# Valet Services: Improving Hidden Servers with a Personal Touch

Lasse Øverlier[1,2] and Paul Syverson[3]

[1] Norwegian Defence Research Establishment, P.B. 25, 2027 Kjeller, Norway
`lasse.overlier@ffi.no`, `http://www.ffi.no/`
[2] Gjøvik University College, P.B. 191, 2802 Gjøvik, Norway
`lasse@hig.no`, `http://www.hig.no/`
[3] Center for High Assurance Computer Systems
Naval Research Laboratory Code 5540, Washington, DC 20375
`syverson@itd.nrl.navy.mil`, `http://chacs.nrl.navy.mil/`

**Abstract.** Location hidden services have received increasing attention as a means to resist censorship and protect the identity of service operators. Research and vulnerability analysis to date has mainly focused on how to locate the hidden service. But while the hiding techniques have improved, almost no progress has been made in increasing the resistance against DoS attacks directly or indirectly on hidden services. In this paper we suggest improvements that should be easy to adopt within the existing hidden service design, improvements that will both reduce vulnerability to DoS attacks and add QoS as a service option. In addition we show how to hide not just the location but the existence of the hidden service from everyone but the users knowing its service address. Not even the public directory servers will know how a private hidden service can be contacted, or know it exists.

## 1 Introduction

Hidden Servers are a means to offering attack-resistant services. A server that is accessible but hidden can resist a variety of threats simply because it cannot be found. These threats include physical and logical threats to the service itself. But they also include threats to the people offering the service and attempts to prevent general access to the service provided.

Since 2004, hidden services have been offered that use Tor to underly services offered from hidden locations. These were introduced [11] as resistant to distributed DoS since they were designed to require a DDoS attack on the entire Tor network in order to attack a hidden server.

Recent events have placed Tor prominently in the international media as a tool to allow people to access Internet sites even if they are behind filtering firewalls or if the large commercial search engines are cooperating with local authorities to provide only censored offerings. However, at least as important as obtaining information is the ability for people in these environments to disseminate information. Besides resisting DDoS and physical threats, hidden servers have also been recommended for preserving the anonymity of the service offerer and to resist censorship. Specifically, Undergroundmedia.org has published a guide to "Torcasting" (anonymity-preserving and censorship-resistant podcasting). And both the Electronic Frontier Foundation and Reporters Without Borders have issued guides that describe using hidden services via Tor to protect the safety of dissidents as well as to resist censorship. Even in more open societies bloggers have lost their jobs because employers were unhappy about the blog sites.

Hidden services thus have a clear value and appeal. But, their resistance to some adversaries is limited. In [17], we demonstrated location attacks on hidden servers deployed behind Tor that locate a hidden server quickly and easily, often within minutes. The suggested countermeasures to those attacks have been implemented. But other threats remain. Hidden servers are accessed via a publicly listed small set of relatively long-lived *Introduction Points.* Anyone with access to a hidden service can easily discover the Introduction Points. This can lead to a DoS race that the hidden server is likely to lose, since setting up Introduction Points and disseminating associated information is somewhat resource intensive. To address this limitation in the current hidden service design we propose the introduction of *Valet nodes.* There can be far more valet nodes than introduction points associated with a hidden server. Relatedly, it is much easier to generate and disseminate valet node information than introduction point information.

In Sect. 2 we present previous work on hidden services and availability together with a brief look into how Tor's hidden services work. In Sect. 3 we give a description of the valet node design. In Sect. 4 we discuss the security of the design. In Sect. 5 we present our conclusions.

## 2 Previous Work on Availability and Hidden Services

Location hidden services build upon anonymous communication, which was first described by David Chaum [7] in 1981. Most of the early work in this area focused on high-latency communications, like email. Low-latency anonymous communication, such as currently dominates Internet traffic, got new focus in the late 1990's with the the introduction of onion routing [14]. In 1995, shortly before onion routing was initially deployed, the first low-latency commercial proxy for web traffic, the Anonymizer [2] became available. Proxy services like Anonymizer and Proxify [19] work by mixing traffic from multiple clients through a single point so that any accessed servers are only able to trace back clients to the anonymizing proxy, and not to the actual users. This requires complete trust in the proxy provider and will unfortunately be easy to abuse since we now have a single point of failure, a single point of compromise and *a single point of attack.*

Distributed low-latency anonymous communication systems include the original onion routing [14], the Freedom Network [6] (deployed in 1999), and the current version of onion routing, Tor [11]. These are more resistant to the above mentioned vulnerabilities because they proxy communication through multiple hops; at each hop the communication changes its appearance by adding or removing a layer of encryption (depending on whether it is traveling from the circuit originator to responder or vice versa). They all use public-key cryptography to distribute session keys to the nodes along a route, thus establishing a circuit. Each session key is shared between the circuit initiator (client) and the one node that was given that key in establishing the circuit. Data that passes along the circuit uses these session keys. Both Freedom and Tor have a default circuit length of three nodes. For more details consult the above cited work. Another low-latency distributed system is JAP/Web MIXes [4]. It is based on mix cascades (all traffic shares the same fixed path) and thus, unlike the above systems, its security is not based on hiding the points at which traffic enters and leaves the network. It is thus not directly usable for hidden services as they are described below.

The property of hiding the location of a service in order to sustain availability was introduced in Ross Anderson's Eternity Service [1]. Focusing on availability and longevity of data, the Eternity service stores files at multiple locations, encrypted and prepaid for, during a certain period of time. Freenet[8] was the first system to use a peer-to-peer network with the goal of censorship resistance enabling a service to have (some) availability even when only one of the nodes is available. Splitting the stored files up into minor pieces and storing them on multiple nodes of the network also added robustness. However it has numerous security vulnerabilities, e.g., clients must trust the first nodes they connect to for all network discovery and hence anonymity protection. Both Freenet and GNUnet[3] communication builds upon mix-net[7] technology for sending messages to other nodes, and must trust the availability of the underlying network. Publius [15] was designed to guarantee persistence of stored files, like Eternity and unlike Freenet. Tangler [27] additionally makes newly published files dependent on previous ones, called *entanglement*, thereby distributing incentives for storing other nodes' files as well. Free Haven [10] uses a network of nodes with a reputation system involving contracts between nodes to store data for others. But Free Haven does not define the underlying anonymous communication channel, and this is where many of the availability issues are located.

Tor is not a publishing service and does not store information like the above mentioned censorship-resistant systems, but Tor facilitates something called *hidden services.* The hidden service design supports the anonymous access of common services, e.g. a web service, enabling users of the network to connect to these services without knowing the server's location (IP address). Tor builds this functionality upon the assumption that if a node cannot be located, it cannot be (easily) stopped or in other means shut down. The hidden service design relies on a rendezvous server, which mates anonymous circuits from two principals so that each relies only on himself to build a secure circuit. The first published design for a rendezvous service was for anonymous ISDN telephony [18] rather than Internet communication. As such it had very different assumptions and requirements from the rendezvous servers we describe, some of which we have already noted above. A rendezvous server for IRC chat was mentioned in [14]; however, the first detailed design for a rendezvous server for Internet communication was by Goldberg [13]. It differs in many ways from rendezvous servers as used by Tor's hidden services, and we will not discuss Goldberg's design further here.

## 2.1 Location-hidden Services in Tor

In the current implementation of Tor, a connection to a hidden service involves five important nodes in addition to the nodes used for basic anonymous communication over Tor.

– HS, the Hidden Server offering some kind of (hidden) service to the users of the Tor network, e.g. web pages, mail accounts, login service, etc.
– C, the client connecting to the Hidden Server.
– DS, a Directory Server containing information about the Tor network nodes and used as the point of contact for information on where to contact hidden services.
– RP, the Rendezvous Point is the only node in the data tunnel that is known to both sides.
– IPo, the Introduction Point where the Hidden Server is listening for connections to the hidden service.

A normal setup of communication between a client and a hidden service is done as shown in Fig. 1. All the displayed connections are anonymized, i.e., they are routed through several anonymizing nodes on their path towards the other end. Every arrow and connection in the figure represents an anonymous channel consisting of at least two or more intermediate nodes. (Hereafter, we use 'node' to refer exclusively to nodes of the underlying anonymization network, sometimes also called 'server nodes'. Although we are considering the Tor network specifically, the setup would apply as well if some other anonymizing network were used to underly the hidden service protocol. Unlike the other principals above, C and HS may be anonymization nodes or they may be merely clients external to the anonymization network.)

First the Hidden Server connects (1) to a node in the Tor network and asks if it is OK for the node to act as an Introduction Point for his service. If the node accepts, we keep the circuit open and continue; otherwise HS tries another node until successful. These connections are kept open forever, i.e., until one of the nodes restarts or decides to pull it down.[4] Next, the Hidden Server contacts (2) the Directory Server and asks it to publish the contact information of its hidden service. The hidden service is now ready to receive connection requests from clients.

In order to communicate with the service the Client connects (3) to DS and asks for the contact information of the identified service and retrieves it if it exists (including the addresses of Introduction Points). There can be multiple

[4] In Tor any node in a circuit can initiate a circuit teardown.

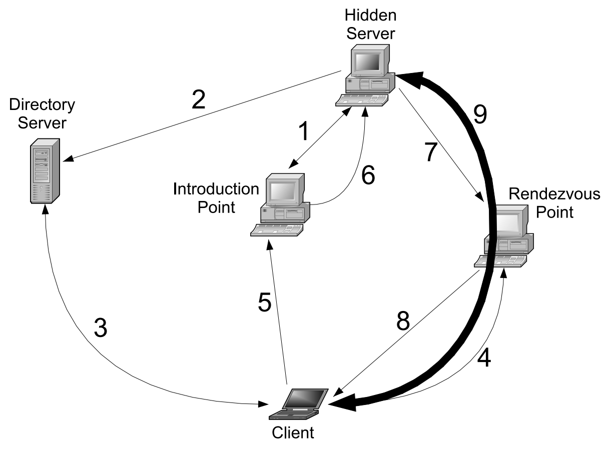

**Fig. 1.** Normal use of hidden services and rendezvous servers

Introduction Points per service. The Client then selects a node in the network to act as a Rendezvous Point, connects (4) to it and asks it to listen for connections from a hidden service on C's behalf. The Client repeats this until a Rendezvous Point has accepted, and then contacts (5) the Introduction Point and asks it to forward the information about the selected RP.[5] The Introduction Point forwards (6) this message to the Hidden Server, which determines whether to connect to the Rendezvous Point or not. If OK, the Hidden Server connects (7) to RP and asks to be connected to the waiting rendezvous circuit, and RP then forwards (8) this connection request to the Client.

Now RP can start passing data between the two connections and the result is an anonymous data tunnel (9) from C to HS through RP.

## 2.2 Threats to Hidden Services

Until now most papers on anonymizing networks have focused on the threats of locating users and services in the network, and addressed different threat scenarios like intersection attacks [22, 29] and traffic analysis [21, 24].

A large adversary will be able to correlate network traffic going into and out from a distributed anonymizing network. For hidden services inside such a network this will also be true if large (or critical[6]) portions of the network can be observed at any given time. Using techniques from Serjantov and Sewell [24] even a smaller adversary can match timing to and from nodes in the network. If a suspected communication channel is to be verified, e.g. "A is talking to B", this will quite easily be confirmed by simple statistical methods. Murdoch and

[5] Optionally, this could also include authentication information for the service to determine from whom to accept connections.

[6] E.g. smaller bounded parts of the network including the communicating nodes.

Danezis [16] controlled a service accessed by clients through the Tor network and thereby were able to trace the route of traffic through the Tor network (but not all the way to the client). This attack involved probing the entire network. It was achieved on an earlier and much smaller Tor network than now exists.

In [17], we demonstrated on a live anonymizing network how effective intersection attacks can be in locating hidden servers and clients. The paper also describes other vulnerabilities in the hidden services design that makes it simple for an attacker to locate the Hidden Server.

All these attacks address how to *locate* either the user or the hidden service. But there are other threats that are important, like preventing Denial-of-Service attacks. This is of major concern when we are trying to achieve availability for the service, because even if the adversary cannot locate the server, the next best thing will be to shut down the possible access methods (or channels) for contacting it.

In [25] Stavrou and Keromytis describes how to use an indirection-based overlay network (ION) for DDoS protection by using packet replication and packet path diversity combined with redirection-based authentication. This is currently not applicable to the current implementation of the Tor hidden service protocol as Tor is based on TCP communication and the described ION requires a stateless protocol.

In the current Tor design, hidden services publish their contact information on a directory server describing to any user how she can connect to them. This information contains, amongst other things, a signed public key and a list of introduction points to contact in order to get a connection to the hidden service (cf. Sect. 2.1). This list can be abused by an attacker targeting all the introduction points with a DoS attack and thereby disabling the hidden service.

In addition, if a node is chosen to be an introduction point for a hidden service, it will be able to easily discover this through the general availability of the contact information retrieved from the directory services. This availability makes it possible for an adversary to shut down a service by attacking its introduction points, and also makes it possible to stop some selected services, e.g. by threatening all introduction points to avoid being associated with a particular service descriptor.

The directory server will also be able to see all hidden services that are published and can enumerate them, identify when they first became available, identify all their introduction points, and contact all the services (that do not require additional authorization).

If the directory servers are compromised, or if all of them are subject to a DoS attack, this could effectively shut down the entire network. For more information on this consult [11]. We do not address this possibility here.

## 3 Valet Service

Valet service adds another layer of protection to the hidden service design. By re-enabling some parts of the reply-onion technology from the original onion routing design [14], we will hide the Introduction Points from the users, and we will also extend the functionality of the hidden service.

### 3.1 Overview

We introduce a method of accessing the Introduction Points without knowing their location. This is quite similar to the situation of hiding the hidden service in the first place, but now we want to hide more nodes (all Introduction Points), and we want to make only a few of the contact points visible to any user or node in the network at a specific time.

To accomplish this we introduce *Valet Service nodes*, or simply *Valet nodes*. These nodes are the new contact points for the Client, they can be different for different clients or groups of clients, and will enable the service to maintain a limited number of Introduction Points, but multiple contact points. So neither the public nor the real users know about the identity of several of these nodes at a time. We also avoid having the Valet nodes knowing which services they are assisting, and we make sure that the Valet nodes do not know more than one Introduction Point per connection request. The information about these Valet Service nodes are found in Contact Information tickets which will be discussed later.

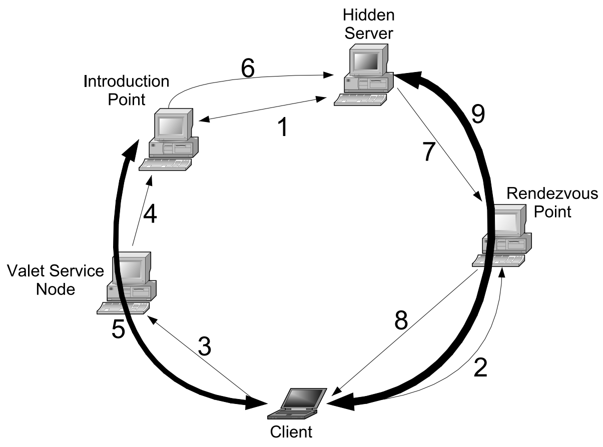

**Fig. 2.** Use of Valet Service

In Fig. 2 we illustrate that the Hidden Server tunnels (1) out to at least one Introduction Point, and creates a listener for contact requests. The Client tunnels (2) out to a Rendezvous Point as in the original setup, and constructs the rendezvous information (including, e.g., authentication information) that it will send to the Hidden Server.

Using information about the Valet Service nodes in the Contact Information ticket, the Client tunnels (3) out to the Valet node. The Valet node receives a

Valet token encrypted with its public key, and containing information about the Introduction Point. The Valet node then extends (4) the circuit to the Introduction Point so that the Client now can communicate directly (5) and securely with the Introduction Point without knowing who this is, just that it is communicating with a node authorized by the Hidden Server.

The Client then uses information from the Contact Information ticket to authenticate the connection through the Introduction Point (IPo) and delivers to it the client's message to be forwarded (6) to the Hidden Server.

Based on the received information, the hidden service now determines whether to contact (7) the Rendezvous Point and complete the anonymous tunnel (8&9) setup with the Client, or to drop the request.

There are several challenges in this extension of the protocol, e.g., lost and expired tickets and the selection of contact information, but we will address these after a more detailed description of how valet services work.

### 3.2 Description

We divide the connection phase of contacting the hidden service into five parts:

1. The Hidden Server's setup of the Introduction Points and the construction and distribution of the Contact Information tickets.
2. The Client setting up a Rendezvous Point and contacting the Valet node.
3. The Valet node extending the circuit to the Introduction Point.
4. The Client authenticating and sending contact and authentication information to the Introduction Point, which forwards this to the Hidden Server.
5. The Hidden Server contacting the Rendezvous Point and finalizing the connection with the Client.

We will address the construction and distribution of the Contact Information tickets at the end so the reader will learn how the information is used and hence better understand why it contains what it does.

**Client contacting Valet Service node** Following Fig. 2, we assume that the Hidden Server has set up (1) a set of Introduction Points to be used by the clients and distributed information about how to connect to them. (See discussion of "Distributing tickets" below in this section.)

Contact Information tickets are shown in Table 1. Each contains a signed list of Valet nodes that are allowed to contact the Introduction Points and optional authentication information.

Before the contact to the Valet node can be completed the Client must first have selected a Rendezvous Point and connected to it (2) through an anonymizing tunnel. The Client will instruct RP to listen for authenticated connections and pass this RP contact information along to the Hidden Server.

Now the Client selects one of the Valet nodes listed in the Contact Information ticket and constructs a tunnel (3) to it, similar to the tunnel it created to the Introduction Point in the original version.

| | |
|---|---|
| $ValetServiceNode_1$ | One Valet node to forward the Client's information to an Introduction Point |
| $TimestampVS_1$ | Expiry time of this Valet Token |
| $ValetToken_1$ | Identifier for this Valet's connection to one or more Introduction Points, encrypted with the Valet node's public key |
| $PublicServiceKeyIPo_1$ | Provides the client with the public service key of $IPo_1$ |
| $ValetServiceNode_{2\&3...}$ | Other Valet node(s) |
| $TicketID$ | Ticket identifier for client to show to IPo and HS |
| $TimeStampCI$ | Validity period for information in Contact Information ticket |
| $AuthorizeCtoHS$ | Optional authorization information for ticket (or for C) to connect to HS |

**Table 1.** Contact Information ticket

**Valet node contacting Introduction Point** At this point the Client needs to extend the circuit to the Introduction Point. The extra functionality of adding the Valet node requires that we must have a method of assuring that we are talking to the Introduction Point without knowing its location or public key. This can be solved by having the Introduction Point associate a special *service key* with each associated hidden service contact point. This IPo service key pair and a corresponding key identifier, is generated by the Hidden Server, and the private part of the key pair together with the key identifier is transferred to the Introduction Point upon setup of the listener. In other words, when acting as an Introduction Point, the node will use the given private key for authenticating in an extension of the tunnel to it, and not its usual private node key. In addition the Introduction Point is told which Valet nodes are allowed to use this key when connecting. The public part of the service key is sent to the client through the Contact Information ticket shown in Table 1. The Introduction Point is not given the public service key thus making it harder to find out the hidden service with which it is associated.

| | |
|---|---|
| $IPo_1$ | Identity of Introduction Point |
| $TimeStampIPo_1$ | Valet Token's validity period |
| $\{ValetIdentifierIPo_1\}_{sign,enc}$ | Identifies the connection at Introduction Point $IPo_1$ |
| $\rightarrow privateIPo_1ServiceKeyID$ | The key for $IPo_1$ to use for extension of the circuit |
| $\rightarrow ValetServiceNodeID_1$ | Identifies the valet node allowed to extend to $IPo_1$ |
| $\rightarrow TimeStampIPo_1$ | Valet Identifier's validity period |

**Table 2.** Content of Valet Token including the Valet Identifier

The Client must send information to the Valet Service node that the Client itself is unable to read. The Client finds this information in the Contact Information ticket as the *Valet Token*, see Table 2, and it is encrypted (by the Hidden Service) using the public key of the Valet Service node. The token contains the identity of the Introduction Point, a time stamp, and a *Valet Identifier*. The Valet Identifier is used for identifying to the Introduction Point what service key to use and for confirming which Valet Service node is allowed to extend to it using this key. The Valet Identifier is signed with the private service key (IPo verifies the signature by producing the signature itself) and encrypted with the public node key of the Introduction Point.

But the Client also needs to give information to the Introduction Point upon extension of a tunnel similar to the way a usual tunnel extension is done. The message, an *Extend Tunnel Message*, contains normal circuit extension parameters, including a DH start $g^x$, added replay protection and identification of the Valet node. The Extend Tunnel Message is encrypted with the public part of the service key to ensure that only the Introduction Point can receive the message.

So the Client creates an *Extend Tunnel Message* and submits (3) this together with the *Valet Token* to the Valet node asking it to extend the tunnel (circuit) to the Introduction Point inside the Valet Token. The Valet node extracts the identity of the Introduction Point from the Valet Token and extends (4) the circuit to this node by forwarding the Extend Tunnel Message. The Introduction Point checks the message for the correct Valet node, correct key ID and signatures, and replies to the Client to complete the DH exchange using $g^y$ and a verification of correct key, as in current design. The Client now has a secure communication channel (5) to the Introduction Point without knowing its real identity. And the Introduction Point knows which of the hidden service descriptors the channel is associated with, and as before it knows nothing about the Client's identity.

**Client contacting Hidden Service** Now the Client has to send the Rendezvous Point's contact information to the Hidden Service via the Introduction Point. The Client sends (5) the $TicketID$ found in the Contact Information ticket to the Introduction Point to identify the use of the ticket, and it also attaches a time stamp and the information going to the Hidden Service (encrypted with the Hidden Service's public key). The Introduction Point checks the $TicketID$ and $TimeStampIPo$ and then forwards (6) the Hidden Server message (see Table 3) containing contact information of the Rendezvous Point together with the first part of a DH-key exchange and optional authentication information. If we wanted to identify the Valet node used for contacting the hidden service we could do this in the authentication field. The message is protected from interception by the Valet node via the DH-key the Client exchanged with IPo.

**Hidden Server contacting Rendezvous Point** After authorizing the connection from the Client, the Hidden Server connects (7) to the Rendezvous Point to finalize the connection of the anonymous tunnel.

The Rendezvous Point authenticates the request based on the $RendezvousPoint$ information the Hidden Server received from the Client (Table 3), and then forwards (8) the finalization of the Client to Hidden Server DH key exchange with the (optional) new Contact Information ticket to the Client. Then the Ren-

| | |
|---|---|
| $TicketID$ | Information for IPo to verify ticket access before forwarding message to HS |
| $TimeStampIPo$ | to verify validity period |
| The following is encrypted with HS's public key | |
| $RendezvousPoint$ | Contact and authentication information for HS to contact the RP |
| $g^x$ | First part of Client's ephemeral DH-exchange with Hidden Server |
| $TimeStampHS$ | period of validity of this request |
| $TicketID$ | Information to identify the ticket to HS |
| $AuthorizeCtoHS$ | (optional) authentication information for ticket or for C towards HS |

**Table 3.** Message from Client to Introduction Point

dezvous Point connects the two tunnels forming (9) the authenticated, secure and anonymous channel between the Client and the Hidden Server.

**Constructing Contact Information tickets** Uptime history and bandwidth availability are the most important factors when the hidden service is constructing contact information tickets for its clients. [7] The hidden service first has to select a set of nodes with high uptime to use as Introduction Points and as Valet nodes. For each of the Introduction Points the hidden service constructs a service-access public-key pair and submits one part, the "private"[8] key to the Introduction Point, and the other "public" key is to be put into the Contact Information ticket as shown in Table. 1. The key pair is given an identifier, $privateIPo_1ServiceKeyID$, to help identify the use of the correct key upon connection requests. In addition the hidden service constructs $TicketID$s allowing different clients (i.e. different tickets) to connect to the same Introduction Point. These $TicketID$s are sent to the Introduction Point together with the private key and the key identifier upon setup of the Introduction Point's listener.

So after selecting a Valet node, the hidden service is now ready to construct the Contact Information ticket. First it packs the service key ID together with a validity period and the identity of the Valet node into the *Valet Identifier*. The private *service* key is then used to sign the Valet Identifier before it is encrypted with the public *node* key of the Introduction Point (cf. Table 2).

The Valet Identifier is then put into the *Valet Token* together with a time stamp and the identity (the public node key) of the Introduction Point. The Valet Token is encrypted with the public key of the Valet node and put into the Contact Information ticket together with its public node key and the public service key of the associated Introduction Point.

The Contact Information ticket is now built up of identifiers of Valet nodes, a Valet node validity period, and the corresponding Valet Token with the asso-

[7] In Tor there currently exists no certification of this information as this would require active measurements of a nodes' capacity and availability during operation.

[8] The key is generated by HS not IPo, but used as a private key by HS (signing) and IPo (decrypting).

ciated public service key of the Introduction Point. There can be as many Valet nodes listed in a ticket as the hidden service finds appropriate. The ticket will also contain a $TicketID$, a validity period for the ticket, and optional authentication information to be used when the Client connects to the Hidden Server.

*Distributing tickets* For distribution of the tickets we must look at two different scenarios; authorized users only or also allowing anonymous users. The different vulnerabilities of these will be discussed in Sect. 4. In either scenario, although especially applicable to anonymous users, another distinction is whether traditional directory servers are used or distributed hash tables (DHTs). We first present distribution via directory servers since this is closer to the current usage over the Tor network.

By using the protocol described here the hidden service will be able to keep track of users and build a "reputation" for a user or a group of users. This is of course something an authenticated user might be subjected to by a service anyway, but we will use this to create QoS for both types of users, authenticated and anonymous.

Each time a user is connecting to a hidden service he either sends publicly known information (contained in a ticket publicly available at a Directory Server), or some authentication based on information he shares with the hidden service. Now the hidden service is able to set up different QoS based on what category the client is in. E.g. an authorized client can have access to a larger number of Valet nodes, Introduction Points or even just use higher bandwidth nodes in IPo and RP connections than an anonymous user will get. We can also imagine that a hidden service will use more trusted or higher bandwidth nodes in some of its tunnels based on this information. Anonymous users may also be given different QoS based on previous experience, e.g. an anonymous user connecting for the thousandth time could get better (or worse?) QoS than a first time user. This would imply either a pseudonymous user profile or a bearer instrument for tracking reputation.

In the existing hidden service design one major problem is that everyone can find all the Introduction Points to any service for which they know the *.onion* address. Another threat to security is that the Directory Server is able to identify and count all services and their startup times, and in addition locate all their contact information since the listings are not encrypted (but signed). We suggest in this paper a simple countermeasure addressing these issues.

First we must look at the two different scenarios:

1. The service is to be publicly available if a client knows the *.onion* address.
2. The service is open to authorized clients with valid tickets already received, e.g. through an off-line distribution channel, and will not use the Directory Servers.

The latter scenario is managed by the functionality of the network. As noted, ticket control is then maintained through the connections, but to counter the first scenario we propose the following simple scheme.

The Client has somehow (e.g. by a link on a web page, phone call, letter, etc.) received the $q.onion$ address of the service to contact, where $q$ is derived from the public key of the service, e.g. $q = hash(PK + value)$. We use this address to create the service descriptor index, e.g. $hash(q +' 1')$, to use at the Directory Server. The downloaded value, $Q$, is the Content Information ticket encrypted with a symmetric encryption scheme using a key derived from the public key, e.g., $hash(q +' 2')$. So both the descriptor index and the descriptor content are hidden from the Directory Server. Now a client must be in possession of the public key (or the $q.onion$) address of the hidden service in order to find and decrypt its Contact Information ticket. After receiving $Q$, the Client extracts the content, finds the public key, checks whether the signature matches, and does a confirmation to see if this key really is the one corresponding to the $q.onion$ address. If so, she has confirmed receiving correct contact information for this service.

Since there is no way of deriving $hash(q +' 2')$ from $hash(q +' 1')$ without having $q$, the Directory Server cannot find the contact information of unknown $.onion$ addresses.

So what about private entries? If we want to permit groups of users to connect with different QoS, the hidden service gives them different cookies, grouping them so they e.g., can use $hash(q +' 1' + cookie)$ for lookup, and $hash(q +' 2' + cookie)$ for decrypting. Now we are also making it impossible for the Directory Server to count how many services it has listed. In addition, the cookies can be based on the client's authentication data, enabling only that specific client to download and decrypt the associated Contact Information ticket.

The scheme should also be expanded by using a date/time value inside the hash calculation to include a time period, e.g., current date, so a listing can exist anonymously without revealing when the service started to exist. Combining this with a time stamp could have the Directory Server store the entry for a longer (or shorter) period of time than default. And of course for authenticated users we only need to give the Client several Contact Information tickets with varying lifetimes. Typically any client should always have a long-term ticket and one or more short-term tickets.

In order to verify an update of information inside the directory service during the entry's life time, we propose a simple reverse hash chain scheme where the initial contact to the Directory Server is followed by an iterated hash value, $v_n = hash^n(v)$, known only to the hidden service itself. For each update of this index (e.g. of $hash(q +' 1' + date)$) the new encrypted ticket is accompanied by the value, $v_{k-1}$, enabling the directory server/DHT to verify the update using $v_k = hash(v_{k-1})$.

To adapt to the current and future improvements in hash collision techniques it is probably wise to increase the number of bits used in the .onion address from today's 80-bit (16 * base32(5-bit) characters) address to e.g. 256-bit (using 44 * base64[9](6-bit) characters, including an eight-bit version and extension value

[9] Use "special" base64 e.g., '/'→'-', '+'→'_'

as the first byte of the address). An evaluation of hash algorithms will not be discussed here.

*Distributed Hash Tables* The list of Tor servers is sensitive in at least two ways. First, it is the means by which clients bootstrap into anonymity: we cannot assume that clients can anonymously obtain an initial list of Tor nodes. Second, the list of nodes is sensitive. If different sets of nodes are given to different users, then it is possible to separate the source of traffic according to the nodes carrying it.

Neither of these is as much of an issue for hidden services. The IP addresses of clients acquiring information on available hidden services may be assumed to already be anonymized. Since there is neither a need for, nor a preexisting expectation of, an authoritative list of hidden services, partitioning is less of an issue as well. In any case, there is nothing to prevent someone from listing at directory servers two or more distinct sets of information for the same hidden service and selectively announcing one or the other set to distinct individuals.

For this reason, authoritative directory servers for hidden services as a core part of the Tor network are not necessary. Indeed, the primary motivation for their use initially has been one of convenience. They are an available and already used infrastructure for Tor users that distributes one kind of server information (Tor nodes). So, it was easy to add another kind (hidden servers). Obviously it would be better not to overload these servers in either functionality or workload.

One could set up another set of directory servers specifically for hidden services, but given the above considerations, hidden services would seem to be an advantageous application for distributed hash tables (DHTs), such as CAN [20], Chord [26], Pastry [23], or Tapestry [30].

DHTs are decentralized systems that can be used to support many applications including file storage and retrieval. They are known for their efficient tradeoffs of decentralization, scalability, robustness and routing efficiency. Their application in anonymous communication is not as straightforward as is sometimes supposed, and require careful design to avoid security pitfalls [5, 9]. Presenting a careful design is beyond the scope of this paper. However, we note that the basic design of *.onion* addresses based on hashing of keys are naturally amenable to DHTs.

## 4 Security

### 4.1 Availability

The greatest threat to availability in the original hidden service design is the collection of all Introduction Points in the information stored on the Directory Server. This makes it easy to either *directly* attack the Introduction Points, or *indirectly* attack them, e.g. by threatening the Directory Server or the Introduction Point operator not to list a specific service.

By using tickets and Valet nodes we have enabled the possibility of a hidden service to exist entirely on its own without anyone being able to identify it unless he can either guess or by some other method get ahold of the public key (or the *.onion* address) of the service. If a group of people can distribute the *.onion* address in private, no one should be able to find the service used, or know of its very existence.

We have addressed two types of hidden service users – *the authorized user* and *the anonymous user*. The hidden service might want to offer different QoS to the different users, in addition to the persistent and general need to provide availability. The anonymous user might allow himself to be recognized as a previous user of the service without revealing his identity, much like the use of *cookies* in HTTP[12]. For example, such an anonymous user might over time be considered trustworthy (enough) to be given higher availability and bandwidth than plain anonymous users. So we can have QoS also for anonymous connections. Availability and QoS for an authorized user can be very flexible, based on what the Hidden Servers require.

In addition, in our design it is impossible for the Directory Servers to count and identify the services as they can in the currently deployed hidden service system. Further, they will now only be able to extract the total number of "client groups"—not likely to be useful information. More important is the removal of the Directory Server's ability to confirm the existence of a service. Now a service will be able to announce its Contact Information on a Directory Server and remain private/unannounced.

But most important, the Introduction Points are hidden from the users, from the Directory Servers, and from the public. We have thus enhanced the availability of the Introduction Points. If the Contact Information tickets are publicly accessible, we must assume that a Valet node or an Introduction Point with reasonable effort will be able to determine which hidden service is using that Introduction Point. But only knowing one or two Introduction Points still leaves the others available for use.

*Finding the Introduction Points* is still possible for an adversary. If we select a small number of valet nodes this will move us closer to the original hidden service design, as the valet nodes now are the vulnerable points of a potential DoS attack. If the number of valet nodes is huge, we make it easier for an adversary to collect all the introduction points through owning a valet node in all "groups" associated with each introduction point. So we have to find a trade-off, and we expect to be better off in the lower count case.

Using $n$ as the number of nodes in the network, $c$ as the number of compromised nodes, $i$ as the number of Introduction Points, and $v$ as the number of Valet nodes per Introduction Point, we can get an expression for the probability of revealing all the Introduction Points to the adversary. The probability $P_s$ for a specific combination of $c$ compromised nodes in $i+1$ groups is given by Eq. 1.

$$P_s(x_j \text{ in valet group } j) = \frac{\binom{G_0}{x_0}\binom{G_1}{x_1}\cdots\binom{G_i}{x_i}}{\binom{n}{c}} = \frac{\binom{n-i\cdot v}{c-x_1-\ldots-x_i}\binom{v}{x_1}\cdots\binom{v}{x_i}}{\binom{n}{c}} \qquad (1)$$

Here 0 is the index for being outside all the valet node groups, i.e., $G_0 = n - i \cdot v$, $x_0 = c - x_1 - x_2 - \ldots - x_i$. and all other $G_j$ are given to be $v$. The number of compromised nodes $c$ must be larger or equal to the number of introduction points $i$, otherwise the probability will be zero.

$$P(n, c, v, i) = \sum_{x_1, x_2 \ldots, x_i} \frac{\binom{n-i \cdot v}{c - \sum_1^i x_j} \binom{v}{x_1} \cdots \binom{v}{x_i}}{\binom{n}{c}} \tag{2}$$

The probability of the adversary having concurrent presence in all groups is given by Eq. 2 where $c \geq i$ and we sum over the values: $x_1 = 1, \ldots, min(v, c - i + 1)$; $x_2 = 1, \ldots, min(v, c - i - x_1 + 2)$; $x_3 = 1, \ldots, min(v, c - i - x_1 - x_2 + 3)$; up to $x_i = 1, \ldots, min(v, c - \sum_{j=1}^{i-1} x_j)$, where the upper limit of the x-values is $v$ as indicated.

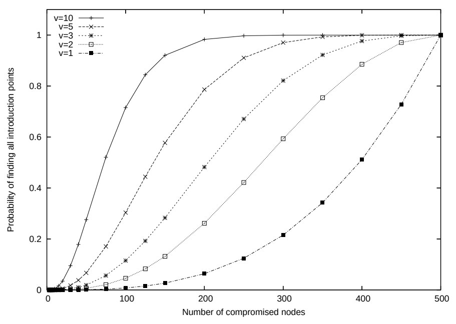

**Fig. 3.** Probability of finding all *i=3* introduction points each using $v$ valet nodes in a *n=500* network

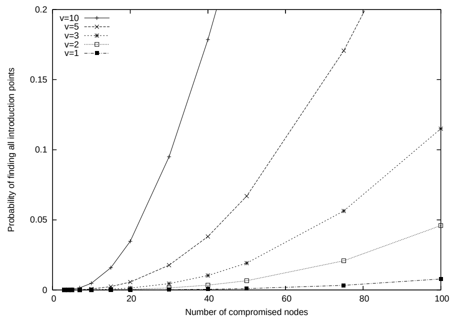

**Fig. 4.** Lower left corner of Fig. 3

Figure 3 and 4 shows how the use of valet nodes hides the introduction points in the current system using three IPos. The probability is plottet against the number of compromised nodes, $c$, in a system with $n = 500$ nodes in total. Even when using ten valet nodes per IPo the adversary must control 25 nodes in order to have a 10% chance of locating all three introduction points. Using only three valet nodes per IPo the adversary must control almost 100 nodes to achieve the same probability.

As described in Sect. 2.2 the probability of locating all introduction points for a hidden service is 1 in the current implementation independent of how many network nodes the adversary controls. Figure 5 shows the probability of an adversary, controlling $c$ nodes of the network, to be able to locate all the $i$ introduction points when using valet nodes. The more valet nodes added per IPo, the higher the probability of locating all (presence in all groups), and if we add more IPos keeping the number of valet nodes constant the probability goes down. We observe that the number of valet nodes is a more significant factor than the number of introduction points. E.g. the strongest protection

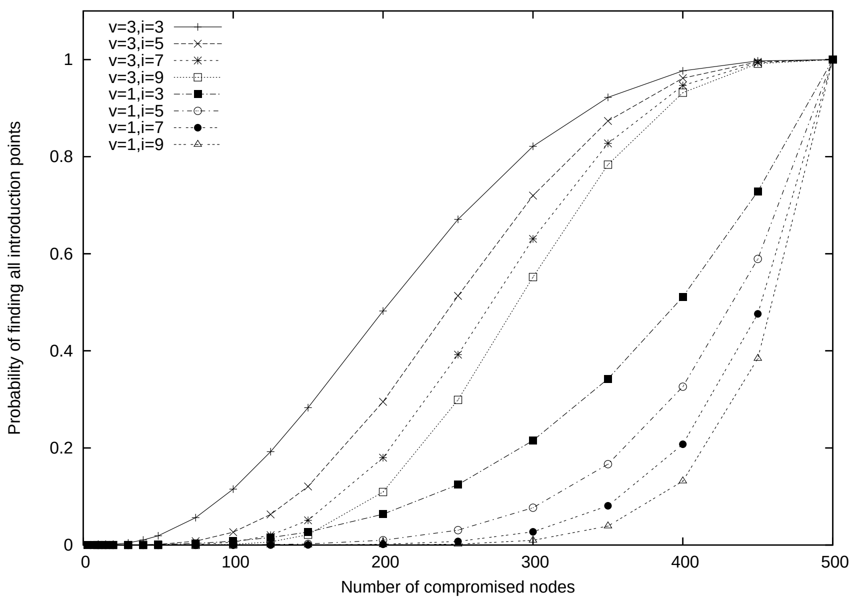

**Fig. 5.** Probability of finding all $i$ introduction points, each using $v$ valet nodes in a network of $n$=*500* nodes

occurs in the case of using only a single valet node per IPo, but this will as previously mentioned affect the service's availability. Using only one valet node gives an insider adversary the same number of nodes to attack. We also observe that when using nine introduction points and only one valet node per IPo, the adversary will have to control around 400 nodes in order to have the same 10% probability of locating them all.

So this is a question of balancing availability with security (*anonymity*). Even if the service is now involving more nodes in the network, unavailability will only happen if all introduction points are down or if the combination of either all the IPos or their associated valet nodes are down at the same time. Given that the introduction points now are hidden from the public, we find that the removal of targeted DoS attacks is more significant than the introduction of more nodes. In Fig. 6 we compare the relative distributions of a network of 100 and 1000 nodes and observe only tiny variations in the probability distribution caused by the changing relative sizes of $i$ and $v$ compared to $c$ and $n$.

Based on this we estimate that good protection of the service should consist of at least three introduction point combined with at least two valet nodes per introduction point, and should be combined with the possibility of differentiated QoS as described in Sect. 4.2.

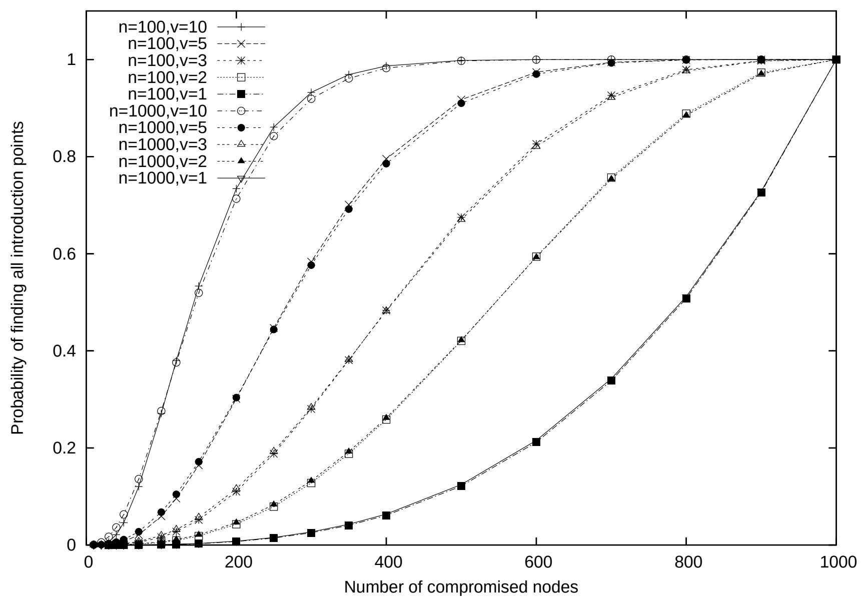

**Fig. 6.** Comparing *n=1000* to an upscaled *n=100* for a service using three introduction points with varying number of valet nodes.

## 4.2 Quality of Service

As described in Sect. 3.2 we can now differentiate QoS for the users. But there are potential problems with the described methods.

If a user wants to stay anonymous and untraceable he must start with a public ticket every time (the paranoid variant), or trust the service to supply semi-public tickets for every connection, which e.g. the user can check by connecting multiple times using the public ticket(s). As these are open public services connected to by anonymous clients this is a simple verification.

When it comes to authorized users, a service may access the Rendezvous Point through different nodes giving a specific ("deserved") bandwidth to the user. This might reduce the set of nodes the service is selecting from during set up of the tunnel.

## 4.3 MitM Attack

The Valet Service node will not be an additional danger for performing a man-in-the-middle attack because the Client is able to authenticate the Introduction Point by the use of its service key pair. And, as in the current version of the hidden service protocol, the Introduction Point will not be able to perform a MitM attack due to the authentication of the hidden service.

## 4.4 DoS Attacks

A client connecting multiple times to a Valet node and sending messages requesting decryption of the Valet Token and extension of circuits may cause a problem for the Valet node. But a Valet node should only need to unpack a Token once, and then may cache it for later reuse within its lifetime. The other actions are normal extension, involving the Client, and simple forwarding of information, so this should not be easy to abuse. By not accepting extension requests twice for the same tunnel, the attacker may be forced to set up a new tunnel through another node before every attempt.

Sending multiple connection requests to the Introduction Point could be a potential problem. But if all Introduction Points tear down their connection circuits upon finishing a connection, the Client will be forced to perform too many operations compared with the effect created on the Introduction Point. If it becomes necessary, an Introduction Point can simply ignore service requests from specific Valet nodes.

If the chosen Valet node is down, a Client simply chooses another Valet node in the ticket. If they are all down or unavailable, this will affect every client using the same ticket. But the Valet nodes should be as many and constructed in such a way that it is possible for a user to either use a long term ticket previously received, or use an anonymous ticket for a first connection. There should be enough variety in the construction of the tickets to make it prohibitively difficult for an adversary to take down all Valet nodes of a service even if she did know them all.

This should apply to the Introduction Points as well, so we must make certain that Valet nodes only know the same Introduction Point for the same service to minimize this threat factor.

A large threat to the public tickets scheme is uploading of false tickets to the Directory Servers (or into the DHT). For known services it is easy to counter by signing, but for encrypted tickets this is an issue. We proposed a reverse hash chain scheme (Sect. 3.2) to counter this since the false updates will be invalid. But the scheme will raise issues in synchronization of directory servers and DHTs, which we will not address here.

## 4.5 Ticket Problems

If a client lost its ticket or the ticket expires, it must either use a long term ticket, or download a new contact information ticket from the network. If authentication is required, or if the client has a privileged QoS, this can now be resubmitted and the next ticket received should get the client back on the same level of QoS. Another way of restoring QoS even to anonymous users is to store this information in separate cookies alongside the tickets.

So what if the public key of the hidden service has become exposed to someone who should not have it? Will we have any possibility to hide the hidden service again once we are "found"? Not with today's design. But if we require authentication and there is suspicion of someone knowing the .onion address and thereby having the Contact Information (e.g. by lots of valid connection requests with false authentication information), the protocol could be extended to distribute new public key information with the new Contact Information upon the next authenticated request.

### 4.6 Colluding connection nodes

If one of the service's Valet nodes happens to be colluding with the Client, it will be able to collect information about at least one Introduction Point to this service. This takes us to the same scenario as in the current version of the hidden service design, except that now we also have several other Introduction Points that will continue to route users to the service thus maintaining availability. Adversary Alice must now own a node in every set of Valet Service nodes used for the Introduction Points, something which should be highly unlikely if the grouping of nodes are constructed with this in mind.

Alice will not be able to find out if she owns an Introduction Point unless she also owns the corresponding Valet node or the Client. As a Valet node she will know if she also controls the Introduction Point, but she will not be able to confirm which service this is until she gets the corresponding ticket. As a Client there is the possibility of matching the DH-exchange and thereby determine if she controls the Introduction Point. But even if Alice should happen to find one of her nodes as an Introduction Point this would still not reveal any additional information over the currently deployed system, where a node can trivially know the service for which it is chosen to be an Introduction Point.

## 5 Conclusion

Hidden services are now widely deployed and of increasing interest for individuals, corporations, governments, and organizations. We have here presented an extension of the current hidden service design that improves availability and resistance to DoS through the introduction of valet nodes, which hide service introduction points. The new design also facilitates the use of "completely hidden services": only clients that know a hidden service's *.onion* address or its public key will be able to connect to it or even verify it's existence. The new protocol also allows differentiation of the quality of service given to clients, regardless of whether they are anonymous or authenticated.

## References

1. Ross J. Anderson. The eternity service. In *Proceedings of Pragocrypt '96*, 1996.
2. The Anonymizer. http://www.anonymizer.com/.
3. Krista Bennett and Christian Grothoff. GAP – practical anonymous networking. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. Springer-Verlag, LNCS 2760, March 2003.

4. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.
5. Nikita Borisov. *Anonymous Routing in Structured Peer-to-Peer Overlays.* PhD thesis, UC Berkeley, Spring 2005.
6. Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.
7. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
8. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.
9. George Danezis, Chris Lesniewski-Laas, M. Frans Kaashoek, and Ross Anderson. Sybil-resistant dht routing. In *Computer Security – ESORICS 2005*, September 2005.
10. Roger Dingledine, Michael J. Freedman, and David Molnar. The Free Haven Project: Distributed anonymous storage service. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability.* Springer-Verlag, LNCS 2009, July 2000.
11. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
12. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. IETF RFC 2616, June 1999.
13. Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet.* PhD thesis, UC Berkeley, December 2000.
14. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.
15. Aviel D. Rubin Marc Waldman and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.
16. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy.* IEEE CS, May 2005.
17. Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy.* IEEE CS, May 2006.
18. Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.
19. Proxify.com. http://www.proxify.com/.
20. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

21. Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.
22. Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.
23. Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.
24. Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *Computer Security – ESORICS 2003*, October 2003.
25. Angelos Stavrou and Angelos D. Keromytis. Countering DoS attacks with stateless multipath overlays. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 249–259, New York, NY, USA, 2005. ACM Press.
26. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.
27. Marc Waldman and David Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 126–135, November 2001.
28. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.
29. Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 7(4):489–522, 2004. A preliminary version of this paper appeared in [28].
30. Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.