue. Users unfamiliar with Firefox will need to read the documentation linked to from the

[3] http://tor.eff.org/docs/tor-doc-win32.html

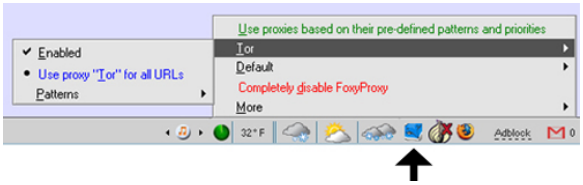

Figure 5.8: FoxyProxy Icon in Firefox Status Bar.

configuration page. From a usability perspective, this has to do with Firefox's interface and is outside of the evaluation of Torbutton itself.

At this point, the first core task is complete. This allows the user to avoid the confusing configuration step criticized in the previous section. To complete CT-2, the user must be running Vidalia and Privoxy and have them both enabled. The analysis of this step in the previous section applies here. However instead of changing the configuration settings in Firefox, the user simply locates the Torbutton cue in the status bar and clicks it once. The cue will change to "Tor Enabled" in green lettering. The status can be seen at a glance, which is consistent with G8. CT-3 will proceed as in the previous section. To disable Tor and return to direct browsing, the user simply clicks the Torbutton cue a second time and it returns to "Tor Disabled." It is unclear whether users can be expected to know to do this (G2), however the change in icon is sufficient for meeting G3.

CT-2 is greatly simplified using Torbutton. The user does not have to change any settings within Firefox or wade through the confusing documentation. CT-4 is also simplified, however Torbutton still does not necessarily eliminate the non-critical errors discussed in the previous section—disabling Vidalia or Privoxy in trying to turn off anonymous browsing, instead of clicking the Torbutton cue within Firefox. However, while it is unclear in the previous section whether users will know enough to go back into the connection settings in Firefox and determine which setting to change to in order to disable Tor, we think it is reasonable to expect the user to at least stumble upon the correct solution with Torbutton even if its not the most intuitive step to completing CT-4.

5.7 Tor, Vidalia, and FoxyProxy

FoxyProxy [2] is a Firefox extension similar to Torbutton but it provides more advanced functionality. It allows users to quickly enable and disable the use of any proxy server,

not just Tor, and it allows the creation of custom access control lists to specify that certain domains always be accessed through Tor, with a different proxy, or directly. As we are testing the core tasks only, we will be testing FoxyProxy only as a method of configuring Firefox to work with Tor and we will not be evaluating its advanced functionality. The other feature that FoxyProxy prominently advertises is that it does not need Privoxy to work. Torbutton can also be configured to not use Privoxy—it is an unsung option in its preferences. However this option was not described on Tor's configuration page (nor explicitly on Torbutton's page) and the user would not likely be aware of it in performing the core tasks the walkthrough is evaluating. For this reason, we evaluated the configuration and use of Torbutton with Privoxy. However we will evaluate FoxyProxy without Privoxy.

CT-1 for FoxyProxy is similar to Torbutton. The user goes to the Firefox add-ons directory, installs the extension, and restarts the browser. However upon restart, the user enters a set-up dialogue to assist with the second core task. The first window asks if the user would like to configure FoxyProxy for use with Tor. If yes, the next window asks if the user is using Tor with or without Privoxy. Proceeding without Privoxy, the next window asks for Tor's local port number and states, "if you don't know, use the default," which is port 9050. The next window asks: "Would you like the DNS requests to go through the Tor network? If you don't understand this question, click yes." The next screen has the advanced filter settings, and the final window alerts the user to make sure Tor is running (G1).

This dialogue solves many of the usability problems encountered in the past sections. The user knows explicitly that Privoxy is not needed. The port and DNS settings both have a default setting and the dialogue clearly tells the user what to do if she does not understand the unfamiliar language (G6). The DNS setting allows the user to avoid a dangerous error (G5) even if their mental model is not sufficient to understand DNS (G2). The user is also reminded to ensure they are running Tor (G1), avoiding a non-critical error (G4), although no indication is given as to how to run Tor (G2)—in fact, the user likely wants to run Vidalia and not Tor.

The filtering options are complicated, however the user can avoid them and enable FoxyProxy to run all traffic through Tor. Enabling Tor for all URLs is more complicated than with Torbutton. The user has to click through a few menus (see Figure 5.8). However the user does not need to use Privoxy, which simplifies the second core task. Core task 3 is no different. Disabling Tor with Privoxy involves entering the menu

and choosing "Completely disable FoxyProxy." The same pitfalls as described in the Torbutton section apply to this core task.

5.8 Torpark

Torpark [9] is a standalone anonymous browser based on Firefox with built-in support for Tor. It is designed to run off a USB token so that a user can run it on a public computer that has an accessible USB port. Unlike the Tor bundle which had multiple versions, the Torpark webpage has only one version which is clearly marked (it also offers the source code, but this is in a distinct section). The installation is a self-extracting archive, so the user simply specifies the directory to which to extract the application folder. The installation process is far simpler: download the clearly marked installer, extract to a folder, and the user is done. The first core task meets all the relevant guidelines.

To run Torpark, the user opens the executable. Before opening, a warning message is displayed which states, "Torpark secures the anonymity of your connection, but not the data you send. DO NOT [sic] use identity compromising information such as your name, login, password, etc. unless you see a closed padlock icon at the bottom status bar of the browser. Torpark should not be run on untrusted computers, as they may have malware or keystroke logging software secretly installed." This warning helps the user understand the system model of Tor (G2), which they may have misconceptions about if their mental model expects, say, that the packets will be filtered as with Privoxy. The language is relatively clear and non-technical (G6), and should be understandable to an intermediate-level internet user.

A second dialogue window opens informing the user that Torpark is establishing a connection to the Tor network. After connecting, the browser itself opens. The browser comes with several preinstalled extensions. One is Torbutton, which is enabled by default and does not use Privoxy by default. This settings allow the user to prevent the dangerous error of DNS leaks without needing Privoxy. A second extension is NoScript. This extension prevents websites from running Java applets and scripting by default. This is very important because it helps prevent the other dangerous error associated with Java applets (G5). A notification is displayed when a site attempts to run a client-side script or an applet that is blocked, with the ability for the user to add the site to a safe list (among other options). However blocking applets is a necessary but not sufficient measure if users are given the option to whitelist sites. The user may disable NoScript or

whitelist everything indiscriminately if the danger is not adequately communicated. If the warning message was amended to include a warning about applets, Torpark may help prevent this dangerous error. There is, however, a flip-side to having NoScript enabled by default. It introduces a new interface that the user must be able to interact with—the usability of which has not been studied and is outside the scope of this work. If we assume the worst case where the user simply ignores the prompts, many modern webpages will not function correctly without client-side scripting. This creates new usability problems and a negative incentive for using Torpark.

Another extension displays your external IP address (*i.e.*, the IP of the exit node) in the status bar, which is a persistent indicator of the system status (G8). However this requires the user to know what their actual IP address is to determine that the displayed IP is different and Tor is working correctly. Thus it is not enough to complete core task 3. In fact, the user is not given any cues that would allow her to confirm that her traffic is being sent through Tor correctly, which violates G2 and G3. She could, of course, visit the Tor detector website mentioned above, which would provide the feedback she is seeking. Thus, if Torpark's default homepage was the Tor detector or if it offered a link to it, then CT-3 could be completed very easily.

Like in the previous sections, determining the anonymity set cannot be completed with Torpark. And given that Torpark uses Torbutton by default, CT-4 is the same as in the Torbutton evaluation with one notable difference: the Tor application is not running and so the user will not get confused and disable Tor in the application instead of the browser. Thus it prevents some of the non-critical errors uncovered is the previous sections.

5.9 Comparison and Summary of Results

We have performed a cognitive walkthrough for four different configurations of a Firefox-based browser with Tor. The first three configurations are largely interchangeable. The fourth, using the standalone browser Torpark, may not be as desirable to users who want to anonymise more than their browser. Tor can be used to anonymise instant messaging, file sharing, email clients, or nearly any application that uses the internet. If the user is going to anonymise more than their browser, they will need to install Tor, Privoxy, and Vidalia anyway—and therefore they would probably find it more convenient to simply configure Firefox to use these applications than to use an additional browser.

	Installation	Configuration	Verification	Disabling
Manual Config	Difficult	Very Difficult	Easy	Very Difficult
Torbutton	Difficult	Easy	Easy	Very Easy
FoxyProxy	Difficult	Very Easy	Easy	Easy
Torpark	Very Easy	Very Easy	Very Difficult	Very Easy

Table 5.1: Summary of Usability Results.

The summary of the results for the four core tasks are in Table 5.1. Of the first three configurations, manually configuring Firefox has the most usability problems. We found problems with the documentation and a complete lack of instructions on how to disable Tor. Furthermore, even if the user figures out how to enable and disable Tor, the options are in a menu and several windows away. There is also no cue within the browser as to whether Tor is enabled or disabled at any given time, however the documentation links to a Tor detector webpage that verifies your traffic is being routed through the Tor network.

With Torbutton and FoxyProxy, it is conceptually more intuitive how to enable and disable Tor, it is a quicker process with a better interface, it does not require the user to read documentation, and there is a persistent indicator that shows the enabled/disabled status of Tor at all times. Thus Torbutton and FoxyProxy beat manual configuration on the second and fourth core tasks based on G1, G2, G3, G6, G7, and G8. Manual configuration has no advantage over Torbutton or FoxyProxy on any of the core tasks.

Between Torbutton and FoxyProxy, we found Torbutton to have the better interface. The user has to simply click it once to enable or disable Tor; the user has to go through a menu to achieve the same with FoxyProxy. Therefore Torbutton is better on the fourth core task based on G7. Both have persistent indicators in a visual cue on the status bar that changes colour according to the settings, and so they tie in terms of G8. However FoxyProxy has an advantage over Torbutton in the configuration dialogue by walking the user through the steps (instead of assuming the defaults) and imparting information to the user about the dangerous error of DNS leaks, in that Privoxy is not needed, and reminding the user to ensure Tor is running. Therefore FoxyProxy is better than Torbutton on the second core task based on G2, G4, and G5. Note however that Privoxy does offer features (*i.e.*, packet scrubbing) that FoxyProxy does not, and so

an unarticulated security/usability trade-off is at play in requiring the user to choose between using Privoxy or not.

Torpark has many advantages over using any preceding combination of Firefox, Tor, Privoxy, Torbutton, and FoxyProxy. First it does not require any documentation to install or configure. Assuming users do not like to read documentation, this is a huge advantage over using the components separately. The set-up dialogues are well written and make clear the limits of Tor in a hostile environment. None of the first three configurations do anything to thwart the dangerous error of allowing the execution of Java applets, while Torpark has Java disabled by default through the NoScript extension. While this is better in terms of G5, NoScript also disables JavaScript may also present new usability problems given that the many webpages which use client-side scripting will now be broken. Where Torpark fails, and the first three configurations succeed, is that it does not offer any way to verify that traffic is going through the Tor network aside from displaying the external (exit node's) IP address, which may be sufficient for advanced users but certainly not for novice users.

5.10 Concluding Remarks

The two configurations that use the Firefox extensions are clearly more usable than manual configuration. However both Torbutton and FoxyProxy have their advantages and disadvantages. Our recommendation would be to add the set-up dialogue of FoxyProxy to the user interface of Torbutton. This synthesised extension would have the same one-click interface as Torbutton but would go through a small dialogue the first time it is run asking the user whether she wants to use Privoxy, giving her the option of eliminating DNS leaks (using the same style of language as FoxyProxy), and reminding her to ensure Vidalia is running. We would also recommend that it includes a warning similar to the one provided in Torpark.

These recommendations expose the difficulty of helping users prevent dangerous errors without introducing terminology and concepts that are too difficult for the novice user to understand. This consideration raises the question of who Tor's target user is. Our results show that novice users will have difficulty with at least one of the core tasks no matter which current deployment option they choose. Furthermore, the technical jargon and unfamiliar language of Tor's documentation is clearly not meant for the novice user. Is Tor simply too complicated for novice users to understand? Perhaps,

but Tor might be made *more* accessible to novice users with two-tier documentation and properly structured defaults—simple instructions followed by instructions for advanced users, and configuration dialogues like FoxyProxy's, with statements that if a user does not understand a concept, she should choose a default.

The extensions eliminate the need for the user to consult the 'How to Torify' page for configuring Firefox, but we would still recommend that two major changes be made to this page. The first is that one of the three different configuration options is chosen and placed at the top of the section on Firefox, and this becomes the dominant way of configuring Firefox for novice users. The second recommendation is that instructions are added to explain how to disable Tor.

None of the configurations allow the user to see the size of her anonymity set. We would recommend that Vidalia add information to its 'View Network' window about the number of users connected to each node in the network, although we recognize this metric may not be meaningful to the novice user. Java Anon Proxy (JAP) [16] has the visual cue of an anonymity meter that moves from low to medium to high, and something similar could be added for the novice user, as previously suggested in [36].

Finally we would suggest two modifications to Torpark. The first is that it makes some effort to link to the Tor detector website—either by making it the default homepage, using a custom home page that contains information about Tor and includes a link to the Tor detector, or at the very least, putting it in the browser bookmarks. The second recommendation would be that Torpark indicates to the user that running Java applets could jeopardize her anonymity. This may be preferable to simply disabling all Java applets and scripts by default, given that this disablement will introduce new usability problems with many websites no longer functioning correctly. Explicitly articulating the security concern and explaining clearly and concisely how to disable Java would make Torpark better at preventing dangerous errors, although any solution to this problem would ideally be user-tested. This information could be given in the warning message or on a custom homepage like the one mentioned above.

With the exception of the anonymity meter, our recommendations are largely linguistic and would not require major revision to the programs themselves. Determining clear, meaningful explanations and instructions is difficult. A technique that may yield improved wording involves conducting a user study where a novice user is presented with an explanation and then asked to describe it to another novice user. Noting the use of terminology, metaphors and the level of detail across many subjects may yield

commonalities allowing an improved explanation.

In Chapter 4, we considered the case of 'anonymous' edits made to Wikipedia and postulated that the originators of these kinds of edits were unaware that the edits would be linked to their organizations. It follows from this hypothesis that users' mental model of the internet is inadequate. With respect to IP addresses, this is not surprising. Computer scientists and engineers have gone to great lengths to prevent users from ever having to deal with an IP address: domain name servers (DNS) map numerical IP addresses to more easily remembered linguistic domain names, while the Dynamic Host Configuration Protocol (DHCP) allows an IP address to be issued by a server to a computer in manner that is invisible to the user. Since IP addresses are protocol-based, its easy for users to not realize the privacy threat they pose. Hindrance to the wide adoption of Tor is likely supplemented by an unawareness of its benefits. Tor could also benefit by better framing its purpose and the privacy threats it is attempting to address on its website. This will help motivate users to deploy the software correctly and may help increase the accuracy of their mental models of the internet.

In conclusion, we have noted numerous usability problems that a novice user is likely to encounter in installing, configuring, and using Tor. It is our hope that the individual benefits of the examined software tools will be combined in order to help Tor achieve affinity among novice users. We also hope our guidelines will prove to be a useful compilation for future work in usable privacy.

Chapter 6

Concluding Remarks

6.1 A Review of the Problem

The motivating problem of this thesis originates from the tension between the current legal environment and the potential liability problems that surround these anonymity enhancing technologies. Many security tools have faced a similar legal dilemma: while empowering honest citizens, these tools can also be used in a manner that facilitates unlawful activities. Strong cryptographic protocols have been controversial, and were once subject to strict export regulations in countries like the United States. More recently in a 2006 report, the United States Federal Plan for Cyber Security and Information Assurance Research and Development[1] states concern over steganography noting that because 'steganography secretly embeds additional, and nearly undetectable, information content in digital products, the potential for covert dissemination of malicious software, mobile code, or information is great.'

On anonymity, the same report identifies *trackback* and *attribution* as useful functions for mitigating 'the shortcomings of the Internet's design by denying attackers anonymity and safe haven.' Trackback is defined as the ability to trace data to its point of origin, while attribution is the ability to link the data to an identity—whether it is a digital identity like an IP address or a physical one like a name. The report notes that trackback is difficult when data is routed through multinational servers in countries without strong diplomatic ties to United States law enforcement, and it notes the difficulties of tying a digital identity to a physical one. These considerations are equally relevant to Canadian

[1] `http://www.nitrd.gov/pubs/csia/csia_federal_plan.pdf`

law enforcement, as well as in other countries around the world.

We have argued in earlier chapters that anonymity networks will likely be designated a legal status similar to that of cryptography: an acknowledgment of their potential for unlawful behaviour that is ultimately outweighed by the privacy rights of honest citizens using the technology. Furthermore, we note that there are alternatives to anonymity networks for criminals who wish to remain anonymous. Computers can be attacked, compromised, and used as launching pads for unlawful online behaviour. Weakening anonymity networks is not likely to solve the root problem, only displace it at the cost of unconditional anonymity for honest citizens. For these pragmatic reasons, we propose that managing the problem is likely to be more effective than attempting to solve it.

6.2 The Structure of the Problem

The field of economics has important insight into the analysis of incentive structures, and how incentives can be changed to affect desired outcomes. The problem of anonymity is rooted in what economists call asymmetrical information. Anonymity is a service that is sought after by both lawful and unlawful persons. These persons know whether they are lawful or not, however the service provider has no obvious way of distinguishing between them. Providing anonymity to unlawful users has a cost to the service provider—potential liability or server confiscation—while providing the same service to lawful users does not have that cost. Therefore information that is relevant to the service being provided is asymmetrically distributed: the user knows whether they will create liability issues for the provider while the provider does not.

Three economists were awarded the Nobel Prize in 2001 for their analysis of markets with asymmetrical information. One of them, George Akerlof, identified two problems that tend to emerge [15]. The first is adverse selection. Services that would exclude certain consumers in a market of perfect information gives these consumers a comparative advantage in an asymmetric market where they hold the relevant information. As a result, such services have a tendency to attract the very consumers they would like to exclude, unless if they can find a way to restore the balance of information. The second problem identified by Akerlof is moral hazard. Moral hazard argues that the consumers of services may adjust their behaviour in a manner that is costly to the service. Applied to anonymity networks, adverse selection suggests that unlawful users have a greater incentive to use the network than lawful users and thus would be disproportionate in

the userbase. Moral hazard suggests that anonymous users may behave in ways that are marginally more malicious than nonanonymous users.

Methods to restore the symmetry of information are also of interest to economists. Game theorists define signaling as any successful method that allows consumers to communicate the relevant information about themselves to the service provider. For example, in previous chapters we have illustrated adverse selection and moral hazard in terms of medical insurance. A low-risk individual may be willing to voluntarily receive a clinical examination by a doctor or disclose their medical record (and many countries with private medical insurance providers have regulations requiring this disclosure). Legitimate banks used to build impressive buildings to signal to consumers that they were not a 'fly by night' operation. Michael Spense, another one of the three economists to win the 2001 Nobel prize, argued that a PhD is an effective job-market signal that increases an employer's chances of hiring a skillful and resourceful worker [67].

6.3 Summary of Original Contributions

We can begin addressing the problem of adverse selection, and its effects, by proposing changes that are engineered to mitigate risk and foster a positive incentive structure in anonymity networks. Honest users can choose to have a stake in their own anonymity by operating a server in the network. Some anonymity networks require this as a method for increasing available bandwidth, but the predominant anonymity network, Tor, does not. However the recent case of Tor servers being confiscated in Germany raises questions of the legal liability of operating such a server, and the ambiguity of the situation is a negative incentive for potential operators. Anonymity networks could better recruit volunteers if it provided a protocol that would allow an exit node to repudiate a message. In Chapter 3, we proposed two methods that allow a server to prove that a message originated from a digital identity other than its own. This resolves an important legal concern that results from adverse selection in anonymity networks.

It may be impossible for an anonymity network to discriminate between lawful and unlawful users before granting access to its service. However it is possible for an anonymity network to distinguish users who repeatedly perform unlawful actions if the users have a reputation. The concepts of reputation and anonymity appear to be mutually exclusive, however in Chapter 4 we provide a protocol that would allow a digital identity to be banned from submitting future messages to an anonymity network. A good reputation

of past behaviour is a signal that lawful users can use to communicate their lawfulness to the providers of an anonymity network, and this protocol can be used to directly combat adverse selection.

If unconditional anonymity is to be preserved, we need to make a compelling case that the privacy rights of lawful citizens outweigh the unfortunate facilitation of unlawful behaviour. This argument is premised on lawful citizens using anonymity networks. If Chapter 5, we examined one of the primary reasons lawful citizens do not use the anonymity network Tor: it is difficult to deploy. Given that unlawful users have an added incentive to overcome these difficulties, by virtue of their actions, that many lawful citizens do not (this is not without exception as we have noted), usability problems can exacerbate adverse selection. To combat this, we propose improvements that can be made to the deployability process and application interfaces to make Tor more attractive to novice users, and ultimately to make anonymity more accessible to ordinary citizens.

While combating adverse selection is the primary theme of this paper, the proposed solutions have secondary benefits. For example, increasing the number of volunteer operators in an anonymity network will increase the available bandwidth. The ability to ban unlawful behaviour could also be leveraged to prevent use of the anonymity network for high bandwidth purposes like downloading large media files. This is discouraged by many anonymity networks, including Tor, however there is an incentive for users to use an anonymity network to download media if its copyrighted and could provoke civil action from industry. In conjunction, these two improvements could see an increase in the global bandwidth of Tor. Its important to note that this does not necessarily make Tor faster for users: an increase in global throughput will likely attract new users who did not use the network prior to the increase because it was too slow, but the addition of new users decreases the share of throughput available to each individual user. This phenomenon is known as induced demand and should be relevant to bandwidth in an anonymity network, although it does not appear to have been empirically studied in the literature.

Finally, we note as a contingency that if the legality of anonymity networks should become limited only to networks that provide the ability to trace a message to the digital identity of its originator (contrary to our prediction), the protocol outlined in Chapter 4 provides a method for selectively tracing messages. The Access Control Server (ACS) would reveal the root nym of the user to law enforcement—who are running the Authentication Server (AS) and could link it to an IP address. To work, this would

require an incentive for the exit nodes to comply with the protocol (*i.e.,* only release messages that have a properly formed nym). The threat of legal action against non-compliant nodes nay work if the node is in an appropriate country but this does eliminate the multinational dimension of traceability (which is one of the main reasons we believe traceability is not legally tenable). It does however limit the jurisdictional requirements to a single server in a single country. This method is an improvement over the selective traceability in [13] because the integrity of the relationship between the message and IP address is the sole responsibility of law enforcement, and they do not need to rely on the trustworthiness of any other entity.

6.4 Future Work

The work in this thesis could be extended in a few important ways. While digital credentials are extremely versatile, they are computationally demanding. If more efficient cryptographic primitives could be combined in a way that provided all the needed properties of the solutions in Chapter 3 and 4, this would be a significant improvement. In particular, if the properties could be achieved using cryptographic primitives that is not patented, then our contributions could be deployed in free, open-source software like Tor without the complexities of licensing proprietary technology. With regard to the usability study in Chapter 5, further insight into the usability of Tor could be achieved by conducting a laboratory or field study of Tor with real participants performing the core tasks. Our research could also benefit from technologies that could increase the reliability of an IP address as a digital identifier. For example, since IP addresses are geographical, perhaps round-trip time could be used to falsify the validity of a claimed IP address. Our research could also benefit from an investigation into the possibility of some sort of authentication protocol between a customer and their internet service provider that increases the integrity of an IP address.

6.5 Final Remarks

We believe our research has made a significant contribution to resolving adverse selection, and its effects, in anonymity networks. We do not claim to have completely solved this problem, and in fact, we have expressed skepticism over the proposition that it can be

completely solved. However we do claim to have made notable progress in managing the problem. We have contributed three novel protocols, two for exit node reputability and one for provable revocable access, and have performed the first usability study of the deployability of any anonymity network. We also hope that our multidisciplinary approach to the subject has provoked insights that a purely technical explanation could not capture.

Appendix A

Glossary of Terms

Anonymity: Actions are not linkable to the actor's identity or to other actions by the same actor; namelessness.

Adverse Selection: Services have a tendency to be allocated to the most costly customers. For example, those with life-threatening jobs tend to purchase life insurance.

Blind Signature: A method allowing Bob to sign a message for Alice without being able to read it.

Cognitive Walkthrough: A method for evaluating the usability of an application. Using a set of guidelines, the evaluator performs core tasks while checking for potential usability issues.

Confirmation Bias: A cognitive bias that encourages new information to be interpreted in a way that confirms one's existing hypotheses. It also refers to a tendency to avoid information that may falsify one's beliefs.

Digital Credential: A method allowing Alice to have personal information encoded by Bob into a signed document. Alice can blind the digital credential before using it so Bob cannot recognize it and trace it to her. The information inside the credential can be revealed in part, in full, or in other flexible ways. For example, a digital credential version of a driver's license could permit Alice to prove she is over 19 without revealing her actual age, or that her name is not that of a subject of interest without revealing what her name is.

Exit Node Repudiation: A method allowing an exit node in an anonymity network like Tor to disclaim authorship of a message it forwards on behalf of an anonymous sender.

Group Signature: A method allowing Alice to anonymously sign a document on behalf of a group. Only the group manager can link a signature to the group member who originated the signature.

Internet Protocol (IP) Address: A (semi-)unique numerical identifier for participants in intranet and internet communication. At the time of writing, 72.14.207.99 is the IP address of the Google.com web server.

Mix Network: A collection of specialized nodes that forward messages in batches after decrypting and reordering them. A mix network provides anonymity by obfuscating the relationship between a message and its sender.

Moral Hazard: Reduction of risk to an individual causes the individual to engage in more risky behaviour. For example, drivers with car insurance may take fewer precautions when driving than those without.

Nonce: A random value that is used one time only in a cryptographic protocol to make the transaction unique, and to prevent one-side of a past transaction from being replayed.

Pseudonymity: Actions are not linkable to the actor's identity but are linkable to other actions by the same actor; using an alternate name.

Tor: A popular anonymity network for hiding one's IP address while web browsing. The Tor network is loosely based on mix networks.

Veronymity: Actions are linkable to the actor's identity; using a true name.

Bibliography

[1] Anonymity bibliography. Hosted by *Freehaven*.
`http://freehaven.net/anonbib/` (accessed Oov 2007)

[2] Foxyproxy 2.2.1
`http://foxyproxy.mozdev.org` (accessed Nov 2006)

[3] *Harms v. Miami Daily News Inc.* 127 So. 2d 715. Fla. Dist. Ct. App., 1961.

[4] *Personal Information Protection and Electronic Documents Act* (Canada). Chapter P-8.6, S.C. 2000, c.5.

[5] *Privacy Act* (Canada). R.S.C. 1985, c. P-21.

[6] Privoxy 3.0.3
`http://www.privoxy.org` (accessed Nov 2006)

[7] Tor 0.1.1.25
`http://tor.eff.org` (accessed Nov 2006)

[8] Torbutton 1.0.4
`http://freehaven.net/~squires/torbutton/` (accessed Nov 2006)

[9] Torpark 1.5.0.7a
`http://www.torrify.com` (accessed Nov 2006)

[10] Vidalia 0.0.7
`http://vidalia-project.net` (accessed Nov 2006)

[11] A. Acquisti, R. Dingledine, and P. Syverson. On the Economics of Anonymity. *Proceedings of Financial Cryptography (FC '03)*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 2742, 2003, pages 84–102.

[12] C. Adams. A classification for privacy techniques. *University of Ottawa Law & Technology Journal: special issue on anonymity, privacy, and identity*, Volume 3, Issue 1, 2006, pages 35–52.

[13] L. von Ahn, A. Bortz, N. J. Hopper, and K. ONeill. Selectively traceable anonymity. *DTC Research Report 2006/14*, Digital Technology Center, University of Minnesota, June 2006.

[14] L. von Ahn, A. Bortz, N. J. Hopper, and K. ONeill. Selectively traceable anonymity. *Sixth Workshop on Privacy Enchancing Technologies: Proceedings of PET 2006*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 4258, 2006, pages 208–222.

[15] G. A. Akerlof. The market for "lemons": quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*. Volume 84, Issue 3, August 1970, pages 488–500.

[16] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: a system for anonymous and unobservable internet access. *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability (2000)*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 2009, 2001, pages 115–129.

[17] M. Bishop. Psychological acceptability revisited. In "Security and Usability," `O'Reilly`, 2005, pages 1–12.

[18] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. *Proceedings of 2004 ACM Workshop On Privacy In The Electronic Society (WPIES 2004)*, ACM Press, 2004, pages 77–84.

[19] S. Brands. "Rethinking public key infrastructures and digital certificates: building in privacy." *MIT Press*, 2000.

[20] S. Brands. A technical overview of digital credentials. Available online. February 20, 2002.
`http://www.sics.se/~lra/thesis/library/brands02technical.pdf`

[21] R. T. Carback, J. Clark, A. Essex, and S. Popoveniuc. On the independent verification of a Punchscan election. *Proceedings of University Voting Systems Competition (VoComp 2007)*, 2007.

[22] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. *Proceedings of the 15th USENIX Security Symposium*, 2006, pages 1–16.

[23] D. Chaum. Blind signatures for untraceable payments. *Advances in Cryptology: Proceedings of Crypto 82*, 1983, pages 199–203.

[24] D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *Journal of Cryptology*. Springer, Volume 1, Issue 1, January 1988, pages 65–75.

[25] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, Volume 24, Issue 2, 1981, pages 84–88.

[26] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ACM Press, 1988, pages 11–19.

[27] D. Chaum and E. van Heyst. Group signatures. *Advances in Cryptology EUROCRYPT 91*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 547, 1991, 257–265.

[28] D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. *Technical Report CS-TR-880*, University of Newcastle upon Tyne, 2004.

[29] J. Clark, A. Essex, and C. Adams. On the security of ballot receipts in E2E voting systems. *Proceedings of the Workshop on Trustworthy Elections (WOTE 2007)*, 2007.

[30] J. Clark, A. Essex, and C. Adams. Secure and observable auditing of electronic voting systems using stock indices. *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2007)*, 2007.

[31] J. Clark, P. Gauvin, and C. Adams. On controlling IP address dissemination using digital credentials within mix networks. *Proceedings of On the Identity Trail International Workshop on Anonymity*, 2007.

[32] J. Clark, P. Gauvin, and C. Adams. Exit node repudiation for anonymity networks. Forthcoming book chapter in an *On the Identity Trail* publication, 2008.

[33] J. Clark, P. van Oorschot, and C. Adams. Usability of anonymous web browsing: an examination of Tor interfaces and deployability. *Proceedings of the Third Symposium On Usable Privacy and Security (SOUPS 2007)*. ACM Press, *ACM International Conference Proceeding Series*, Volume 229, 2007, pages 41–51.

[34] L. Cranor. Privacy policies and privacy preferences. In "Security and Usability," `O'Reilly`, 2005, pages 447–472.

[35] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: design of a type III anonymous remailer protocol. *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003, pages 2–15.

[36] R. Dingledine and N. Mathewson. Anonymity loves company: usability and the network effect. *Pre-Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, 2006, pages 497–508.

[37] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004, pages 303-320.

[38] R. Dingledine, V. Shmatikov, and P. Syverson. Synchronous batching: from cascades to free routes. *Fourth Workshop on Privacy Enchancing Technologies: Proceedings of PET 2004*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 3424, 2005, pages 186–206.

[39] B. Edelman. Adverse Selection in Online 'Trust' Certifications. *Pre-Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, 2006, pages 529–552.

[40] A. Essex, J. Clark, R. T. Carback, and S. Popoveniuc. Punchscan in practice: an E2E election case study. *Proceedings of Workshop on Trustworthy Elections (WOTE 2007)*, 2007.

[41] I. Goldberg. On the security of the Tor authentication protocol. *Sixth Workshop on Privacy Enhancing Technologies: Proceedings of PET 2006*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 4258, 2006, pages 316–331.

[42] D. Goldschlag, M. Reed, and P. Syverson. Hiding Routing Information. *Proceedings of the First International Workshop on Information Hiding*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 1174, 1996, pages 137–150.

[43] P. Golle. Reputable mix networks. *Fourth Workshop on Privacy Enchancing Technologies: Proceedings of PET 2004*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 3424, 2005, pages 51–62.

[44] P. Golle, M. Jakobsson, A. Juels and P. Syverson. Universal re-encryption for mix networks. *Proceedings of the 2004 RSA Conference*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 2964, 2004, pages 163–178.

[45] N. Good and A. Krekelberg. Usability and privacy: A study of KaZaA P2P file sharing. In "Security and Usability," *O'Reilly*, 2005, pages 651–667.

[46] A. Hintz. Fingerprinting websites using traffic analysis. *Third Workshop on Privacy Enhancing Technologies: Proceedings of PET 2003*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 2482, 2003, pages 229–233.

[47] M. Jakobsson, A. Juels and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. *Proceedings of the 11th USENIX Security Symposium*, 2002, pages 339–353.

[48] P. C. Johnson, A. Kapadia, P. P. Tsang, and S.W. Smith. Nymble: anonymous ip-address blocking. *Seventh Workshop of Privacy Enhancing Technologies: Proceedings of PET 2007* (to appear), 2007.

[49] C. Karat, C. Brodie, and J. Karat. Usability design and evaluation for privacy and security solutions. In "Security and Usability," *O'Reilly*, 2005, pages 47–74.

[50] I. Kerr. Anonymity. Encyclopedia article (in progress). `idtrail.org/files/Anonymity%20Entry%20_Kerr.pdf`

[51] K. Kirby. Bidding on the future: Evidence against normative discounting of delayed rewards. In *Quarterly Journal of Experimental Psychology*, Volume 126, 1997, pages 54–70.

[52] D. Koblas and M. R. Koblas. Socks. *Proceedings of the 11th USENIX Security Symposium*, 1992, pages 77–83.

[53] K. McLeod. "Freedom of expression: overzealous copyright bozos and other enemies of creativity." *Random House*, 2005.

[54] A. Menezes, P. van Oorschot, S. Vanstone. "Handbook of applied cryptography." *CRC Press*, 1997.

[55] J. Muir and P. van Oorschot. Internet geolocation and evasion. *Technical Report TR-06-05*, School of Computer Science, Carleton University, April 2006.

[56] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, 2005, pages 183–195.

[57] J. Nielson. Heuristic evaluation. In "Usability inspection methods," *Wiley & Sons*, 1994, pages 25–62.

[58] J. Oates. German police seize Tor servers. *The Register*, September 11, 2006. `http://www.theregister.co.uk/2006/09/11/anon_servers_seized/`

[59] W. Ogata, K. Kurosawa, K. Sako and K. Takatani. Fault tolerant anonymous channel. *Proceedings of the First International Conference on Information and Communication Security*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 1334, 1997, pages 440-444.

[60] V. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. *Proceedings of SIGCOMM 2001*. ACM Press, *ACM SIGCOMM Computer Communication Review*, Volume 31, Issue 4, 2001, pages 173–185.

[61] S. Peltzman. The Effects of Automobile Safety Regulation. *The Journal of Political Economy*, Volume 83, Issue 4, 1975, pages 677–726.

[62] J. Raymond. Traffic analysis: protocols, attacks, design issues, and open problems. *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability (2000)*. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 2009, 2001, pages 10-29.

[63] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, Volume 16, Issue 4, 1998, pages 482-494.

[64] M. Rennhard and B. Plattner. Introducing MorphMix: peer-to-Peer based anonymous internet usage with collusion detection. *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, 2002, pages 91–102.

[65] D. J. Solove. A Taxonomy of Privacy. *University of Pennsylvania Law Review*, Volume 154, Number 3, January 2006, pages 477–560.

[66] D. J. Solove. 'I've got nothing to hide' and other misunderstandings of privacy. *San Diego Law Review*, Volume 44, 2007.

[67] M. Spense. Job Market Signaling. *Quarterly Journal of Economics*, Volume 87, Issue 3, 1973, pages 355–374.

[68] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, 1997, page 44.

[69] P. Wason. On the failure to eliminate hypotheses in a conceptual task. *Quarterly Journal of Experimental Psychology*, Volume 12, 1960, pages 129–140.

[70] C. Wharton, J. Rieman, C. Lewis, and P. Polson. The cognitive walkthrough method: a practitioner's guide. In "Usability Inspection," *Wiley & Sons*, 1994, pages 84–89.

[71] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. *Proceedings of the 8th conference on USENIX Security Symposium*, 1999, page 14.