Implementierung und Evaluierung von versteckten Diensten basierend auf offenen Standards

eingereicht von

André Egners
Matrikelnummer: 251122

Gutachter: Prof. Dr. rer. nat. Dr. h.c. Otto Spaniol
Zweitgutachter: Prof. Dr.-Ing. Ulrike Meyer
Betreuender Assistent: Andriy Panchenko, M.Sc.
Abgegeben am: 01.07.2009
Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Datum, Unterschrift
Inhaltsverzeichnis

1 Einleitung .......................... 1
   1.1 Motivation ........................................ 1
   1.2 Problembeschreibung ..................... 2
   1.3 Ziele und Vorgehensweise .................. 3
   1.4 Übersicht der Diplomarbeit ............. 4

2 Grundlagen .......................... 5
   2.1 Tor & Onion-Routing ................. 5
   2.2 Tor Hidden Services .................. 8
      2.2.1 Protokollablauf ...................... 8
      2.2.2 Komponenten ..................... 10

3 Verwandte Ansätze .................. 13
   3.1 Leistungsanalyse .......................... 13
   3.2 Valet-Services ......................... 14
      3.2.1 Protokollablauf .......................... 15
      3.2.2 Alternative mit optionalem Rendezvous Point .... 16
      3.2.3 Valet-Tickets ...................... 17
      3.2.4 Ticketverteilung .................. 18
      3.2.5 Sicherheitsimplikationen ........ 19
   3.3 Weitere Verbesserungsvorschläge ...... 20
   3.4 Enttarnung von Hidden Services ......... 21
   3.5 Deskriptorformate ....................... 23
      3.5.1 Tor Hidden Service Deskriptor ........ 23
      3.5.2 Valet-Ticketsystem ............... 24
   3.6 Informationsverteilung .................. 24
      3.6.1 Tor Verzeichnisserver ............. 25
      3.6.2 Verteilte Hash-Tabellen .......... 25
   3.7 Alternative Hidden Service Konzepte ... 26
      3.7.1 I2P - Invisible Internet Project .... 26
         3.7.1.1 Funktionsweise .................. 26
         3.7.1.2 Eepsites ...................... 27
      3.7.2 Freenet .............................. 28
         3.7.2.1 Einfügen von Datensätzen ....... 28
         3.7.2.2 Abrufen von Datensätzen ....... 29
   3.8 DeleGate .............................. 29
3.8.1 Virtual Socket Association Protokoll (VSAP) .................................. 30
  3.8.1.1 Problematik .................................................. 30
  3.8.1.2 Vorteile ..................................................... 30
  3.8.1.3 Zusammenfassung ......................................... 31

3.8.2 HTTP Protokoll Erweiterung ......................................................... 31
  3.8.2.1 Problematik .................................................. 31
  3.8.2.2 Vorteile ..................................................... 31
  3.8.2.3 Zusammenfassung ......................................... 31

3.8.3 Zusammenfassung ................................................................. 31

3.9 SOCKS ................................................................. 32
  3.9.1 Problematik ..................................................... 32
  3.9.2 Vorteile ......................................................... 33
  3.9.3 Zusammenfassung ............................................... 33

3.10 Reverse Proxy ............................................................... 33
  3.10.1 Problematik .................................................... 33
  3.10.2 Vorteile ....................................................... 34
  3.10.3 Zusammenfassung ........................................... 34

3.11 Multiplexingprotokolle .......................................................... 34
  3.11.1 SockMUX ......................................................... 34
  3.11.2 Session Control Protokoll (SCP) .................................. 35
    3.11.2.1 Aufbau .................................................. 35
    3.11.2.2 Zusammenfassung .................................... 37

4 Das Shalon Hidden Service Protokoll .................................. 39
  4.1 Shalon - Aktueller Stand ...................................................... 39
  4.2 Designübersicht .......................................................... 40
  4.3 Erweiterte Proxy Server Deskriptoren .................................. 42

4.4 Hidden Service Betriebsmodi .................................................. 42
  4.4.1 TLS-Gateway Modus ............................................... 42
  4.4.2 Open/Public Modus ............................................. 44

4.5 Hidden Service Konfiguration ............................................... 45

4.6 Hidden Service Deskriptoren ................................................. 46

4.7 Hidden Service Kontaktmöglichkeiten .................................. 49
  4.7.1 Shalon Clients .................................................... 49
  4.7.2 Non-Shalon Clients ............................................. 50

4.8 Deskriptorabruf ............................................................. 50
  4.8.1 Shaloninterner Deskriptorabruf ................................ 51
  4.8.2 Webschnittstelle ............................................... 51

4.9 Contact Point Funktionsweise .............................................. 54
  4.9.1 Übersicht .......................................................... 54
  4.9.2 Implementierung ................................................. 54

4.10 Quality of Service .......................................................... 56

4.11 WebMUX ................................................................. 56
  4.11.1 Ziele ............................................................. 57
  4.11.2 Protokollverhalten ............................................ 57
  4.11.3 Packetaufbau .................................................. 58
  4.11.4 Verbindungsmanagement ....................................... 59
# Inhaltsverzeichnis

<table>
<thead>
<tr>
<th>Kapitel</th>
<th>Titel</th>
<th>Seite</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11.5</td>
<td>Kontrollnachrichten</td>
<td>59</td>
</tr>
<tr>
<td>4.11.6</td>
<td>Zusammenfassung</td>
<td>60</td>
</tr>
<tr>
<td>4.12</td>
<td>Multiplexing-Circuits</td>
<td>60</td>
</tr>
<tr>
<td>4.13</td>
<td>Protokollablauf</td>
<td>61</td>
</tr>
<tr>
<td>4.13.1</td>
<td>Hidden Service Setup</td>
<td>61</td>
</tr>
<tr>
<td>4.13.2</td>
<td>Shalon Client Kontakt</td>
<td>62</td>
</tr>
<tr>
<td>4.14</td>
<td>Zusammenfassung</td>
<td>63</td>
</tr>
<tr>
<td>5</td>
<td>Evaluation</td>
<td>65</td>
</tr>
<tr>
<td>5.1</td>
<td>Versuchsaufbauten</td>
<td>65</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Laborumgebung</td>
<td>65</td>
</tr>
<tr>
<td>5.1.2</td>
<td>PlanetLab</td>
<td>66</td>
</tr>
<tr>
<td>5.1.2.1</td>
<td>Technologie</td>
<td>66</td>
</tr>
<tr>
<td>5.1.2.2</td>
<td>Versuchsaufbau</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>Proxy-Performanzanalyse</td>
<td>67</td>
</tr>
<tr>
<td>5.3</td>
<td>DHT Suchoperationen</td>
<td>71</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Variation der Netzgröße</td>
<td>71</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Langlaufende Deskriptorsuche</td>
<td>73</td>
</tr>
<tr>
<td>5.4</td>
<td>Labor Messungen</td>
<td>74</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Hidden Service Clients</td>
<td>74</td>
</tr>
<tr>
<td>5.4.1.1</td>
<td>Non-Shalon Hidden Service Client</td>
<td>74</td>
</tr>
<tr>
<td>5.4.1.2</td>
<td>Shalon Hidden Service Client</td>
<td>75</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Verbindungsdurchsatz</td>
<td>75</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Round Trip Zeiten</td>
<td>77</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Hidden Service Setup</td>
<td>77</td>
</tr>
<tr>
<td>5.5</td>
<td>PlanetLab Messungen</td>
<td>79</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Hidden Service Clients</td>
<td>80</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Verbindungsdurchsatz</td>
<td>81</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Round Trip Zeiten</td>
<td>82</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Hidden Service Setup</td>
<td>82</td>
</tr>
<tr>
<td>5.6</td>
<td>Tor Vergleichsmessungen</td>
<td>83</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Hidden Service Clients</td>
<td>84</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Verbindungsdurchsatz</td>
<td>86</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Round Trip Zeiten</td>
<td>87</td>
</tr>
<tr>
<td>5.6.4</td>
<td>Hidden Service Setup</td>
<td>87</td>
</tr>
<tr>
<td>5.7</td>
<td>WebMUX Evaluation</td>
<td>89</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Durchsatz</td>
<td>90</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Round Trip Zeiten</td>
<td>90</td>
</tr>
<tr>
<td>5.8</td>
<td>Zusammenfassung</td>
<td>91</td>
</tr>
<tr>
<td>6</td>
<td>Sicherheitsanalyse</td>
<td>93</td>
</tr>
<tr>
<td>6.1</td>
<td>Angreifermodell</td>
<td>93</td>
</tr>
<tr>
<td>6.2</td>
<td>Angriffsszenarien</td>
<td>94</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Deanonymisierung</td>
<td>94</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Denial of Service</td>
<td>95</td>
</tr>
<tr>
<td>6.3</td>
<td>Unterschiede zu Tor Hidden Services</td>
<td>96</td>
</tr>
<tr>
<td>6.4</td>
<td>Zusammenfassung</td>
<td>97</td>
</tr>
</tbody>
</table>
7 Zusammenfassung & Ausblick

7.1 Erreichte Ziele .............................................. 99
7.2 Ausblick .................................................. 101
    7.2.1 Contact Point Protokoll ......................... 101
    7.2.2 Sichere Deskriptoren .................................. 102
    7.2.3 Replikation von Deskriptoren ....................... 102
    7.2.4 Server Circuit Multiplexing .......................... 103
    7.2.5 Guard Nodes ........................................... 103
    7.2.6 Quality of Service der Contact Points ................ 104
    7.2.7 Alternative Implementierung ....................... 104
7.3 Offene Probleme ........................................... 104

Abbildungsverzeichnis ........................................... 109

Literaturverzeichnis ............................................ 113

A Anhang

A.1 DHT Messungen ........................................... 115
A.2 Hidden Service Setup & Hidden Service Contact ................. 119
KAPITEL 1

Einleitung

In diesem Kapitel wird eine Motivation für diese Diplomarbeit gegeben. Außerdem wird auf die hier vorhandene Problemstellung näher eingegangen und die Ziele und der vorgeschlagene Lösungsansatz kurz erläutert. Anschließend wird zusätzlich ein Überblick über die weiteren Kapitel dieser Diplomarbeit gegeben.

1.1 Motivation

In der heutigen Gesellschaft ist die Privatsphäre von Personen ein schützenswertes und zunehmend wichtiger werdendes Gut. Schon in der UN-Menschenrechtscharta wurde wie folgt auf die Privatsphäre hingewiesen:

"No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks.“(UDHR1 - Article 12)

Analysen können erstellt werden, um vertrauliche Daten oder Informationen zu den Beziehungen Nutzer untereinander zu sammeln. Diese Daten können nicht nur die

1Universal Declaration of Human Rights
Sicherheit der Nutzer gefährden, sondern stattdessen bestimmten Interessengruppen einen hohen finanziellen Nutzen bieten. Es ist möglich, diese Verbindungsdaten so zu verschleiern, dass es nicht ohne geringen Aufwand möglich ist, Beziehungen zwischen Kommunikationspartnern zu inferieren. Hierbei muss die Verschleierung der Daten nicht auf bestimmte Anwendungsprotokolle beschränkt sein.


1.2 Problembeschreibung

Anonyme Kommunikation im Sinne der Sender-Anonymität, ist heute über verschiedene Anonymisierungsdiensle möglich. Um die Responder-Anonymität zu erreichen, stellen die Tor Hidden Services den populärsten Ansatz dar. Diese erlauben es durch einfache Konfiguration, dem Benutzer einen Hidden Service, von seinem lokalen Rechner aus anzubieten. Die Tor Software stellt anschließend über einen komplizierten Mechanismus diesen Dienst im Tor Netzwerk zur Verfügung. Dieser angebotene Dienst ist nur über das Tor Netzwerk erreichbar. Möchte man mit die-
1.3 Ziele und Vorgehensweise


Neben der Erweiterung der Shalon Software werden weiterhin folgende Punkte untersucht: Mit geeigneten Zertifikaten für den Hidden Service, oder den Proxy Server der als Kontakt punkt dient, muss Schutz vor Impersonation Angriffen geboten werden. Es muss sichergestellt werden, dass der Benutzer eines Hidden Services tatsächlich mit diesem kommuniziert und nicht auf einen Hidden Service des Angreifers weitergeleitet wird. Weiterhin wird die Möglichkeit der Ende-zu-Ende Verschlüsselung...

1.4 Übersicht der Diplomarbeit

KAPITEL 2

Grundlagen

Im Folgenden wird das Hidden Service Konzept, wie es im Tor Netzwerk existiert beschrieben. Zusätzlich werden hier die fundamentalen Bestandteile des Tor Netzwerkes und die Grundlagen anonymer Kommunikation sowie der eingesetzten Techniken erläutert.

2.1 Tor & Onion-Routing

Das Tor Netzwerk bildet ein Overlay-Anonymisierungsnetzwerk\footnote{Ein Overlay-Netzwerk bildet ein logisches Netzwerk das auf eine bestehende Netzinfrastruktur aufsetzt} dass es Kommunikationspartnern erlaubt, anonym über das Netzwerk zu kommunizieren. Hierbei bietet Tor nur Anonymität des Clients gegenüber dem Server, die sogenannte \textit{Sender Anonymity}. Es bietet Schutz vor Analyse der Verbindungsdaten im Internet, also der Analyse des Netzwerkverkehrs. Der gebotene Schutz vor solchen Verkehrsanalysen, die auf Netzwerkpacketebene arbeiten, beschränkt sich auf \textit{nicht-globale} Angreifer. Der Angreifer ist nicht in der Lage, das gesamte Netz zu beeinflussen und an jeder Stelle Nachrichten abzuhören. Durch genaue Analyse des Paketkopfs der TCP-Pakete, lassen sich Routinginformationen (z.B. Quelle, Ziel, Größe) über das Paket herleiten, die diese im Gegensatz zu den Nutzdaten nicht verschlüsselt werden können. Durch ausreichende Aggregation von Paketkopfinformationen lassen sich Kommunikationsmuster des überwachten Subjets herleiten, welche Aufschluss darüber geben, mit wem kommuniziert wird. Um eine solche Analyse erfolgreich durchführen zu können, muss der Überwacher direkten Zugriff auf die Netzwerkpakete haben. Handelt es sich bei dem Überwacher beispielsweise um einen ISP\footnote{Internet Service Provider} stellt dies kein Problem dar. Tor ist der Lage diese Verbindungsinformationen so zu verschleiern, dass es nicht ohne Weiteres möglich ist, Beziehungen von Kommunikationspartnern herzuleiten. Die Verschleierung der Daten ist nicht auf bestimmte Anwendungsprotokolle beschränkt, sondern bietet die Möglichkeit, beliebige, auf TCP basierende Protokolle zu verschleiern.

Diese Relays werden genutzt, um sogenannte *Tor-Circuits* aufzubauen. Ein solcher Tor-Circuit beinhaltet in der Regel drei Tor-Relay Hops und die Kommunikation wird jeweils in drei Schichten verschlüsselt. Beim Aufbau eines solchen Tor-Circuits vereinbart der Client mit jedem der OR auf dem Pfad einen Schlüssel, der für die Kommunikation genutzt wird. Bei diesem handelt es sich um einen über Diffie-Hellmann vereinbarten symmetrischen Schlüssel. Eine Nachricht wird so verpackt, dass jeweils nur der aktuelle OR die für ihn bestimmte Schicht mit dem beim Circuit-Aufbau vereinbarten Schlüssel, entschlüsseln kann. Dieser schickt die Nachricht wei-

3Stand Oktober 2008
2.1. Tor & Onion-Routing


Abbildung 2.2: Onion-Routing


Verbindungen transparent zu tunneln. Für das Senden von Daten über HTTP wird zusätzlich ein lokaler Filter-Proxy (z.B. Privoxy, Polipo) betrieben, welcher rudimentäres Filtern auf Anwendungsschicht ermöglicht.

2.2 Tor Hidden Services

Das Hidden Service Protokoll von Tor dient dazu, es einem Dienstanbieter zu ermöglichen, anonym einen Dienst anzubieten. Im Gegensatz zu regulären Tor-Clients steht hierbei die sogenannteResponder-Anonymität im Vordergrund. Da hierbei die IP-Adresse des Dienstes vor den Clients verborgen wird, ist der Dienstanbieter nicht nur anonym, sondern kann auch gegen Denial of Service Angriffe geschützt werden. Bei der Entwicklung des Hidden Service Konzepts von Tor wurde auf vier Kernpunkte besonderen Wert gelegt:

- **Zugriffsschutz:** Der Dienstbetreiber muss in der Lage sein, Anfragen zu filtern, um keinem Flooding zum Opfer zu fallen.
- **Robustheit:** Dem Dienstbetreiber soll es möglich sein, über einen längeren Zeitraum unter demselben Pseudonym erreichbar zu sein. Weiterhin muss dies auch bei Ausfall von ORs möglich sein.
- **Verleumdungsschutz:** Die Knoten die bei der Verbindungsherstellung von Client und Hidden Service beteiligt sind, dürfen nicht für diese Tatsache verantwortlich gemacht werden können.
- **Anwendungstransparenz:** Für den Dienstbetreiber sollte es nicht nötig sein, seinen Dienst in irgendeiner Form speziell zu modifizieren, um als Hidden Service zu arbeiten.

2.2.1 Protokollablauf

Im Folgenden wird der Protokollablauf beschrieben, der nötig ist, um einen Hidden Service zu veröffentlichen und eine Anfrage entgegenzunehmen. Alle in Abbildung dargestellten Verbindungen sind anonymisiert, also Tor-Circuits mindestens der Länge drei. Die hier verwendeten Begriffe werden in Abschnitt erläutert.

1. Der Dienstanbieter erstellt ein Schlüsselpaar, mit dem sein Pseudonym eindeutig identifizierbar ist. Außerdem wählt er zufällig einige ORs aus, die als Introduction Points fungieren sollen.


---

4http://www.privoxy.org/
5http://www.pps.jussieu.fr/~jch/software/polipo/
6dt. Antwort-Anonymität
7Überflutung des Server mit sehr vielen Anfragen. Ähnliches Verhalten wie bei einem Denial of Service Angriff

4. Der Dienstnutzer erstellt einen Tor-Circuit zu einem zufälligen OR. Dieser Tor-Knoten soll als Rendezvous-Point fungieren.

5. Der Dienstnutzer einen weiteren Tor-Circuit zu einem der Introduction Points des Hidden Service auf.


8. Der Rendezvous Point verbindet die Circuits von Client und Hidden Service miteinander.

9. Dienstanbieter und Dienstnutzer können miteinander kommunizieren, ohne das
Kapitel 2. Grundlagen

der Rendezvous Point und die anderen an der Kommunikation beteiligten Knoten die Identität der beiden kennt.

2.2.2 Komponenten


Verzeichnisserver Zusätzlich zu den in Abschnitt \ref{section:verzeichnisserver} erläuterten Aufgaben, werden die Verzeichnisserver von Hidden Service Betreibern genutzt, um die benötigten Kontaktinformation zu publizieren. Dienstnutzer nutzen die Verzeichnisserver, um Informationen (z.B. Introduction Points) zu erhalten.


Tor-Circuit Tor-Circuits werden vom lokalen Onion-Proxy erstellt und stellen eine Verbindung standardmäßig über drei Hops von OR dar, welche gemäß dem Onion Prinzip \ref{section:tor_prinzip} gesichert sind. Durch Multiplexen von mehreren TCP Streams über einen Circuit wird nicht nur die Anonymität erhöht, sondern es werden auch Ressourcen gespart, die für die kryptographische Sicherung des Circuits beim Aufbau benötigt werden.

Tor-Knoten (Onion Router & Onion Proxy) Tor-Knoten sind Teil des Overlay-Netzwerks, das aus der Menge der Tor-Knoten und der Directory-Server gebildet wird. Viele dieser Knoten sind untereinander über eine TLS Verbindung miteinander verbunden. Der lokale Onion-Proxy (OP) sorgt dafür, dass Verzeichnisinformationen geladen und Circuits erstellt werden können. Weiterhin verfügt der OR über einen Langzeitschlüssel, der seine Identität abbildet, sowie einen Kurzeitschlüssel, welcher
2.2. Tor Hidden Services

für die Verschlüsselung der Tor Onion-Nachrichten genutzt wird. Zusätzlich wird der Langzeitschlüssel genutzt, um die TLS Zertifikate und den OR Deskriptor zu signieren. Der Deskriptor eines OR enthält einige Informationen wie zum Beispiel dessen Schlüssel, Adresse und Bandbreite.

Es wird außerdem auf alternative Ansätze zur anonymen Veröffentlichung und Verteilung von Information, wie I2P \[11\] und Freenet \[8\] eingegangen.

Neben der Diskussion der alternativen Ansätze werden zusätzlich einige Protokolle, die für das im nächsten Kapitel vorgestellte Design interessant sind, vorgestellt.

### 3.1 Leistungsanalyse


Weiterhin hängt die vom Benutzer tatsächlich wahrgenommene Qualität des Hidden Service Protokolls maßgeblich von dessen Reaktionszeit ab. Hierbei wurde die Anzahl der involvierten Knoten, die zur Kommunikation zwischen Client und Hidden Service benötigt werden, als sehr wichtiger Faktor identifiziert.

3.2 Valet-Services


Aufgrund des schlechten Schutzes der Introduction Points vor DoS-Angriffen, werden, neben einer neuartigen Informationsverteilung zu Kontaktaufnahme mit einem

\[1\]

Vorhandene Circuits werden erweitert statt vollständig neu aufgebaut

### 3.2.1 Protokollablauf

3.2.2 Alternative mitoptionalem Rendezvous Point


3.2. Valet-Services

auf dem genutzten Ticket, dynamisch ausgewählt werden. Der Hidden Service kann also entscheiden, ob direkt über die schon bestehende Verbindung kommuniziert werden soll, oder ob er einen Circuit zu dem vom Client spezifizierten Rendezvous Point erstellt.

Abbildung 3.2: Valet-Service ohne Rendezvous Point

3.2.3 Valet-Tickets


Kapitel 3. Verwandte Ansätze

Abbildung 3.3: Valet-Service mit optionalem Rendezvous Point


3.2.4 Ticketverteilung


z.B. ID1 = Hidden Service tunnelt über langsamen Pfad zum Rendezvous Point
statt der üblichen .onion-Adresse vor, einen Service-Deskriptor Index zu verwenden. Der Wert hash(q + '1') könnte zum Beispiel genutzt werden, um die Tickets auf dem Verzeichnis zu speichern und zu suchen. Der Inhalt des Valet-Tickets Q wird durch symmetrische Verschlüsselung geschützt. Der benutzte Schlüssel wird aus dem Wert hash(q + '2') abgeleitet. Somit ist sowohl der Service-Deskriptor Index, als auch dessen Inhalt (Q), vor dem Verzeichnisserver geschützt. Ohne im Besitz des öffentlichen Schlüssels des Hidden Services zu sein, woraus q berechnet wird, ist es also nicht möglich die nötigen Kontaktdaten zu finden und zu entschlüsseln. Neben diesen öffentlichen Einträgen sind aber auch private Einträge denkbar, zum Beispiel für den Fall, dass verschiedenen Benutzergruppen verschiedene QoS-Parameter zugeordnet werden sollen. Durch einen Cookie in der Berechnung (hash(q + '1' + cookie)) lassen sich Benutzer in Gruppen aufteilen. Dies hat außerdem zur Folge, dass es dem Verzeichnisserver nicht mehr möglich ist, die Anzahl der registrierten Hidden Services zu ermitteln, da keine eindeutige Abbildung von öffentlichen Schlüsseln zu Einträgen existiert.

Um ungültige Ticket(-Updates) zu verhindern, wird die Benutzung einer Reverse-Hash-Chain vorgeschlagen. Diese stellt sicher, das Updates jeweils aus Anwendung der Hash-Funktion auf den vorherigen Wert überprüft werden kann.

Vom Client verlorene oder abgelaufene Tickets müssen entweder durch neue ersetzt werden, oder es muss ein Langzeitticket vorhanden sein. Falls der öffentliche Schlüssel eines Hidden Services bekannt wird und zu viele Verbindungsanfragen mit falscher Autorisierung eingehen, sollte eine Möglichkeit existieren, neue Kontaktinformationen zu generieren und die nötigen Schlüssel zu verteilen.

Im nächsten Abschnitt wird auf die Sicherheitsimplikationen eingegangen, die sich durch den Einsatz von Valet-Nodes und dem Valet-Ticket System ergeben.

### 3.2.5 Sicherheitsimplikationen


---

5 $q = \text{hash}(PK + \text{value})$

6 hash(q + '2') kann nicht aus hash(q + '1') abgeleitet werden

7 $v_n = \text{hash}(v_{n-1})$

8 EXTEND: Tor Befehlt zum Erweitern eines Circuits
dungen beendet werden. Der Aufwand des Clients ist deutlich höher als der Effekt, der durch den Angriff erzielt werden kann.


3.3 Weitere Verbesserungsvorschläge


Aus der Perspektive des Hidden Services lohnt es sich, zusätzlich mehrere Circuits

### 3.4 Enttarnung von Hidden Services


Des Weiteren kann bei der Enttarnung zwischen dem Client- und Server-Szenario unterschieden werden. Beim Client-Szenario befindet sich der Hidden Service nicht auf einem Tor-Server sondern außerhalb des Tor-Netzes auf einem Client. Beim anderen Fall hingegen wird versucht, den Netzwerkverkehr zusätzlich durch den schon vorhandenen, zu schützen und zu verstecken. Der Angreifer kann also feststellen ob sich der Hidden Service außerhalb des Tor-Netzes befindet. Dies ist möglich, da alle Knoten des Netzes in einer zugänglichen Liste vorhanden sind und so Verbindungen von Client- und Server-Knoten unterschieden werden können. Falls
im zweiten Fall einzelne Streams zwischen Client und Hidden Service identifizierbar sind, lässt ein Vorgängerangriff durchführen. Über eine Statistik, welche IP-Adressen auf dem kontrollierten Knoten, bei gesuchtem Muster gesehen wurden, lässt sich bei einem Rendezvous-Circuit der Länge $m$ der Hidden Service identifizieren. Möglich wird dies, da eine IP-Adresse in ca. $\frac{1}{10}$ Fällen auftritt wenn der Hidden Service den ersten Knoten im Rendezvous-Circuit wählt. Falls $m$ sehr viel kleiner als die Gesamtanzahl der Netzwerk-Knoten ist, lässt sich der Hidden Service enttarnen. Neben diesem Angriff lässt sich über die RTT\(^{10}\) der vom Angreifer ausgehenden Anfrage, die physische Distanz zum Hidden Service schätzen.

Kontrolliert der Angreifer, neben einer Weiterleitung, auch noch den Rendezvous Point, können die Angriffe verbessert werden. Da der Rendezvous Point vom Angreifer gewählt wird und der Knoten für Weiterleitung für einen Rendezvous-Circuit in Frage kommt, ist der Angreifer in der Lage, festzustellen, ob dieser der zweite oder der letzte Knoten ist. So lässt sich leicht feststellen ob der Angreifer den ersten Knoten im Rendezvous-Circuit kontrolliert, also direkt mit dem Hidden Service verbunden ist. Durch diesen Angriff lassen sich sowohl Hidden Services auf Clients, als auch solche, die auf Server-Knoten laufen, enttarnen.


Mit diesen Angriffen lassen sich Hidden Services binnen Stunden, oder sogar weniger Minuten enttarnen. Durch die Nutzung von Guard Nodes sinkt die Wahrscheinlichkeit...
keit eines erfolgreichen Angriffs drastisch. Werden allerdings genug solcher Knoten
deaktiviert, so kann der Hidden Service enttarnt werden. Hierbei kann der Angreifer
durch Schichtung von Guard Nodes gebremst werden. Erfolgt die Auswahl der
Guard Nodes - in Kombination mit deren Schichtung - nicht zufällig und basierend
auf bestehenden Vertrauensverhältnissen, sinkt die Erfolgswahrscheinlichkeit eines
Angriffs noch weiter und die Enttarnung kann eventuell vollständig verhindert wer-
den.

3.5 Deskriptorformate

Um die Hidden Services zu kontaktieren, werden spezielle Informationen benötigt. In
Tor sind diese als Hidden Service Deskriptor bekannt. Overlier und Syverson haben
sich in [16] mit deren Weiterentwicklung beschäftigt. Im Folgenden wird genauer auf
die von Tor verwendeten Deskriptoren eingegangen. Die von Overlier und Syverson
vorstellte Ticket-Variante ist in Abschnitt 3.2.3 vorgestellt worden und wird somit
nur kurz zusammengefasst.

3.5.1 Tor Hidden Service Deskriptor

Die von Tor verwendeten Hidden Service Deskriptoren [25] existieren in drei ver-
schiedenen Versionen. Version V0 und Version V2 werden hier vorgestellt. Da die
Version V1 nicht praktisch eingesetzt wurde, wird der Vollständigkeit halber auf die

Die erste Version V0 der Hidden Service Deskriptoren beinhaltet folgende Felder:

<table>
<thead>
<tr>
<th>Feld</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL</td>
<td>Schlüssellänge</td>
</tr>
<tr>
<td>PK</td>
<td>Hidden Service Public Key</td>
</tr>
<tr>
<td>TS</td>
<td>Timestamp</td>
</tr>
<tr>
<td>NI</td>
<td># Introduction Points</td>
</tr>
<tr>
<td>Ipt</td>
<td>Pointer zu IPo Liste</td>
</tr>
<tr>
<td>SIG</td>
<td>Signatur obiger Felder</td>
</tr>
</tbody>
</table>

Hierbei gehört der genutzte Schlüssel zu dem Hidden Service. Der Timestamp gibt
die Zeit der verstrichenen Sekunden seit 1970 an. Die Liste der Introduction Points

Die Version V2 wird ab der Tor Version 0.2.0.10-alpha unterstützt und besteht aus
folgenden Teilen: Der Rendezvous-service-descriptor besteht aus 160-Bit base32
codierten Zeichen und ändert sich periodisch. Client und Hidden Service selbst können
diese ID jeweils berechnen. Zur Bestimmung der ID wird folgendes ausgewertet.

descriptor-id = H(permanent-id | H(time-period | cookie | replica))
permanent-id = H(public-key)[:10]


### 3.5.2 Valet-Ticketsystem


### 3.6 Informationsverteilung

Im Folgenden werden zwei verschiedene Ansätze der Informationsverteilung für Anonymisierungsnetzwerke vorgestellt. Zum Einen die Verzeichnissever, die von Tor Eingesetzt werden um Deskriptoren zu speichern und abzurufen, und zum anderen

---

12\(H(\text{time-period} \mid \text{cookie} \mid \text{replica})\)

13\(\text{YYYY-MM-DD HH:MM:SS}\)
3.6 Informationsverteilung

Verteilte Hash-Tabellen (DHT)\(^{14}\).

3.6.1 Tor Verzeichnisserver

Die von Tor eingesetzten Verzeichnisserver stellen ein wichtiges Element des Netzwerkes dar. Sie werden von den Onion-Routern des Tor Netzwerks genutzt, um Deskriptoren zu veröffentlichen und abzurufen. Weiterhin sind sie auch in der Lage, Deskriptoren für Tor Hidden Services zu speichern und zu veröffentlichen. Es existieren nur wenige solcher Verzeichnisserver. Die Verzeichnisserver sammeln Informationen über die Onion-Router des Netzwerkes. In bestimmten Abständen generieren die Verzeichnisserver ein signiertes Dokument, welches die zulässigen OR enthält. Unter den Verzeichnisserver findet eine ständige Abstimmung statt, welche der ORs sich in diesem signierten Dokument befinden werden.

3.6.2 Verteilte Hash-Tabellen


Da jeder Teilnehmer nur für einen bestimmten Bereich des ID Raums zuständig ist, werden Suchanfragen auf eine bestimmte Weise durchgeführt. Wird eine Anfrage an einen Knoten gestellt und ist dieser nicht selbst für diesen Datensatz verantwortlich, so leitet er die Anfrage an den Knoten weiter, der am nächsten an diesem Wert liegt. Dieser nächste Knoten wird mit hoher Wahrscheinlichkeit entweder selbst für den Datensatz verantwortlich sein oder einen Knoten kennen der eventuell für den Datensatz verantwortlich ist. Ist auch dieser nächste Knoten nicht für den Datensatz zuständig, wiederholt die diese Prozedur solange bis ein entsprechender Knoten gefunden wurde. Hierbei kann es aufgrund der Konstruktion von DHTs nicht eintreten, dass kein Knoten für den Datensatz verantwortlich ist, außer der Datensatz selbst existiert nicht.

Der Vorteil einer DHT liegt darin, dass die Werte gleichmäßig auf alle Knoten verteilt werden, die Suchanfragen geringe Komplexität besitzen und diese Struktur auch für eine sehr große Teilnehmerzahl skalierbar ist. Ein weiterer Vorteil dieser Art der Informationsverteilung liegt darin, dass die Anzahl der Teilnehmer nur schwer zu bestimmen ist. Weiterhin können aufgrund der Größe des ID Raums nicht alle Datensätze effizient sequenziell gesucht werden.

Die von Shalon genutzte DHT basiert auf Kademlia\(^{21}\), welche in einer Implementierung von Mojito vorliegt. Die Suche nach Datensätzen verläuft hier in der Regel mittels einer zufälligen ID aus dem ID Raum. Wird die ID nicht gefunden, so wird hier die ID die als nächstes an der ursprünglichen Anfrage liegt zurückgeliefert. Auch

\(^{14}\)Distributed Hash Table
für das in Kapitel 4 vorgestellte Hidden Service Design wird diese DHT zur Speicherung der Hidden Service Deskriptoren genutzt. Die Suche nach einem Hidden Service Deskriptor läuft hierbei natürlich nicht zufällig ab, sondern basiert auf einer bestimmten ID. Hierzu sei auf die Kapitel 4.6 und 4.8 verwiesen.

In Shalon treten die Shalon Server Knoten der DHT als aktive Knoten bei. Die aktive Teilnahme schließt das Speichern von Datensätzen und das Weiterleiten von Anfragen mit ein. Client Knoten, wie auch die der Hidden Services, treten der DHT als sogenannte passive Knoten bei. Dies hat zur Folge, dass diese Knoten nicht durch eine Suche gefunden werden können und nicht für die Speicherung von Datensätzen genutzt werden können. Suchanfragen können diese Knoten aber dennoch stellen.

3.7 Alternative Hidden Service Konzepte


3.7.1 I2P - Invisible Internet Project


3.7.1.1 Funktionsweise


ein Padding der übergeordneten Schicht zu ermöglichen. Vorausgesetzt es existieren genügend Nachrichten, dann kann dadurch die auch Nachrichtenanalyse erheblich erschwert.


Im Gegensatz zu Tor, verfügt I2P nicht über eine große Menge von Exit-Nodes (hier: Outproxy), da das Verlassen des I2P Netzes nicht vorgesehen ist. Trotzdem ist es durch geeignete Konfiguration möglich, ein Exit-Node zu betreiben. Um einen Exit-Node nutzen zu können, muss dieser, wie auch in Tor, den Benutzern bekannt gemacht werden. Da kein zentraler Verzeichnisserver und es außerdem kein Mechanismus zu Verteilung dieser speziellen Informationen existiert, können die verfügbaren Outproxies in I2P nicht effizient bekannt gegeben werden. Weiterhin müssen die Adressen der Outproxies manuell in der Konfiguration eingetragen werden.

3.7.1.2 Eepsites

Neben verschiedenen I2P Anwendungen wie I2Psnark (BitTorrent), I2Phex (Filesharing), I2PMail (Anonyme eMail) und I2PTunnel, existiert außerdem die Möglichkeit, anonym einen Webserver zu betreiben. Da diese sogenannte Eepsite primär für das Bereitstellen von anonymen Webseiten vorgesehen sind, ist in der I2P Software ein Webserver integriert. Jedoch lassen sich auch beliebige andere Dienste mittels einer Eepsite anbieten, wie zum Beispiel einen Tunnel in das Tor Netzwerk oder ein anonymer IRC Server.


3.7.2 Freenet


3.7.2.1 Einfügen von Datensätzen

Die Datensätze sind unter bestimmten Schlüsseln abgespeichert. Diese Schlüssel werden durch die Berechnung der SHA-1 Hashfunktion bestimmt und haben damit eine Länge von 160-Bit. Wird nun eine Anfrage für einen bestimmten Schlüssel gestellt, so wird die Anfrage dynamisch, abhängig von dem Schlüssel, durch das Netzwerk geroutet. Hierbei entscheidet jeder Knoten auf der Route bei Erhalt der Anfrage wohin und ob diese weitergeleitet wird.


Um dieses Problem zu beheben existieren die SSK (signed-subspace keys). Diese Schlüsselart erlaubte es jedem Benutzer seinen eigenen, persönlichen Namensraum zu definieren. Um diesen Namensraum zu erzeugen, generiert der Benutzer ein zufälliges Schlüsselpaar. Um den Wert für den öffentlichen Namensraum zu bestimmen, werden unabhängig Hashwerte von dem öffentlichen Teil des Schlüssels und der Be-
3.8. DeleGate


Beim Einfügen eines neuen Datensatzes, wird dieser auf den Knoten gespeichert die weitere Datensätze mit ähnlichen Schlüsseln verwalten.

3.7.2.2 Abrufen von Datensätzen


3.8 DeleGate


Nach längerer Zusammenarbeit mit dem Entwickler der DeleGate Software, wurden zunächst eine Reihe von Erweiterungen der Software diskutiert und integriert. DeleGate sollte im Weitern genutzt werden, um entfernt einen Port zu öffnen und

Im Folgenden werden einige Protokolle diskutiert welche, als interessante Grundlage für die Entwicklung von Hidden Services hätten dienen können.

3.8.1 Virtual Socket Association Protokoll (VSAP)

Bei dem VSAP Protokoll handelt sich nicht um ein standardisiertes Protokoll im Sinne der IETF oder RFC Publikationen. Die Dokumentation und Beschreibung besteht nur aus den Kommentaren des Entwicklers im Quellcode. VSAP ist als Erweiterung des SOCKS Protokoll entwickelt worden, um mehr als nur eine Verbindung durch einen geöffneten Port entgegennehmen zu können. Im Gegensatz zu SOCKS handelt es sich nicht um ein Bytelevel Protokoll, sondern basiert auf der ASCII Kodierung.


3.8.1.1 Problematik

Der Hidden Service müsste hierbei dafür sorgen, dass nach Beendigung einer Verbindung eine neue Verbindung bereit steht, um die nächsten Clientanfragen entgegenzunehmen. Dies führt zu einem Overhead, da der Hidden Service hierbei damit beschäftigt ist, ständig neue Verbindung zu diesem Proxy Server aufzubauen. Um den Overhead des Tunnelaufbaus zu senken könnte die Session-Wiederverwendungsfunktion der SSL-Implementierung genutzt werden.

3.8.1.2 Vorteile

Im Gegensatz zu SOCKS ermöglicht es dieses Protokoll, einen bestimmten Port zu reservieren, der exklusiv für BIND Anfragen des Protokolls genutzt wird. Damit es möglich, nach Entgegennehmen einer Verbindung über den reservierten Port sofort eine weitere BIND Anfrage für diesen Port zu stellen und eine weitere Verbindung entgegenzunehmen.
3.8. DeleGate

3.8.1.3 Zusammenfassung


3.8.2 HTTP Protokoll Erweiterung


3.8.2.1 Problematik

Wie bei SOCKS und VSAP besteht auch hier das Problem, dass es nur möglich ist, eine Verbindung entgegenzunehmen. Möchte man weitere Verbindungen entgegennehmen, muss der Hidden Service dafür sorgen, dass eine weitere Verbindung zum Proxy Server erstellt wird und ein weiterer Port geöffnet wird.

3.8.2.2 Vorteile

Verglichen mit dem VSAP Protokoll besteht hier der Aufwand zum Entgegennehmen von eingehenden Verbindungen nur aus einem Befehl und nicht aus vier. DeleGate kann weiter als HTTP Proxy Server benutzt werden. Somit würde bei der Verwendung dieses Ansatzes auch auf ein standardisiertes Protokoll zurückgegriffen werden.

3.8.2.3 Zusammenfassung

Wie auch bei der Verwendung von VSAP müssen hier nur wenige Änderungen vorgenommen werden. Der Hidden Service muss auch hier dafür sorgen, dass nach Entgegennahme einer Verbindung ein neuer Tunnel erstellt wird, um die nächste Anfrage entgegenzunehmen. Würde statt dieses Verbindungs-Pools ein Multiplexingprotokoll auf der bestehenden Verbindung eingesetzt, so würde der Aufwand der ständigen Be reithaltung von Verbindungen entfallen.

3.8.3 Zusammenfassung

Da das in DeleGate vorhandene, experimentelle Multiplexingprotokoll weder dokumentiert, noch standardisiert ist, wurde die Implementierung eines anderen, standardisierten Multiplexingprotokolls vorgeschlagen. Das Protokoll WebMUX, auf welches
später noch eingegangen wird, sollte integriert werden. Entsprechende Bibliotheken sind öffentlich zugänglich und wurden von dem Entwickler in die Software aufgenommen.


3.9 SOCKS


- CONNECT
- UDP ASSOCIATE
- BIND


3.9.1 Problematik

3.9.2 Vorteile


3.9.3 Zusammenfassung


3.10 Reverse Proxy


3.10.1 Problematik

der Anforderung, dass der Contact Point nicht für seine Aufgabe zur Rechenschaft gezogen werden kann.

3.10.2 Vorteile


3.10.3 Zusammenfassung


3.11 Multiplexingprotokolle


Hier werden nun zwei Multiplexingprotokolle vorgestellt, die als Grundlage für die Shalon Hidden Services zu Anfang in Frage gekommen sind. Zum Einen das SockMUX Protokoll und zum anderen das Session Control Protokoll. Ersteres ist Bestandteil der DeleGate Software, die Anfangs als mögliche Grundlage evaluiert wurde. Dieses Protokoll wurde primär entwickelt, um DeleGate Instanzen miteinander zu verbinden und über diese Verbindung mehrere gleichzeitige Client-Sessions zu ermöglichen.


3.11.1 SockMUX

Das SockMux Protokoll wurde in der DeleGate Software integriert, um effizient zwischen DeleGate Proxy Servern zu kommunizieren. Nachdem zwei Endpunkte über
3.11. Multiplexingprotokolle

dieses Protokoll miteinander verbunden sind, können Verbindungen angenommen werden und über eine bestehende persistente Verbindung durch Multiplexing kommuniziert werden. Das Protokoll unterstützt sowohl uni- wie auch bidirektionale Multiplexingverbindungen und wird als generische TCP Weiterleitung betrieben. Hierbei werden also keine weiteren Operationen auf den Datenstreams unternommen.

Dieses Protokoll ist weder spezifiziert, dokumentiert, noch ist es standardisiert worden. Hiermit handelt es sich also um ein exotisches und experimentelles Protokoll. Aufgrund dieser Eigenschaften ist es für die hier vorliegenden Zwecke ungeeignet, denn der Reverse-Engineering Aufwand würde den erzielten Mehrwert klar übersteigen. Weiterhin existiert dieses Protokoll ausschließlich in der DeleGate Software.

3.11.2 Session Control Protokoll (SCP)


Im Folgenden Abschnitt werden der Protokollablauf und Nachrichtenspezifikation von SCP diskutiert.

3.11.2.1 Aufbau


<table>
<thead>
<tr>
<th>FLAGS</th>
<th>SFPR0000</th>
<th>Connection ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Length</td>
</tr>
</tbody>
</table>
Die SCP Pakete haben einen 64-Bit Header, gefolgt von 0 oder mehreren Bytes Daten. Das FLAGS Feld erlaubt jeweils das Setzen einer Option. Diese können SYN, FIN, PUSH oder RESET sein.

SYN öffnet eine neue Verbindung, FIN schließt eine bestehende Verbindung. PUSH markiert jeweils eine Grenze einer Nachricht der Anwendungsschichtdaten. RESET sorgt für den Abbruch einer Verbindung.


Zustände:

- Closed
- OpenWrite, CloseWrite
- OpenSynRead, CloseSynReset
- OpenReadWrite
- CloseWrite, CloseRead

Ein Verbindungszustand kann sich nach dem Empfangen eines Paketes mit SYN, FIN, oder RESET ändern. OpenSynRead bedeutet beispielsweise, dass ein Client eine Verbindung geöffnet hat, den Outputstream geschlossen hat und nun auf ein SYN Paket des Servers wartet. Im Zustand OpenResetRead hat ein Client eine Verbindung geöffnet und sofort wieder geschlossen und erwartet nun ein SYN Paket zum vollständigen Schließen der Verbindung.
3.11. Multiplexingprotokolle

### Listing 3.2: SCP Events

Die Ereignisse, die von den verschiedenen Zuständen akzeptiert werden, sind jeweils nach absteigender Priorität sortiert. Falls Client oder Server ein fehlerhaftes Paket erhalten, wird die Verbindung geschlossen.

#### 3.11.2.2 Zusammenfassung

Das Session Control Protocol in der vorliegenden Form ist ein simples Protokoll wodurch sich einige Nachteile ergeben. Die Autoren des WebMUX Protokolls \[12\], welches durch SCP maßgeblich beeinflusst wurde, haben eine Reihe von möglichen Nachteilen identifiziert.

1. SCP kann einen Deadlock erzeugen, falls nicht unendlich viel Speicher zur Verfügung steht.
2. Es besteht keine Möglichkeit, verschiedene Anwendungsprotokolle über eine SCP Verbindung zu multiplexen.
3. 8 Byte Overhead im SCP Paketheader, aufgrund des 64-Bit Alignments. 32-Bit Alignment ist ausreichend da 64-Bit nur sehr selten benötigt wird.
4. Ohne Flusskontrolle würde ein unendlich großer Puffer für Client benötigt werden.
5. SYN kostet eine zusätzliche RTT.

Diese Nachteile haben unter anderem zur Entwicklung des WebMUX Protokoll geführt. Weiterhin wurde das SCP weder erfolgreich praktisch implementiert, noch anderweitig evaluiert. Somit ist dieses Protokoll ungeeignet für die hier vorliegenden Zwecke.
KAPITEL 4

Das Shalon Hidden Service Protokoll

In diesem Kapitel wird detailliert auf das, in dieser Diplomarbeit entwickelte, Hidden Service Design eingegangen. Die relevanten Implementierungsmerkmale, die verwendeten Protokolle, sowie die Gesamtfunktionsweise werden erläutert. Zu Beginn wird der aktuelle Stand des Shalon Anonymisierungsdienstes [5], der als Grundlage dient, vorgestellt.

4.1 Shalon - Aktueller Stand


1Distributed Hash Table
4.2 Designübersicht


4.2. Designübersicht

Abbildung 4.2: Shalon Hidden Services Übersicht

der Entwicklung des hier vorgestellten Konzeptes auch die Verwendung von standardisierten Protokollen im Vordergrund steht, wurde die HTTP Implementierung um einen entsprechenden Befehl erweitert. Dieser erlaubt es, unter Angabe eines gewünschten Ports, einen solchen Server Prozess innerhalb des Proxy Servers zu starten.


4.3 Erweiterte Proxy Server Deskriptoren


4.4 Hidden Service Betriebsmodi

Das für Shalon entwickelte Hidden Service Protokoll biete eine Vielzahl von verschiedenen Möglichkeiten als Client den Hidden Service zu kontaktieren. Aber auch der Hidden Service als solcher hat bei der Initialisierung die Möglichkeit, zwischen zwei verschiedenen Varianten zu wählen.

(a) TLS-Gateway Modus

(b) Open/Public Modus

4.4.1 TLS-Gateway Modus

In der ersten Variante (a) startet der Hidden Service automatisch lokal einen TLS-Gateway Prozess, welcher Clientverbindungen über die Contact Point Verbindungen entgegen nimmt und diese an den eigentlichen lokalen Hidden Service weiterleitet. Das TLS-Gateway hat ein mit dem privaten Schlüssel des Hidden Services signiertes Zertifikat installiert, welches für die Authentifizierung gegenüber dem Client genutzt wird. Weil der Client über den Hidden Service Deskriptor im Besitz des öffentlichen
4.4. Hidden Service Betriebsmodi

Listing 4.1: Unsignierter erweiterter Proxy Server Deskriptor

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<proxy version="0.1">
    <contact>
        <name>Name of server owner</name>
        <email>valid mail address</email>
        <phone>if available</phone>
        <abuse>e-mail</abuse>
    </contact>

    <server>
        <ports>
            <port type="plain">3128</port>
            <port type="ssl">3129</port>
        </ports>
        <address version="4">IPv4 Address</address>

        <exitpolicy>
            <get>
                <port>
                    <min>0</min>
                    <max>1024</max>
                </port>
            </get>
        </exitpolicy>
    </server>

    <rendezvous>
        <support>proxy supports rendezvous (true/false)</support>
        <secure>allow secure connections (true/false)</secure>
    </rendezvous>
</proxy>
```
Abbildung 4.3: TLS-Gateway Modus

Schlüssels des Hidden Services ist, kann die Echtheit des Zertifikates beim Verbindungsauflauf über die Signatur geprüft werden. Diese Option stellt End-to-End Sicherheit zwischen Client und Hidden Service her. Der Client kann über den Hidden Service Deskriptor erfahren, ob die \textit{end\_to\_end\_security} Option gesetzt ist.

Die zwei verschiedenen Modi führen zu Implikationen für die Contact Points. Werden von den Contact Points auch gesicherte Verbindungen geduldet, so sind diese nicht zwangsläufig in der Lage, den Inhalt der Kommunikation einzusehen. Die jeweilige Regelung für eingehende Verbindungen am Contact Point wird von diesen im Proxy Server Deskriptor, unter der \textit{rendezvous} Option kenntlich gemacht. Wie in Abschnitt 4.3 erläutert, gibt der Contact Point in seinem Deskriptor an, welche Art von Verbindungen an einen Hidden Service weitergeleitet werden können. Wird der Hidden Service im TLS-Gateway Modus betrieben, kommen nur Contact Points in Frage, die die \textit{secure} Option in ihrem Deskriptor auf \textit{true} gesetzt haben. Contact Points, die diese Option auf \textit{false} gesetzt haben, geben damit an, dass sie nur unverschlüsselte Verbindungen erlauben.

4.4.2 Open/Public Modus

Die zweite Variante (\textit{b}) kommt ohne TLS-Gateway aus, weil hier kein Wert auf End-to-End Sicherheit gelegt wird. Clients verbinden sich direkt über den Contact Point und müssen keine zusätzliche TLS-Verbindung mit dem Hidden Service initiieren. Die Kommunikation vom Client bis zum Contact Point ist somit nicht gegen das Ausspähen der transferierten Daten geschützt. Der Contact Point kann alle Daten, die zwischen Client und Hidden Service kommuniziert werden, lesen.
4.5 Hidden Service Konfiguration

Der Anbieter eines Hidden Service verfügt über eine Reihe von Optionen. Neben den im Abschnitt 4.4 beschriebenen Betriebsmodi können die folgenden Optionen vom Benutzer beeinflusst werden:


Neben diesen Optionen lässt sich auch die Anzahl der verwendeten Proxy Server auf der Kette zwischen Hidden Service und Contact Point kontrollieren. Um ausreichende Anonymität zu gewährleisten, werden hier inklusive des Contact Points, mindestens drei empfohlen. Die Anzahl der verwendeten Proxy Server auf dieser Kette kann sich direkt auf wahrgenommene Reaktionszeit und Durchsatz auswirken. Auch hierzu sei auf das Evaluationkapitel 5 verwiesen.

4.6 Hidden Service Deskriptoren


- **Contact Points:** Die Shalon-Server, die genutzt werden können um den Hidden Service zu kontaktieren. Ein Hidden Service kann über eine Reihe von Contact Points verfügen.

- **Server:** Die Repräsentierung der einzelnen Contact Points. Diese enthalten Informationen über IPv4 Adresse, sowie den genutzten Port des Hidden Service Protokolls.

- **End-to-End Sicherheitsoption:** Hiermit wird angegeben, ob nur zusätzlich gesicherte Clientverbindungen über das TLS-Gateway des Hidden Services angenommen werden.

Diese Informationen werden abschließend mit dem privaten Schlüssel des Hidden
4.6. Hidden Service Deskriptoren


```
<xml version="1.0" encoding="UTF-8" standalone="no"?/>
<hidden_service>
  <contact_points>
    <server>
      <ip>137.226.143.231</ip>
      <port>1234</port>
    </server>
    <server>
      <ip>137.226.143.232</ip>
      <port>1234</port>
    </server>
    <server>
      <ip>137.226.143.233</ip>
      <port>1234</port>
    </server>
    <server>
      <ip>137.226.143.234</ip>
      <port>1234</port>
    </server>
  </contact_points>
</hidden_service>

Listing 4.2: Unsignierter Shalon Hidden Service Deskriptor

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Descriptor version="0.1">
    A2:92:83:16:36:FF:EF:96" lastModified="1235491579057">
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
          w3.org/2001/10/xml-exc-c14n#WithComments"/>
        <SignatureMethod Algorithm="http://www.w3.or
          g/2000/09/xmldsig#rsa-sha1"/>
        <Transforms>
          <Transform Algorithm="http://www.w3.org
            /2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org
          /2000/09/xmldsig#sha1"/>
        <DigestValue>hwdYH81b4w/kYsXFM+59UUw
```

Listing 4.3: Signierter Shalon Hidden Service Deskriptor
Kapitel 4. Das Shalon Hidden Service Protokoll

48 Kapitel 4. Das Shalon Hidden Service Protokoll

Listing 4.3: Signierter Shalon Hidden Service Deskriptor
4.7 Hidden Service Kontaktmöglichkeiten

Die im Folgenden beschriebenen Kontaktmöglichkeiten für Clients setzten voraus, dass die Clients im Besitz des Hidden Service Deskriptors sind. Das Abrufen des Hidden Service Deskriptors wird in Abschnitt 4.8 beschrieben.


4.7.1 Shalon Clients


(1) Wird der Hidden Service im TLS-Gateway Modus betrieben, so erstellt der Shalon Client einen 3-er Circuit an dessen Ende sich einer der Contact Points des Hidden Service befindet. Es werden hierbei TLS-Sitzungen mit dem ersten und dem zweiten Proxy des Circuits ausgesprochen. Da der dritte Proxy des Circuits der Contact Point ist, wird außerdem die Initiierung der dritten TLS-Sitzung, transparent an das TLS-Gateway des Hidden Service weitergeleitet.

(2) Im Open/Public Betriebsmodus des Hidden Service erstellt der Shalon Client einen 3-er Circuit mit zufälligen Proxy Servern. Über diesen Circuit wird einer der Contact Points des Hidden Services als reguläres Ziel kontaktiert.

Im Open/Public Betriebsmodus ist die Kommunikation zwischen Exit-Node des Circuits zum Contact Point ungesichert und für Dritte lesbar. Im TLS-Gateway Betriebsmodus wird ebendiese Kommunikation durch die Ende-zu-Ende TLS-Verschlüsselung zwischen dem Shalon Client und dem TLS-Gateway des Hidden Service gesichert.

Die Kontaktmöglichkeiten (1) und (2) setzen die Kenntnis der Hidden Service ID und des Hidden Service Deskriptors voraus. Ist der Hidden Service Deskriptor unbekannt aber einzelne Contact Points sind etwa öffentlich bekannt, so können diese als reguläres Ziel von dem Shalon Client kontaktiert werden.

---

2Es wäre auch möglich, einen 3-er Circuit zum Contact Point zu erstellen und anschließend an diesem eine lokale Verbindung zum Hidden Service Port zu starten. Diese Indirektion setzt allerdings voraus, dass der Client den Proxy Port des Contact Points kennt. Eventuell könnte diese Information sinnvoll in die Hidden Service Deskriptoren integriert werden.
4.7.2 Non-Shalon Clients

Um die Kontaktmöglichkeiten von Non-Shalon Clients zu erläutern, stellt der Hidden Service hier beispielhaft einen Web-Server dar. Es sei weiterhin angenommen, dass die Non-Shalon Clients die IP-Adressen und die Ports, unter denen die Contact Points des Hidden Services erreichbar sind, etwa über die DHT Web-Schnittstelle (siehe Abbildung 4.5) oder anderweitig erfahren haben. Außerdem benötigen die Clients Kenntnis über den Betriebsmodus des Hidden Services. Den Clients die nicht das Shalon Anonymisierungsnetzwerk nutzen, bieten sich folgende Möglichkeiten einen Hidden Service zu kontaktieren:

1. Im TLS-Gateway Betriebsmodus des Hidden Service muss das Clientprogramm, im Beispiel der Web-Browser des Clients, die TLS-Verbindung mit dem TLS-Gateway des Hidden Services initiieren.

2. Wenn der Hidden Service im Open/Public Modus betrieben wird, so wird keine weitere Anforderung an das Clientprogramm des Non-Shalon Clients gestellt. Die Verbindungserstellung erfolgt direkt über den Contact Point ohne weitere Sicherheitsprotokolle.

Hierbei sei angemerkt, dass sich die Nutzung von Hidden Services nicht ausschließlich auf Web-Server beschränkt. Stellt der Hidden Service beispielsweise einen IRC Server bereit, so ist es für die Clients erforderlich, die IP-Adresse und den Port des genutzten Contact Points, direkt als IRC Serveradresse anzugeben.

4.8 Deskriptorabruf

Um in der Lage zu sein, einen Hidden Service zu kontaktieren, müssen die nötigen Kontaktinformationen in Form eines Deskriptors abgerufen werden. Für den Abruf
4.8. Deskriptorabruf

eines Deskriptors wird dessen eindeutige ID benötigt. Die ID eines Hidden Service Deskriptors entspricht dem 160-Bit SHA-1 Hashwert des öffentlichen Schlüssels des Hidden Service und ist im Deskriptor, im Element SignedData, als Wert id enthalten. IDs von Hidden Services könnten zum Beispiel über eine Webseite, ähnlich wie Tor Hidden Wiki


Ist der Client nun im Besitz einer ID von einem Hidden Service, so bieten sich ihm mehrere Möglichkeiten den Deskriptor abzurufen und den Hidden Service zu kontaktieren. Im Folgenden wird auf die Methode des Abrufs eingegangen.

4.8.1 Shaloninterner Deskriptorabruf

Handelt es sich bei dem Client um einen Shalon Client, so verfügt der lokale Shalon Proxy über die Funktionalität, Hidden Service URLs zu filtern, den Hidden Service Deskriptor abzurufen und die Verbindung, basierend auf den Deskriptorinformationen, herzustellen.

Die URL

4d59eacdc38a5464abb2cfaffa292831636ffe96.shalon

(4.1)


Falls der Deskriptor gefunden wurde, werden Verbindungen zu zufälligen Contact Points erstellt. Basierend auf den Informationen über den Betriebsmodus des Hidden Service, weisen diese Verbindungen bestimmte Sicherheitsmerkmale auf. Im TLS-Gateway Modus, wie in Abschnitt 4.3 erläutert, erstellt der Client eine Verbindung, die über End-to-End Verschlüsselung zwischen Client und Hidden Service verfügt. In dem in Abschnitt 4.4.2 erläuterten Open/-Public Modus stellt der Contact Point ein reguläres Ziel einer Anfrage dar. Das heißt, dass der Client eine Proxy-Kette mit einer bestimmten Länge erstellt und anschließend über diese eine Verbindungsanfrage an den Contact Point stellt.

4.8.2 Webschnittstelle

Da das entwickelte Hidden Service Design, neben Clients aus dem Shalon Netz, auch Clients von außerhalb dieses Netzes unterstützt, wurde eine Webschnittstelle zum


4Zwecks Browserkompatibilität frei von Doppelpunkten

Ein weiteres Entwicklungsziel war die Migration des Deskriptorabrufs auf den vom Shalon Server benutzten Proxy Port. Es kann dadurch erreicht werden, dass die gesamte Kommunikation des Shalon Netzwerkes über einen Port stattfindet. Der Deskriptorabruf kann somit In-Band geschehen und ist dadurch von regulärer Shalon Kommunikation nicht direkt zu unterscheiden. Ebenfalls bietet dies den Vorteil, dass der Deskriptorabruf anonym und verschlüsselt erfolgen kann. Um dies technisch zu realisieren wurde der nun interne Shalon Proxy Server um einen Filtersatz erweitert. Wird ein bestimmtes URL-Muster einer GET Anfrage gefiltert, so wird eine Suche, im internen DHT initiiert. Unter verschiedenen Anfrage-URLs liefert die Webschnittstelle verschiedene formatierte Deskriptoren zurück. Folgende Formate werden unterstützt:

(a) Unformatierte Proxy Server Deskriptoren

(b) Formatierte Proxy Server Deskriptoren

(c) Unformatierte Hidden Service Deskriptoren

(d) Formatierte Hidden Service Deskriptoren

![Shalon DHT Query Web-Interface](image)

Abbildung 4.6: Shalon DHT Query Web Interface
4.8. Deskriptorabruf


Unter Abbildung 4.8 ist ersichtlich, über welche URL ein formatierter Proxy Server Deskriptor abgerufen werden kann. Das Feld ID repräsentiert hier die ID des Deskriptors, der gesucht wurde. Im Fall von Proxy Deskriptoren wird, bei nicht gefundenen ID, der Deskriptor mit der nächsten ID zurückgeliefert.

Die formatierte Rückgabe einer Suche nach einem Hidden Service Deskriptor beinhaltet, wie anfangs erläutert, Links zu den jeweiligen Contact Points des Hidden Service.

4.9 Contact Point Funktionsweise

In diesem Abschnitt wird auf die zentrale Komponente zwischen Hidden Service und Client eingegangen. Die sogenannten Contact Points dienen hier der Verbindungsvermittlung von Client zu Hidden Service. Im Folgenden wird sowohl eine Funktionsübersicht gegeben und außerdem wird auf die technische Realisierung eingegangen.

4.9.1 Übersicht


4.9.2 Implementierung

Für die technische Realisierung wurde der HTTP Proxy Server Muffin als Grundlage genutzt. Im ersten Schritt wurde dafür gesorgt, dass sich zu diesem Proxy Server ei-
4.9. Contact Point Funktionsweise

Abbildung 4.9: Contact Point Architektur


\[ \text{BIND} \ 1234 \ \text{HTTP/1.1} \] \hspace{1cm} (4.3a)

\[ \text{HTTP/1.1} \ 200 \ OK \ \text{Port :} \ 1234 \] \hspace{1cm} (4.3b)

Nachdem der BIND Befehl erfolgreich verarbeitet wurde, sendet der Contact Point dem Hidden Service eine Nachricht, in welcher der Hidden Service über die Durchführung des Befehls informiert wird (siehe 4.3b). Bei erfolgreichem Starten des Server Prozesses und der Multiplexingverbindung, sendet der Contact Point eine Antwort, die die reservierte Portnummer beinhaltet. Diese Portnummer wird später, zusammen mit der IP-Adresse des Contact Points, einen Teil des Hidden Service Deskriptors darstellen. Ist der Port mit der angeforderten Portnummer auf dem Contact Point belegt, wählt der Contact Point zufällig einen möglichen Port aus.

Nach erfolgreicher Verbindungsersstellung zwischen Hidden Service und Contact Point sind diese nun über einen anonymisierten Tunnel verbunden. Auf diese Verbindung wird anschließend das Multiplexingprotokoll WebMUX aufgesetzt, um mehrere Clientanfragen über eine einzige Verbindung leiten zu können.

4.10 Quality of Service


Weiterhin werden verbundene Clients durch Neuverbinden schneller wieder mit dem Hidden Service verbunden als bei einer Änderung der Contact Points. Um die Ausfälle in Relation zur Dauer der Benutzung zu setzen, wird die sogenannte Death-Ratio (siehe Gleichung 4.4) definiert. Übersteigt die Death-Ratio einen bestimmten Schwellwert, so wird der Contact Point aufgegeben und eine neuer Contact Point bereitgestellt. Dies führt dazu, dass der Hidden Service Deskriptor neugeneriert und veröffentlicht werden muss. Hierbei gibt \#deaths die Anzahl der Verbindungsausfälle zu dem betrachteten Contact Point an. Die Größen $t_{\text{now}}$ und $t_{\text{init}}$ sind werden jeweils in Millisekunden betrachtet. $t_{\text{now}}$ gibt hierbei die aktuelle Zeit an und $t_{\text{init}}$ den Zeitpunkt der Initialisierung der Contact Point Verbindung.

$$\frac{\#\text{deaths}}{t_{\text{now}} - t_{\text{init}}} \quad (4.4)$$

4.11 WebMUX


Die Wahl des WebMUX Protokolls wurde maßgeblich durch die gute Dokumenta-

4.11.1 Ziele


Das WebMUX Protokoll versucht, diese Probleme zu lösen. Es erlaubt die Benutzung der Multiplexingverbindung ohne direkte Aushandlung mit dem Server, Dead-Lock freie Kommunikation basierend auf einem Credit-System und ist nicht auf ein bestimmtes Anwendungsprotokoll beschränkt. Weiterhin können sowohl Server als auch Client Multiplexing Sitzungen initiieren und die Anordnung der Daten bleibt ebenfalls vollständig erhalten. Neben diesen Zielen sollte sich das Protokoll effizient verhalten und ein möglichst einfaches Design aufweisen.

4.11.2 Protokollverhalten

Da es sich um ein Multiplexingprotokoll handelt, werden die Daten in Fragmenten übertragen. WebMUX benutzt standardmäßig eine Fragmentgröße von 512 Bytes. Über diese Größe wird direkt Einfluss auf die Latenz genommen, die durch das zusätzliche Protokoll entstehen kann. Weiterhin kann sich diese Größe bei verschiedenen Clients unterscheiden, da nicht immer genügend Speicher zur Verfügung steht, oder eine geringere Latenz erwünscht ist. Da WebMUX auf einer TCP-Verbindung

4.11.3 Packetaufbau


- MUX_CONTROL 0x00080000
- MUX_SYN 0x00400000
- MUX_FIN 0x00200000
- MUX_RST 0x00100000
- MUX_PUSH 0x00080000
- MUX_SESSION 0xFF000000
- MUX_LONG_LENGTH 0xFF040000
- MUX_LENGTH 0x0003FFFF

```c
typedef unsigned int flagbit;

struct w3mux_hdr {
    union {
        struct {
            unsigned int session_id : 8;
            flagbit control : 1;
            flagbit syn : 1;
            flagbit fin : 1;
            flagbit rst : 1;
            flagbit push : 1;
            flagbit long_length : 1;
            unsigned int fragment_size : 18;
            int long_fragment_size : 32;
            /* only present if long_length is set */
        } data_hdr;
        struct {
            unsigned int session_id : 8;
            flagbit control : 1;
            unsigned int control_code : 4;
            flagbit long_length : 1;
            unsigned int fragment_size : 18;
        } control_hdr;
    }
};
```
4.11. Verbindungsmanagement


WebMUX setzt ein Creditsystem ein, um das Verhungern von Sessions zu verhin- dern. Der Empfänger sendet eine bestimmte Menge Credits, um zu zeigen, dass er bereit ist, diese Menge Daten zu empfangen. Zusätzlich wird nun vom Sender nur soviel gesendet, wie der Empfänger bereit ist zu empfangen. Damit sollen etwaige Deadlocks verhindert werden. Zu Anfang verfügt jede Session über einen Credit von 16 KB. Falls mehr Daten vorliegen, kann die Session über die AddCredit Kontrollnachricht, mehr Credits hinzufügen.

4.11.5 Kontrollnachrichten

Im WebMUX Protokoll existieren eine Reihe von Kontrollnachrichten. Diese Nach- richten können zu jeder Zeit gesendet werden, auch auf einer Session, die noch nicht existiert.

- **InterAtom** wird derzeit nur für Session IDs genutzt. Atoms stellen hierbei eine Kurzform von Strings dar.

- **DefineEndpoint** kann genutzt werden, um einen Endpoint zu definieren und bekannt zu machen.
• SetMSS erlaubt es, die Fragmentgröße unter die verbleibenden Creditanzahl zu senken.
• AddCredit erlaubt es dem Empfänger, die Credits einer Session anzupassen.
• SetDefaultCredit lässt den Empfänger die Fragmentgröße auf den default Wert zurücksetzen.
• NoOp führt zu keinem Ereignis. Alle damit gesendeten Daten werden ignoriert.

4.11.6 Zusammenfassung


Für den Kontext der Shalon Hidden Services ist dieses Protokoll gut geeignet, da es ursprünglich für HTTP entwickelt wurde. Hidden Services sind nicht exklusiv für HTTP entwickelt worden, aber dennoch ist dies das Hauptanwendungsgebiet. Neben dem erfolgreichen Einsatz zwischen Hidden Service und Contact Point bietet WebMUX eine interessante Grundlage für anonymisierte Tunnel. Im nächsten Abschnitt wird daher kurz auf eine experimentelle Implementierung eines solchen Verbindungstyps eingegangen.

4.12 Multiplexing-Circuits


Der Benutzer kann über einen alternativen Port entscheiden, ob er eine Multiplexingverbindung nutzen möchte, oder nicht. Über diesen Port werden Shalon-Tunnel bereitgestellt, die über eine Multiplexingverbindung verfügen. Statt immer neue Verbindungen zu erstellen, werden eingehende Anfragen auf eine Multiplexingsession aufgesetzt. Der Client verfügt über einen Pool von bereitstehenden Multiplexingverbindungen, die direkt genutzt werden können. Stehen anfangs keine solchen Verbindungen bereit, wird eine neue erstellt und in den Pool eingefügt. Außerdem
verfügen diese Verbindungen über einen Timeout. Dies ist nötig, damit ein Client nicht zu lange den gleichen Circuit benutzt und eventuell Informationen darüber ableitbar sind.


4.13 Protokollablauf

In diesem Abschnitt wird zusammenfassend der Protokollablauf der Shalon Hidden Services erläutert und durch zwei abschließende Beispiele für Shalon und Non-Shalon Hidden Service Clients unterstützt.

4.13.1 Hidden Service Setup


### 4.13.2 Shalon Client Kontakt


Wird der Hidden Service im Open/Public Modus betrieben, wird ein regulärer Circuit aufgebaut und der Contact Point über diesen angesprochen. Non-Shalon Clients müssen den Deskriptor entweder über die DHT Webschnittstelle anfragen, oder die Contact Points auf einem anderen Weg "kennenlernen".
4.14 Zusammenfassung

In diesem Kapitel wurde das entwickelte Hidden Service Design detailliert vorgestellt. Zusammenfassend werden nun die Komponenten aufgelistet.

- Die erweiterten Proxy Deskriptoren dienen dem Zweck, die Contact Point Fähigkeit kenntlich zu machen. Weiterhin werden hier Beschränkungen auf die Art der Verbindungen über den Contact Point gekennzeichnet.


- Der Hidden Service verfügt über eine Reihe von Konfigurationsoptionen. Die Anzahl der benutzten Contact Points, der Betriebsmodus und die Tunnellänge zum Contact Point können kontrolliert werden.

- Es existieren die Betriebsmodi TLS-Gateway und Open/Public. Diese unterscheiden sich durch die gebotene Sicherheit und Performanz. Weiterhin haben sie unter Umständen juristische Implikationen für die Betreiber der Contact Points.

- Rudimentäre Quality-of-Service Unterstützung wurde hinsichtlich der Contact Points über die Death-Ratio implementiert.


Als zentrale Komponente sind die Contact Points im vorgestellten Konzept zu verstehen. Der HTTP Proxy Server Muffin wurde hier so erweitert, dass über einen zusätzlichen Befehl eine transparente Weiterleitung über eine bestehende Verbindung erstellt werden kann. Hierüber sind die Clients direkt in der Lage, mit dem Hidden Service in Kontakt zu treten.

Um die Verbindung zwischen Hidden Service und Contact Point auch für mehrere Clients effizient nutzen zu können, wurde das Multiplexingprotokoll WebMUX eingesetzt. Dieses Protokoll nutzt die unterliegende TCP Verbindung und verfügt somit über ein zuverlässiges Transportprotokoll.

KAPITEL 5

Evaluation


Wenn nicht anders gekennzeichnet, dann handelt es sich bei der Shalon Client Software immer um die in JAVA implementierte Version.

5.1 Versuchsaufbauten

Um das Shalon Hidden Service Design möglichst umfassend und realitätsnah zu evaluieren, wurde zwischen zwei Versuchsaufbauten unterschieden. Zum Einen wurden verschiedene Messungen in einer kontrollierten Laborumgebung abgehalten und zum anderen wurde das PlanetLab Testbed genutzt.

5.1.1 Laborumgebung

Die Messungen in der Laborumgebung zeichnen sich dadurch aus, dass der Netzwerkaufbau und die Netzwerktopologie vollständig kontrolliert werden kann. Außerdem existieren hier fast ideale Netzwerkbedingungen. Mit idealen Netzwerkbedingun-
Kapitel 5. Evaluation


5.1.2 PlanetLab

Um eine verteilte Anwendung unter möglichst realitätsnahen Bedingungen zu evaluieren, bietet sich das PlanetLab Testbed \[3\] an. In den nächsten Abschnitten wird auf die PlanetLab Technologie und den für die Evaluation des hier vorgestellten Konzeptes genutzten Versuchsaufbau erläutert.

5.1.2.1 Technologie


\[1\]Stand 23.04.2009
ney stehen eine Reihe von Tools bereit, mit denen die Administration des Slices und den Knoten für den Benutzer etwas erleichtert wird.

5.1.2.2 Versuchsaufbau


Abbildung 5.1: Durchsatz

Abbildung 5.2: Thread Durchsatz Vergleich

5.2 Proxy-Performanzanalyse

Die folgenden Messungen wurden im Lehrstuhlnetzwerk des Lehrstuhls für Informatik IV durchgeführt, um verschiedene Proxy Server unter nahezu idealen Bedingungen hinsichtlich ihrer Performanz zu beurteilen.

In der Einarbeitungsphase stellte sich unter anderem die Frage, welche schon vorhandene Proxy-Software sich besonders für die Zwecke dieser Diplomarbeit eignet.

Im Folgenden sind Vergleichsmessungen angestellt worden im Hinblick auf maximalen Durchsatz, Durchsatz pro Thread und die Verbindungsaufbauzeit. Alle Messungen umfassen jeweils 20 Wiederholungen, um ausreichend genau Konfidenzintervalle (95%) der Ergebnisse zu bestimmen. Der Versuchsaufbau wurde wie folgt gestaltet. Auf vier der Computern wurde jeweils der entsprechenden Proxy Server gestartet. Drei dieser Computer verfügten über zwei CPUs (1GHz Pentium III (Coppermine)) und 2GB RAM. Ein weiterer Rechner verfügte neben einer CPU (Intel(R) Core(TM)2 CPU 6600, 2.40GHz) ebenfalls 2GB RAM. Der Messcomputer ist mit dem letztgenannten identisch. Alle diese Rechner befanden sich außerdem in einem lokalen Netzwerk und waren über Gigabit-Ethernet miteinander verbunden.

Abbildung 5.3: Thread Durchsatz

\[2\text{Virtual Socket Association Protocol}\]

\[3\text{Socket Multiplexer}\]
Um den maximalen Durchsatz, wie in Abbildung 5.1 dargestellt, einer Verbindung zu bestimmen, wurde über verschieden lange Circuits, jeweils ein Datenvolumen von 50 Megabyte transferiert.

Abbildung 5.3 zeigt den maximalen Durchsatz im Hinblick auf einzelne parallele Transfer-Threads. Die Ergebnisse geben unter anderem Aufschluss über das Thread-Scheduling des Proxy Servers.

Bei der Messung der Aufbauzeiten wurde neben RSA- und DH-Schlüsselaustausch, SSL Session Reuse, weiterhin auch zwischen einer JAVA und C Implementierung von Shalon unterschieden. Abbildung 5.4 zeigt die Ergebnisse der JAVA Version. Abbildung 5.5 die Ergebnisse der C Implementierung.

Abbildung 5.5: Circuit Aufbauzeit (Shalon in C)
5.3 DHT Suchoperationen

Da das hier vorgestellte Hidden Service Design eine DHT als Grundlage der Informationsverteilung nutzt, wurden verschiedenen Messungen mit dieser durchgeführt. Um eine ausreichend hohe Knotenanzahl in der DHT zu erreichen wurde PlanetLab genutzt.


Abbildung 5.6: DHT mit 300 Knoten

5.3.1 Variation der Netzgröße

Um das Verhalten der DHT in Abhängigkeit von der Anzahl der Knoten zu untersuchen, wurden Vergleichsmessungen mit 200 und mit 300 Knoten angestellt. In Abbildung 5.6 sind verschiedene Ergebnisse bei einer DHT Größe von 300 Knoten dargestellt und Abbildung 5.7 zeigt das Ergebnis der gleichen Messung mit 200 Knoten in der DHT.
Hierbei lässt sich die Zeit erkennen, die für die Suche nach der angegeben Anzahl von Deskriptoren benötigt wurde. Das Ziel war es hier, eine bestimmte Anzahl von unterschiedlichen Deskriptoren durch zufällige Suche zu finden. Je weiter man sich der Gesamtanzahl der Knoten in der DHT annähert, desto länger dauert die Suche nach der entsprechenden Anzahl unterschiedlicher Deskriptoren.


5.3. DHT Suchoperationen

Um zu bestimmen, wie sich die Suche nach allen Deskriptoren in der DHT verhält, wurde eine Messung mit zufälliger Suche mit einer Woche Laufzeit durchgeführt. Bei einer DHT Größe von etwa 300 Knoten war es nicht möglich, mehr als 235 unterschiedliche Deskriptoren zu finden. Die Grafiken in Abbildung 5.10 zeigen das Verhalten der Suche nach allen Deskriptoren in der DHT. Es konnten maximal 235 unterschiedliche Deskriptoren gefunden werden. Es dauerte durchschnittlich 3,32
Stunden um 235 unterschiedliche Deskriptoren mit zufälliger Suche zu finden und es wurden durchschnittlich 1173 Suchanfragen dafür benötigt.

Abbildung 5.10: Langlaufende Deskriptorsuche

5.4 Labor Messungen


5.4.1 Hidden Service Clients

Die Zeit, die benötigt wird, um einen Hidden Service zu kontaktieren, wurde jeweils für die verschiedenen Clientarten bestimmt. Neben den Non-Shalon Clients wurden auch die Shalon Clients untersucht. Non-Shalon Clients nutzen hierbei, im Gegensatz zu den Shalon-Clients, nicht die Shalon-Software, um die Verbindung zum Contact Point zu anonymisieren. Weiterhin wurde außerdem der Einfluss der Tunnellänge zwischen Hidden Service und Contact Point auf die Kontaktzeiten der Clients untersucht.

5.4.1.1 Non-Shalon Hidden Service Client

Die Non-Shalon Clients verbinden sich direkt mit dem Contact Point. Diese Verbindungen können sowohl über eine End-to-End Verschlüsselung zum Hidden Service verfügen, oder Plain Verbindungen sein. Die Kontaktzeit besteht hier aus dem Öffnen
5.4. Labor Messungen

Abbildung 5.11: Hidden Service Client Kontaktzeiten

Eines Sockets zum Contact Point. Im Fall der End-to-End Verschlüsselung beinhaltet dies außerdem die Aushandlung einer TLS-Sitzung mit dem Hidden Service. In Abbildung 5.11 sind die Ergebnisse abgebildet. Hierbei zeigt sich, dass der Einfluss der Tunnellänge zwischen Hidden Service und Contact Point nur geringen Einfluss auf die Kontaktzeit der Clients hat.

5.4.1.2 Shalon Hidden Service Client


Hier zeigt sich, dass die Tunnellänge zwischen Hidden Service und Client einen größeren Einfluss auf die Kontaktzeit hat. Auch wird hier der Kompromiss zwischen Plain Verbindung und End-to-End Verschlüsselung deutlich. Abbildung 5.12 zeigt eine direkte Gegenüberstellung der Clientarten. In Abbildung 5.13 sind außerdem die erreichten Kontaktzeiten innerhalb des Labornetzes, denen des PlanetLab Testbeds gegenüber gestellt.

5.4.2 Verbindungsdurchsatz

Um den erreichbaren Durchsatz einer Verbindung zwischen Client und Hidden Service zu bestimmen sind in der Messung jeweils zehn Megabyte Daten zwischen Hidden Service und Client übertragen worden. Hierbei ist insbesondere interessant, wie groß der Einfluss von WebMUX und die Verwendung von Shalon an sich ist. Um ausreichend genaue Durchschnittswerte zu erhalten, wurde jeder Durchgang 20-mal
Abbildung 5.12: Shalon Clientarten Kontaktzeiten

Abbildung 5.13: Client Kontaktzeiten (Lab vs PlanetLab)


Zusätzlich wurde das Verhalten der Verbindung bezüglich der gleichzeitig kommunizierenden Threads untersucht. Durch Abbildung 5.15 zeigt sich, dass auch mit dem WebMUX Protokoll eine faire Verteilung der Bandbreite unter den Threads erzielen
5.4. Labor Messungen

Abbildung 5.14: Durchsatz verschiedener Clientarten

lässt.

5.4.3 Round Trip Zeiten

Nicht nur der erreichbare Durchsatz einer Verbindung entscheidet über die wahrgenommene Geschwindigkeit, sondern auch die Round Trip Zeit eines Pakets auf dieser Strecke. Um die RTT zu bestimmen wurde ein Paket zum Hidden Service geschickt, welches dieser anschließend wieder zurückgeschickt. Abbildung 5.16 zeigt die RTT Ergebnisse im Labornetzwerk. Hierbei wurde die Länge des Tunnels zwischen Hidden Service und Contact Point variiert. Es zeigt sich, dass die Tunnellänge erwartungsgemäß größeren Einfluss auf die RTT Zeiten hat, als auf den Durchsatz. Mit steigender Tunnellänge steigt auch die RTT. Dies ist durch die steigende Anzahl an involvierten Knoten zu erklären.

Die Unterschiede zwischen End-to-End Verschlüsselung und Plain Verbindung lassen sich durch die unterschiedlichen Tunnellängen und durch die mehrfache Verschlüsselung zwischen Client und Contact Point erklären. Weiterhin bestehen die TLS-Verbindungen im Plain Modus mit den drei Proxy Servern des Tunnels, woran eine Plain Verbindung anknüpft. Es ist unklar ob die direkte TLS-Verschlüsselung zum TLS-Gateway des Hidden Service, neben der Benutzung des WebMUX Protokolls, einen Einfluss auf dieses Verhalten hat.

5.4.4 Hidden Service Setup

Der Hidden Service selbst muss eine Reihe von Operationen ausführen, um verfügbar zu werden. Insbesondere müssen Verbindungen zu einer bestimmten Anzahl von Contact Points hergestellt werden. Diese Verbindungen bestehen, wie in dem
vorherigen Kapitel erklärt, aus einem anonymisierten Tunnel, an dessen Ende sich der Contact Point befindet. Der Hidden Service muss außerdem, nach erfolgreichem Verbindungsauflauf, dafür sorgen, dass das WebMUX Protokoll auf dieser Verbindung initiiert wird. In Abbildung 5.17 sind die Zeiten, die für eine Contact Point
5.5. PlanetLab Messungen

Verbindung benötigt werden, abgebildet. Im Labornetzwerk wurde diese Messung mit einem Conact-Point durchgeführt, aber die Tunnellänge zwischen Hidden Service und Contact Point variiert. Hier zeigt sich, dass mit steigender Tunnellänge auch die Zeit für den Verbindungsaufbau steigt. Den größten Anteil nimmt hierbei der Aufbau des anonymisierten Tunnels in Anspruch. Es muss mit jedem Proxy Server auf dem Pfad eine neue TLS-Sitzung gestartet werden. Die Einrichtung des WebMUX Protokolls auf dieser Verbindung wird nicht maßgeblich durch die Tunnellänge beeinflusst. Abbildung 5.18 zeigt die unterschiedlichen Setup Zeiten, die im Labornetzwerk und in PlanetLab erreicht werden. \( Mux \) gibt hier die Zeit an, die gebraucht wird, um auf der bestehenden Verbindung zwischen Hidden Service und Contact Point, das WebMUX Protokoll zu starten. Mit \( Tunnel \) ist die Zeit gemeint, die benötigt wird, um den anonymisierten Tunnel zwischen Hidden Service und Contact Point zu erstellen. \( Done \) gibt an nach welcher Zeit das vollständige Setup des Hidden Service abgeschlossen ist. Dies schließt Contact Point Verbindungen und Deskriptorveröffentlichung mit ein.

Abbildung 5.17: Contact Point Setup mit verschiedener Tunnellänge

5.5 PlanetLab Messungen

Da mit dem PlanetLab Testbed eine Umgebung mit weitaus mehr Rechnern zur Verfügung steht, können die Messungen aus dem Labornetzwerk weiter variiert werden. Hier kann zum Beispiel die Anzahl der benutzten Contact Points verändert werden, um die Performanz bei steigender Anzahl von Contact Points zu bestimmen. Auch hier wurden der erreichte Durchsatz, die Kontaktzeiten von Clients, die Setupdauer des Hidden Service, sowie die Round Trip Zeiten bestimmt. Da die PlanetLab Rechner gleichzeitig auch von anderen Personen genutzt werden, kann hier ein realismähnlicher Verhalten des entwickelten Protokolls beobachtet werden. Zu beachten ist hierbei, dass die Ergebnisse, abhängig von der derzeitigen Netzsituation, stark
5.5.1 Hidden Service Clients

Für die Kontaktzeit der Clients mit dem Hidden Service ist sowohl der Deskryptorabruf, als auch der Verbindungsaufbau zu einem der Contact Points des Hidden Service relevant.

Abbildung 5.20: Einfluss der Tunnellänge auf Kontaktzeiten der Hidden Service Clients (PlanetLab)

Auch hier wurde die Tunnellänge zwischen Hidden Service und Contact Points variiert. Die Auswirkungen auf die Kontaktzeiten sind in Abbildung 5.20 sowohl für Shalon Clients, als auch für Non-Shalon Clients zu sehen.

5.5.2 Verbindungsdurchsatz

Im Gegensatz zu der kontrollierten Umgebung des Labornetzwerks bietet PlanetLab ein realitätsnaheres Netzwerkverhalten. Ausgelastete Computer und Netzwerk Links sorgen dafür, dass der erreichte Durchsatz stark schwankt. Abbildung 5.21 zeigt den erreichten Durchsatz, abhängig von der Anzahl paralleler Client-Threads. Hier wurden drei Contact Points vom Hidden Service genutzt. Da die Auswahl des benutzten Contact Points vom Client zufällig geschieht, müssen nicht zwangsläuﬁg alle Threads den gleichen Contact Point benutzen. Trotz starken Schwankungen ist erkennbar, dass mit steigender Anzahl an parallelen Threads, der erreichbare Durchsatz sinkt.
5.5.3 Round Trip Zeiten


5.5.4 Hidden Service Setup

In der PlanetLab Umgebung stehen dem Hidden Service weit mehr mögliche Contact Points zur Verfügung. Deshalb wurden auch hier Messungen mit eins bis zehn Contact Points für den Hidden Service durchgeführt. Es wurde ebenfalls die Tunnelänge zwischen Hidden Service und Contact Point variiert. Dies hat direkten Einfluss auf die Zeit, die für die Konstruktion dieser Tunnel benötigt wird. Abbildung 5.23 enthält sowohl Ergebnisse bei einer Tunnelänge von vier Proxy Servern, als auch für die Länge drei inklusive des Contact Points. Ebenfalls abgebildet ist hier die Zeit, die benötigt wird, um die entsprechende Anzahl von Contact Points zu suchen. Es kann jedoch davon ausgegangen werden, dass diese Zeit bei dem Hidden Service Setup keine Rolle spielt. Proxy Server, die als mögliche Contact Points dienen können, sind als vorläufig zu betrachten, da der Hidden Service erst dann gestartet werden wird, wenn genügend Contact Points zur Verfügung stehen.

Um die Messungen mit Tor durchzuführen, wurde Puppettor verwendet. Bei Puppettor handelt es sich um ein in JAVA geschriebenes Messframework, welches über das Tor Control Protokoll mit Tor Prozessen interagiert. Loesing et al. haben Puppettor benutzt, um die in [14] und [20] veröffentlichten statistischen Auswertungen zu berechnen.

Zugänglich über GIT git://git.torproject.org/git/puppetor
gungen anzufertigen. Das Framework erleichtert es, schnell einen Messaufbau zu erstellen, zu automatisieren und an die eigenen Bedürfnisse anzupassen. Das Kernstück des Frameworks bildet die Schnittstelle zum Tor Control Protokoll. Hiermit ist das möglich, einen Tor Prozess zu beeinflussen und die jeweiligen Log-Ausgaben zu verarbeiten. Da Tor über ein sehr informatives und gut strukturiertes Logging verfügt, ist es möglich, auf die Log-Ausgaben mit einem eventbasierten System zu reagieren – zum Beispiel um die Zeit zwischen verschiedenen Events zu bestimmen. Weiterhin bietet Puppettor die Möglichkeit, lokal mehrere Tor Prozesse zu starten und on the fly zu konfigurieren. Diese Prozesse können sowohl ein lokales privates Netzwerk bilden, oder aber an ein bestehendes Tor-Netzwerk angeschlossen werden. Genau diese Funktionalität hat die hier durchgeführten Messungen erleichtert. Da sowohl im PlanetLab Testbed, als auch im Labornetzwerk, ein privates Tor-Netzwerk betrieben wurde, kann die konfigurierte Messung sehr einfach auf beide Netzwerke angewandt werden. Ebenfalls kann diese Messung ohne zusätzlichen Aufwand, auch mit dem öffentlichen Tor-Netzwerk durchgeführt werden.

5.6.1 Hidden Service Clients


Abbildung 5.24: Tor Hidden Service Client Kontaktzeiten
Um die Tor Hidden Service Clients mit den Shalon Hidden Service Clients vergleichen zu können, müssen einige Vereinbarungen getroffen werden, die den Vergleich ermöglichen. Da Shalon über keine Trennung von Introduction- und Rendezvous-Point verfügt, können die Ereignisse *Introduction Done* und *Rendezvous-Done* in dieser Form nicht abgebildet werden. Um in Shalon einen Hidden Service zu kontaktieren, bedarf es, neben dem Herunterladen des Deskriptors, nur der Kontaktierung des Contact Points. Sind beide Schritte erfolgreich durchgeführt worden, sind Client und Hidden Service verbunden. Abbildung 5.25 zeigt die entsprechenden Ereignisse, die einen Vergleich zulassen. Hierbei wurde, wie in Tor, von Shalon zwischen Hidden Service und Contact Point ein Circuit der Länge drei benutzt. Gleiches gilt für den Circuit zwischen Client und Contact Point. Der Contact Point nimmt hier also die Rolle des Rendezvous Points von Tor ein, denn an diesem werden die Circuits von Hidden Service und Client verbunden. Bei Shalon wurde der TLS-Gateway Modus benutzt, der es erlaubt zwischen Client und Hidden Service eine End-to-End gesicherte Verbindung mit TLS zu erstellen. Außerdem nutzt der Shalon Hidden Service drei Contact Points um das Verhalten von drei Introductions-Points nachzuprüfen.

Abbildung 5.25: Vergleich von Tor und Shalon Client Kontaktzeiten


Hier zeigt sich, dass die Kontaktzeiten der Shalon Hidden Services deutlich unter den Kontaktzeiten den Tor Hidden Services liegen. Der vollständige Verbindungsaufbau eines Tor Hidden Service Client zum Hidden Service involviert neben Circuits zu den Introduction Points des Hidden Service, außerdem den Aufbau eines Circuits zu einem Rendezvous Point. Die Shalon Hidden Service Clients benötigen hierbei
nur einen Circuit, welcher direkt zu dem Contact Point des Hidden Service führt. Dadurch werden insgesamt weniger Ressourcen benötig um den Hidden Service zu kontaktieren.

5.6.2 Verbindungsdurchsatz

Um den erreichten Durchsatz zu bestimmen, wurden den sowohl mit Shalon, als auch mit Tor jeweils zehn Megabyte Daten vom Hidden Service angefordert. Diese Messungen wurden ebenfalls mit eins bis zehn parallelen Threads durchgeführt, um das Verhalten bei gleichzeitigem Zugriff mehrerer Clients zu untersuchen. Sowohl mit Shalon als auch mit Tor sind Hidden Service und Client über eine Entfernung von sechs Hops miteinander verbunden. Abbildung 5.26 zeigt, neben dem erreichten Durchsatz im Labornetzwerk, auch den Durchsatz, der über PlanetLab erreicht wurde. Außerdem zeigt Abbildung 5.27 die Verteilung der Bandbreite auf die einzelnen parallelen Transfer-Threads. Es zeigt sich, dass Shalon den Tor Hidden Service unter Laborbedingungen überlegen ist. Hier existieren weniger Schwankungen. Im PlanetLab Testbed hingegen übersteigt der erreichte Durchsatz von Tor, den von Shalon. Dies kann auf die unterschiedlichen Umgebungszustände zurückzuführen sein.

Abbildung 5.26: Vergleich von Tor und Shalon Durchsatz

Abbildung 5.27: Bandbreitenverteilung auf parallele Threads

Ebenfalls interessant ist der Vergleich des erreichten Durchsatzes von Tor in den Umgebungen des Labornetzwerks, dem PlanetLab Testbed und dem öffentlichen
5.6. Tor Vergleichsmessungen

Tor-Netzwerk. Abbildung 5.28 zeigt den Durchsatz bei steigender Anzahl paralleler Threads in den verschiedenen Umgebungen.

Abbildung 5.28: Tor Durchsatz

5.6.3 Round Trip Zeiten

Auch im direkten Vergleich von den Tor Hidden Services und den Shalon Hidden Services sind die RTT Werte von großem Interesse. Es wurden im Folgenden RTT Messungen von Shalon und Tor im Labornetz, PlanetLab und mit dem öffentlichen Tor-Netzwerk unternommen. Wie in Abbildung 5.29 erkennbar ist, liegen die RTT Werte von den Shalon Hidden Service deutlich unter den RTT Werten der Tor Hidden Services. Diese gilt sowohl für die Umgebung des Labornetzwerkes, als auch für PlanetLab. Als weiterhin interessant erweist sich der Vergleich von den RTT Werten der Tor Hidden Services im Labornetzwerk, im PlanetLab Testbed und dem öffentlichen Tor-Netzwerk.

Abbildung 5.29: Vergleich von Tor und Shalon RTT

5.6.4 Hidden Service Setup

Damit ein Hidden Service Clients zur Verfügung stehen kann, müssen verschiedene Schritte unternommen werden. Sowohl in Shalon, als auch in Tor, muss der Hidden Service dafür sorgen, dass Introduction Points zur Verfügung stehen. Bei den Shalon
Hidden Services übernehmen diese zusätzlich die Rendezvous Points Funktionalität und verbinden die Circuits von Hidden Service und Clients. Nachdem die Introduction-/Contact Points vom Hidden Services für ihre Aufgabe konfiguriert worden sind, generiert der Hidden Service ein Deskriptor. Sowohl bei Shalon, als auch in Tor, enthalten die Deskriptoren eine Liste der genutzten Introduction-/Contact Points. Diese werden von den Clients genutzt, um den Hidden Service zu kontaktieren.
tion-/Contact Points vorzubereiten. Für diese Messungen wurde der Shalon Hidden Service mit drei Contact Points konfiguriert, die den drei Introduction Points von Tor nachempfunden sind. Mit dem Ereignis Descriptor Published ist die Zeit assoziiert, die benötigt wurde, um den Hidden Service vollständig zu konfigurieren und anschließend den Deskriptor zu publizieren. Diese Zeit ist maßgebend dafür, ab wann ein Client den Hidden Service kontaktieren kann. Abbildung 5.31 zeigt die unterschiedlichen Setup Zeiten, die der Hidden Service im Labornetzwerk, PlanetLab und dem öffentlichen Tor-Netzwerk benötigt.

5.7 WebMUX Evaluation


Um einen Vergleich zu der WebMUX Verbindung zu erhalten, wurden die Messungen zusätzlich mit einer direkten Verbindung zum Perl Server realisiert. Hierbei verfügt jeder Client über eine eigene TCP Verbindung.

5.7.1 Durchsatz

Abbildung 5.32 zeigt in der linken Teilabbildung den durchschnittlich erreichten Durchsatz zwischen den beiden Messcomputern. In der rechten Teilabbildung ist die Verteilung des Durchsatzes auf zwei verschiedene Clients zu sehen.

Für die WebMUX Verbindung zeigt Abbildung 5.33 den durchschnittlich erreichten Durchsatz, sowie den Durchsatz von zwei verschiedenen Clients.

Abbildung 5.33: Durchsatz der WebMUX Verbindung

Hier zeigt sich, dass der erreichte Durchsatz mit dem WebMUX Protokoll mit steigender Anzahl paralleler Clients, gleichmäßiger als bei der direkten Verbindung abfällt. Abbildung 5.34 zeigt den direkten Vergleich von direkter Verbindung und der WebMUX Verbindung.

Abbildung 5.34: Durchsatz: WebMUX vs. direkte Verbindung

5.7.2 Round Trip Zeiten

5.8 Zusammenfassung

In diesem Kapitel wurde das entwickelte Hidden Service Design eingehend auf seine Performanzegenschaften untersucht und mit den Tor Hidden Services verglichen. Es hat sich im Ganzen gezeigt, dass die geringe Komplexität des Protokolls einen Geschwindigkeitsvorteil erzielt. Beim direkten Vergleich des Hidden Service Setup, sind sowohl bei Shalon, als auch bei Tor, drei Introduction-/Contact Points verwendet worden. Auch ein Deskriptor muss erzeugt und veröffentlicht werden. Bei dieser Messgröße ist Shalon Tor überlegen. Weiterhin ist anzumerken, dass Tor bestehende Circuits erweitert oder direkt benutzt um einen Introduction Point zu erstellen. Shalon muss hierfür einen eigenen, neuen Circuit erstellen. Aber selbst wenn im Labornetzwerk einige Schwankungen beim Circuitaufbau von Shalon existieren, ist die Gesamtzeit, die gebraucht wird, bis der Deskriptor veröffentlicht wird, maßgeblich für die Setupdauer.


Neben der Kontaktzeit der Clients ist außerdem ein hoher Durchsatz und eine geringe RTT von Vorteil. Auch hier erzielt das vorgestellte Design bessere Ergebnisse als Tor. Sowohl im Labornetz als auch im PlanetLab Testbed betragen die RTT Werte
von Shalon nur einen Bruchteil von den RTT Werten von Tor. Selbst Shalon im PlanetLab, erzielt bessere RTT Werte, als Tor im Labornetzwerk. Neben besseren RTT Werten erreichen die Shalon Hidden Services außerdem einen höheren Durchsatz. Hierbei ist besonders die Faireness von WebMUX als Multiplexingprotokoll zu beobachten, da so eine faire Verteilung der Bandbreite auf parallele Clients möglich ist.

KAPITEL 6

Sicherheitsanalyse

Neben der Evaluation der Performanzeigenschaften des hier vorgestellten Designs, ist außerdem die Analyse der gebotenen Sicherheit wichtig. In diesem Kapitel werden die Sicherheitseigenschaften des vorliegenden Designs analysiert und mit denen der Tor Hidden Services verglichen.

6.1 Angreifermodell

Um die Sicherheit eines Konzeptes beurteilen zu können, muss ein Angreifermodell definiert werden. Durch ein Angreifermodell wird festgelegt, über welche Fähigkeiten und Kapazitäten ein Angreifer verfügt. Anhand der festgelegten Eigenschaften des Angreifers lassen sich dann Aussagen über die gebotene Sicherheit eines Ansatzes treffen.


Im Kontext von Kommunikationsnetzwerken können potentielle Angreifer lokale Netzwerkadministratoren, Internet Service Provider oder auch andere Teilnehmer eines Netzwerks sein. Hiermit lassen sich verschiedene Angreifer weiter in sogenannte lokale und globale Angreifer kategorisieren. Der globale Angreifer hat hierbei Zugriff auf alle Ressourcen und Kommunikationskanäle, der lokale Angreifer hingegen verfügt nur über eine Teilmenge der gesamten Ressourcen.
Da dieses Angreifermodell vornehmlich von theoretischer Natur ist, werden für den vorliegenden Fall die folgenden Einschränkungen gemacht:

- Der Angreifer kann keine der eingesetzten Verschlüsselungstechniken brechen.
- Der Angreifer kontrolliert nicht alle Knoten des Netzwerks.
- Der Angreifer ist kein globaler Angreifer.

### 6.2 Angriffsszenarien


#### 6.2.1 Deanonymisierung


Eine andere Möglichkeit den Hidden Service zu deanonymisieren, ist gegeben, wenn der Angreifer, neben dem ersten Knoten auf der Kette zwischen Hidden Service und Contact Point, zusätzlich den Contact Point kontrolliert. Der Angreifer kann leicht feststellen, ob sein Contact Point von einem Hidden Service genutzt wird, indem er kürzlich geöffnete Ports beobachtet oder seine Software entsprechend modifiziert.

---

[1] Auffüllen der der Daten auf eine bestimmte Packetgröße
Kombiniert der Angreifer die Informationen vom Anfang und vom Ende der Kette, so ist der Hidden Service enttarnt. Falls der Contact Point von mehr als einem Hidden Service genutzt wird, ist die Zuordnung von Port zu Hidden Service nicht mehr ohne Weiteres möglich. Möglicherweise könnte es durch die Verwendung von Cookies zur Authentifizierung, es einem korrupten Contact Point selbst erschwert werden, einen Hidden Service zu kontaktieren.


Um die Erkennung am Contact Point zu erschweren, könnte das Contact Point Protokoll so modifiziert werden, dass die Hidden Service Clients sich über den Standardport von Shalon mit dem Hidden Service verbinden. Dies würde allerdings bedeuten, dass der Verbindungsaufbau der Hidden Service Clients erweitert werden müsste, um dem Contact Point zu signalisieren, dass eine Verbindung mit dem Hidden Service erwünscht wird.


### 6.2.2 Denial of Service

lungen der Verbindung zwischen Hidden Service und Contact Point, steigt auch die Wahrscheinlichkeit, dass dieser Tunnel Knoten des Angreifers enthält.

Ohne Kenntnis über die Contact Points eines Hidden Service zu haben, ist ein Denial of Service Angriff gegen diese nicht durchführbar. Aufgrund der Konstruktion des Verzeichnisses von Shalon ist der Angreifer ohne die ID des Hidden Service nicht in der Lage, den Hidden Service Deskriptor zu finden. Ohne den Deskriptor besteht keine zuverlässige Möglichkeit, die Contact Points eines Hidden Service zu finden und anschließend anzugreifen. Kontrolliert der Angreifer eine ausreichend große Menge an Knoten, so ist es einfacher möglich, den Hidden Service zu enttarnen, statt die Contact Points des Hidden Service über Denial of Service zu attackieren.

6.3 Unterschiede zu Tor Hidden Services

Da das in dieser Diplomarbeit vorgestellte Konzept die Funktionalität von Tor Introduction- und Rendezvous Point ineinander vereint, können bestimmte Angriffe, welche aus [22] bekannt sind, nicht mehr durchgeführt werden. Neben der zusätzlichen Last für den Contact Point, können allerdings noch unklare Sicherheitsimplikationen entstehen. Im Gegensatz zu Tor hat der Client hier keinen Einfluss auf den Rendezvous Point, also den Knoten über den die Verbindung zum Hidden Service hergestellt wird. Die Contact Points von Shalon werden vom Hidden Service selbst gewählt und diese leisten gleichzeitig die Arbeit der aus Tor bekannten Rendezvous Points. Daraus folgt, dass insbesondere der Hidden Service vom Angreifer nicht zur Konstruktion neuer Rendezvous Points gezwungen werden kann. Somit entgeht der Hidden Service der Gefahr, einen vom Angreifer gewählten Rendezvous Point zu nutzen.

Da in Tor auch die Möglichkeit besteht, die im Verzeichnis registrierten Hidden Services aufzuzählen, kann hier vom Angreifer leicht überprüft werden, ob er Introduction Points eines Hidden Service kontrolliert. Im hier vorgestellten Konzept ist dies nicht möglich, da die Suche nach Deskriptoren auf der Generierung zufälliger Suchwerte basiert. Deshalb ist es nicht möglich, effizient das Verzeichnis nach Hidden Service Deskriptoren zu durchsuchen. In Kapitel 5.3 hat sich insbesondere gezeigt, dass die vollständige Aufzählung aller Deskriptoren sehr lange dauern wird, oder gar nicht möglich ist. Auch die Entwicklung eines in dieser Hinsicht robusteren Verzeichnisses scheint möglich.

Da sich die hier benutzten Hidden Service Deskriptoren nur mit der Kenntnis der ID effizient finden lassen, ist der Schutz der Contact Points vor Denial of Service Angriffen, als höher anzusehen als für die Introduction Points in Tor. In Tor können leicht alle Hidden Service Deskriptoren gefunden werden und anschließend mit einem Denial of Service Angriff die Introduction Points angeriffen werden.

Die Abwesenheit von gesonderten Introduction Points im vorgestellten Konzept, bringt jedoch auch Nachteile für den Hidden Service mit sich. Der Hidden Service hat derzeit keine Möglichkeit, Clientanfragen direkt über Shalon abzulehnen. Um dies für Shalon zu erreichen, könnte ebenfalls das Contact Point Protokoll erweitert werden. Der Nachteil einer solchen Erweiterung liegt dann in der schrittweisen Entfernung
von der Verwendung standardisierter Protokolle.

6.4 Zusammenfassung


Durch die Abwesenheit von Guard Nodes beim Verbindungsaufbau von Hidden Service zum Contact Point, steigt die Wahrscheinlichkeit, durch einen Angreifer de-anonymisiert zu werden. Da beim hier vorgestellten Konzept die Contact Points die Aufgaben von Introduction- und Rendezvous Point übernehmen, wird ein Angriffspunkt eliminiert. Der Hidden Service kann hier nicht gezwungen werden, neue Tunnel zu einem möglicherweise vom Angreifer kontrollierten Rendezvous Point aufzubauen. Durch die Einführung von Guard Nodes in Shalon würde sowohl der Schutz regulärer Clients, als auch der Schutz für Hidden Service Anbieter erhöht werden. Um Guard Nodes sinnvoll einsetzen zu können, muss ein geeignetes Vertrauensmodell für Betreiber von Shalon Server entwickelt werden. Weiterführende Konzepte, wie das in Kapitel 3.2 vorgestellten Konzept der Valet-Nodes, führt unweigerlich zu einer höheren Komplexität des Protokolls und fragwürdigem Mehrwert.

Der auf den ersten Blick geringere Schutz der Contact Points vor Denial of Service Angriffen relativiert sich, wenn der Aufwand einer zufälligen Suche nach Hidden Service Deskriptoren mit einbezogen wird. Neben der ineffizienten Suche ohne die ID des Hidden Service zu kennen, könnte zukünftig auch der Deskriptor selbst, besser vor unrechtmäßigen Zugriffen geschützt werden.
Kapitel 6. Sicherheitsanalyse

### 7.1 Erreichte Ziele


In den vorausgegangenen Kapiteln wurde ein Hidden Service Design vorgestellt, welches unter Verwendung von HTTP Proxy Servern ähnliche Funktionalitäten wie das Konzept der Tor Hidden Service bietet. Die Shalon Software wurde weiterentwickelt, so dass ein eigenständiger Proxy Server nicht mehr zwingend erforderlich ist. Der so integrierte Proxy Server bietet zudem die Möglichkeit, entfernt einen Server Prozess zu starten, welcher eingehende Clientverbindungen über eine bestehende Verbindung multiplext. Durch die Verwendung des offenen Protokolls WebMUX wurde es möglich, die Verbindung zwischen Hidden Service und Contact Point effizient zu nutzen. Die Funktionalität des Contact Points ist optional für den Betreiber eines Shalon Servers. Da diese Contact Points der Weiterleitung Kommunikation zwischen Client und Hidden Service dienen, wurde hier die Möglichkeit geboten, die Art der Weiterleitung zu reglementieren. Der Betreiber eines Shalon Servers mit Contact Point hat die Wahl zwischen verschlüsselter und unverschlüsselter Weiterleitung der Kommunikation zum Hidden Service. Im Gegensatz zu den Tor Hidden Services werden hier keine zusätzlichen Introduction Points für die Kommunikation zwischen Clients und dem Hidden Service benötigt. Durch diese Tatsache ist die Komplexität des Pro-
Kapitel 7. Zusammenfassung & Ausblick

tokolls geringer als die der Tor Hidden Services. Die Clients können hier außerdem direkt über die Contact Points mit dem Hidden Service kommunizieren.

Weithin verfügt das entwickelte Konzept über eine höhere Flexibilität. Statt ausschließlich über das Anonymisierungsnetzwerk zu kommunizieren, sind hier auch Verbindungen zum Hidden Service erlaubt, die nicht aus dem Anonymisierungsnetzwerk selbst entspringen. Wenn die Contact Points eines Hidden Service bekannt sind, können diese direkt angesprochen werden. Geschieht dies nicht über Shalon, so sind die Verbindungen nicht automatisch anonymisiert. Hierbei existiert ein klarer Kompromiss zwischen Anonymität und Geschwindigkeit.


Die abschließende Evaluation des entwickelten Konzepts hat gezeigt, dass die reduzierte Komplexität einen Vorteil gegenüber Tor bietet. Sowohl bezüglich der Setupzeiten des Hidden Service, also der Zeit die gebraucht wird um den Hidden Service zu veröffentlichen, als auch bezüglich der Zeit, die Clients brauchen, um diesen den Hidden Service zu kontaktieren, ist das vorliegende Konzept dem von Tor überlegen. Die Setupzeit der Shalon Hidden Service ist etwa um den Faktor 1,8 schneller als die Setupzeit der Tor Hidden Services. Des Weiteren ist auch die Kontaktzeit der
Shalon Clients unter Laborbedingungen um etwa den Faktor 2,3 schneller als die der Tor Hidden Service Clients. Durch das Zusammenführen der Funktionalitäten von Introduction Point und Rendezvous Point zu einem Contact Point, ergibt sich allerdings eine erhöhte Last für den Contact Point.

Da sowohl die RTT als auch der Durchsatz entscheidend zur wahrgenommenen Qualität des Hidden Services beitragen, erzielt das Shalon Hidden Service Design einen deutlichen Vorteil gegenüber den Tor Hidden Services. Die RTT Werte sind unter Laborbedingungen um den Faktor 18 geringer als die RTT Werte der Tor Hidden Services. In der Evaluation wurden detaillierte Vergleiche der Komponenten von Tor und Shalon Hidden Services vorgenommen. Zusätzlich wurden diese Vergleiche nicht nur im lokalen Labornetzwerk durchgeführt, sondern auch mit dem PlanetLab Testbed. Vor allem diese Ergebnisse zeigen ein Verhalten, unter nahezu real-world Bedingungen.

Es hat sich ebenfalls gezeigt, dass das WebMUX Protokoll vor allem für den Datentransfer geeignet ist, der beim Webseitenablauf auftritt. Zusätzlich zu dieser Eigenschaft erweist sich WebMUX Protokoll als äußerst fair unter den Sessions einer Multiplexingverbindung bezüglich der Verteilung der verfügbaren Bandbreite. Unter Laborbedingungen übertrifft der Durchsatz der Shalon Hidden Services den der Tor Hidden Services etwa um den Faktor 1,4.

Weiterhin hat die Sicherheitsanalyse gezeigt, dass das hier vorgestellte Design trotz seiner reduzierten Komplexität Schutz gegen verschiedene Angriffe bietet. Es wird nicht der gleiche Schutz wie von den Tor Hidden Services erreicht, aber in Teilbereichen wie dem Denial of Service Schutz der Contact Points, bietet es vermutlich mehr Schutz als die Tor Hidden Services. Im Ausblick auf das Thema werden einige mögliche Erweiterungen zur Steigerung des Schutzes aufgegriffen.

7.2 Ausblick

Hier werden nun einige Ideen aufgeführt, die während der Bearbeitung der Diplomarbeit entstanden sind. Diese sollen einen Ausblick auf mögliche Erweiterungen, Sicherheitsimplikationen und Probleme geben.

7.2.1 Contact Point Protokoll

dazu führen, dass der Contact Point, aufgrund der erschwerten Nachweisbarkeit von Hidden Service Streams, weniger leicht juristisch für sein Handeln verantwortlich gemacht werden kann.


7.2.2 Sichere Deskriptoren


7.2.3 Replikation von Deskriptoren

Dieses Konzept könnte zur Folge haben, dass die Hidden Services besser gegen einen vollständigen Ausfall aller Contact Points geschützt sind. Wenn der Angreifer nicht alle Deskriptoren des Hidden Service kennt, so können nicht alle Contact Points angriffen werden.

### 7.2.4 Server Circuit Multiplexing

Die neuartigen Multiplexingtunnel von Shalon sind zunächst nur clientseitig relevant. Eine weitere interessante Idee besteht im Multiplexen der Verbindungen zwischen den Shalon Servern. Ein denkbares Szenario könnte wie folgt aussehen:


### 7.2.5 Guard Nodes

Guard Nodes zum Schutz für Hidden Services wurden von Øverlie und Syverson in [22] vorgeschlagen. Diese Erweiterung erzielt, verglichen mit den ebenfalls vorgeschlagenen Valet-Nodes [16], einen größeren Mehrwert. Das Konzept der Guard Nodes in Shalon aufzunehmen, würde wahrscheinlich den Schutz der Hidden Service erhöhen. Indem der Anfang des Tunnels, zwischen Hidden Service und Contact Point, durch einen Guard Node gebildet wird, sinkt die Gefahr, durch einen Angreifer direkt am Anfang des Tunnels enttarnt zu werden. Auch die vorgeschlagene Schichtung von Guard Nodes ist denkbar. Neben der technischen Realisierung ist ebenfalls ein geeignetes Vertrauensmodell für die Betreiber der Shalon Server notwendig. Dies muss so angelegt werden, dass die Knoten, die als Guard Nodes fungieren sollen, tatsächlich vertrauenswürdiger sind, als die übrigen Knoten des Netzwerkes.

Das Konzept der Guard Nodes, kann neben dem besseren Schutz des Hidden Services, auch dazu beitragen, dass normale Shalon Clients besser gegen Enttarnung geschützt werden.
7.2.6 Quality of Service der Contact Points

Die in Kapitel 4.1 beschriebene *Death-Ratio* stellt eine rudimentäre Grundlage zur Beurteilung der Zuverlässigkeit von Contact Points dar. In diese Größe fließen allerdings keine Information über den Grund für den Ausfall des Contact Points mit ein. Fällt ein Contact Point einem Denial of Service Angriff zum Opfer, geht dies negativ in seine Bewertung durch den Hidden Service ein. Um hierfür eine Lösung zu bieten, könnte der Contact Point um Heuristiken erweitert werden, die es erlauben, einen Angriff auf den Contact Point Port zu erkennen. Würde auf dem Contact Point zusätzlich ein Mechanismus existieren, der der direkten Weiterleitung zum Hidden Service vorgelagert ist, so könnte der Contact Point den Hidden Service über den Ausfallgrund informieren. Dies wäre möglich, da die Verbindung zwischen Hidden Service und Contact Point somit nicht automatisch von dem Denial of Service Angriff betroffen sein würde.

Weiterhin wäre auch eine Beurteilung der Verbindung zwischen Hidden Service und Contact Point basierend auf RTT Werten und Durchsatz denkbar. Da die Verbindung zwischen Hidden Service und Contact Point eine Multiplexingverbindung ist, könnten diese Informationen über eine eigene Session ermittelt und ausgetauscht werden.

Ebenso würde sich eine Authentifizierung der Hidden Service Clients über eine gesonderte Session zwischen Contact Point und Hidden Service erreichen lassen.

7.2.7 Alternative Implementierung

Da alle hier verwendeten Protokolle dokumentiert sind, könnte eine alternative Implementierung in einer von JAVA verschiedenen Programmiersprache angestrebt werden. Für das für die Hidden Services benötigte WebMUX Protokoll steht neben der JAVA Implementierung außerdem ein C-Bibliothek zur Verfügung. Es jedoch ebenfalls ein Client in einer anderen Programmiersprache wie etwa Python implementiert werden. Da die Möglichkeit besteht den Deskriptorabruf über HTTP(S) zu vollziehen, stellt die Implementierung des WebMUX Protokolls hierbei die größte Herausforderung dar.

7.3 Offene Probleme

Hier werden nun einige Probleme angesprochen die während der Diplomarbeit beobachtet wurden. Zum Teil sind im vorigen Kapitel Lösungen für diese Probleme vorgeschlagen worden, für andere Probleme hingegen ist ein Lösungsvorschlag zurzeit unklar.

- **Proxy Server Read Timeouts**: Bei den HTTP Befehlen CONNECT und GET ist es nicht sinnvoll das Read-Timeout des Proxy Servers sehr hoch anzusetzen, oder sogar vollständig zu deaktivieren. Dies führt dazu, dass für den BIND Befehl, den der Hidden Service nutzt, ein Problem entsteht. Beim Aufbau der Proxy Kette zum Contact Point ist für die Proxys der Kette nicht
7.3. **Offene Probleme**

Klar, dass es sich um eine Contact Point Verbindung handelt. Daher werden diese bei der Weiterleitung automatisch das konfigurierte Proxy Read-Timeout auf die Verbindung anwenden. Für den Fall, dass bis zu dem Timeout nichts gelesen wird, wird die Verbindung von den Proxy Servern beendet. Dies hat zur Folge, dass die Verbindung zwischen Hidden Service und Contact Point beendet wird. Ist das Read-Timeout der Proxy Server entsprechend niedrig, so muss der Hidden Service ständig neue Verbindungen zu dem Contact Point aufbauen. Dies ist insbesondere ungünstig, wenn der Hidden Service beständig erreichbar sein soll.

Um dieses Problem zu umgehen, könnte der Timeout der Proxy Server so angepasst werden, dass dieser mit dem Timeout der Contact Point Verbindungen übereinstimmt. Für diese Verbindungen kann es sinnvoll sein, wenn die Contact Points in einem bestimmten, nicht zu niedrig gewählten, Intervall wechseln können.


- **Sicherheitsimplikationen**: Da das hier vorgestellte Design von geringerer Komplexität als das Tor Hidden Service Design ist und zusätzlich auf offenen standardisierten Protokollen aufgebaut wird, ist nicht auszuschließen, dass diese Tatsache unklare Sicherheitsimplikationen birgt. Durch die Kombination der Introduction Point und den Rendezvous Point Funktionalitäten zu Contact Points entstehen eventuell neue Risiken für den Betreiber. Außerdem wird der Contact Point damit zum *Single Point Of Failure* um den Hidden Service zu kontaktieren. Des Weiteren ist es mit der aktuellen DHT Implementierung möglich, über eine sequenzielle Suche alle Einträge abzurufen.


\[^{1}\text{einzelne Fehlerstelle}\]
2.1 Tor Netzwerk .............................................. 6
2.2 Onion-Routing ........................................... 7
2.3 Tor Hidden Service mit einem Introduction Point ....... 9

3.1 Valet-Service ............................................ 16
3.2 Valet-Service ohne Rendezvous Point .................. 17
3.3 Valet-Service mit optionalem Rendezvous Point ....... 18

4.1 Shalon Netzwerk .......................................... 40
4.2 Shalon Hidden Services Übersicht ....................... 41
4.3 TLS-Gateway Modus ..................................... 44
4.4 Open-/Public Modus ..................................... 45
4.5 Non-Shalon Kontaktaufnahme ........................... 50
4.6 Shalon DHT Query Web Interface ....................... 52
4.7 Formatierte Hidden Service Query Antwort .......... 53
4.8 Formatierte Proxy Query Antwort ..................... 54
4.9 Contact Point Architektur ................................ 55
4.10 Experimenteller WebMUX-Proxy ......................... 61
4.11 Shalon Hidden Service Setup .......................... 62
4.12 Hidden Service Kontakt (Shalon Client) ............... 63

5.1 Durchsatz ................................................. 67
5.2 Thread Durchsatz Vergleich ............................. 67
5.3 Thread Durchsatz ......................................... 68
5.4 Circuit Aufbauzeit (Shalon in JAVA) .................... 69
5.5 Circuit Aufbauzeit (Shalon in C) ....................... 70
Abbildungsverzeichnis

5.6 DHT mit 300 Knoten ........................................ 71
5.7 DHT mit 200 Knoten ........................................ 72
5.8 Anzahl der benötigten Suchoperationen ................. 73
5.9 Anzahl der benötigten Suchoperationen & doppelte Funde 73
5.10 Langlaufende Deskriptorsuche ......................... 74
5.11 Hidden Service Client Kontaktzeiten ................. 75
5.12 Shalon Clientarten Kontaktzeiten ..................... 76
5.13 Client Kontaktzeiten (Lab vs PlanetLab) ............ 76
5.14 Durchsatz verschiedener Clientarten ................. 77
5.15 Durchsatz pro Thread ....................................... 78
5.16 Hidden Service Round Trip Zeiten .................... 78
5.17 Contact Point Setup mit verschiedener Tunnellänge 79
5.18 Contact Point Setup (Lab vs. PlanetLab) ............ 80
5.19 Shalon Clientarten Kontaktzeiten (PlanetLab) ...... 80
5.20 Einfluss der Tunnellänge auf Kontaktzeiten der Hidden Service Clients (PlanetLab) ................. 81
5.21 Durchsatz verschiedener Clientarten (PlanetLab) ... 82
5.22 Hidden Service Round Trip Zeiten (PlanetLab) .... 83
5.23 Hidden Service Setup (PlanetLab) .................... 83
5.24 Tor Hidden Service Client Kontaktzeiten ............ 84
5.25 Vergleich von Tor und Shalon Client Kontaktzeiten 85
5.26 Vergleich von Tor und Shalon Durchsatz ............ 86
5.27 Bandbreitenverteilung auf parallele Threads ......... 86
5.28 Tor Durchsatz ............................................... 87
5.29 Vergleich von Tor und Shalon RTT ................... 87
5.30 Hidden Service Setup ..................................... 88
5.31 Hidden Service Setup ..................................... 88
5.32 Durchsatz der direkten Verbindung ................ 89
5.33 Durchsatz der WebMUX Verbindung ................. 90
5.34 Durchsatz: WebMUX vs. direkte Verbindung ....... 90
5.35 RTT: WebMUX vs. direkte Verbindung ............. 91
A.1 Suchoperationen vs. Duplicates - 200 Knoten ......... 115
<table>
<thead>
<tr>
<th>Abbildungsverzeichnis</th>
<th>109</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2 Suchoperationen vs. Failures - 200 Knoten</td>
<td>116</td>
</tr>
<tr>
<td>A.3 Suchoperationen vs. Failures - 300 Knoten</td>
<td>116</td>
</tr>
<tr>
<td>A.4 DHT Größe vs. Duplicates</td>
<td>117</td>
</tr>
<tr>
<td>A.5 DHT Größe vs. Failures</td>
<td>117</td>
</tr>
<tr>
<td>A.6 DHT Größe vs. Search Time</td>
<td>118</td>
</tr>
<tr>
<td>A.7 Hidden Service Setup vs. Tunnel Length</td>
<td>119</td>
</tr>
<tr>
<td>A.8 Deskriptor Download &amp; Hidden Service Contact</td>
<td>119</td>
</tr>
</tbody>
</table>
Abbildungsverzeichnis
Literaturverzeichnis


[27] Roger Dingledine, Karsten Loesing, Steven J. Murdoch and Christian Wilms: Design and Evaluation of Tor Hidden Service Performance Improvements.


A.1 DHT Messungen

Abbildung A.1: Suchoperationen vs. Duplicates - 200 Knoten
Abbildung A.2: Suchoperationen vs. Failures - 200 Knoten

Abbildung A.3: Suchoperationen vs. Failures - 300 Knoten
A.1. DHT Messungen

Abbildung A.4: DHT Größe vs. Duplicates

Abbildung A.5: DHT Größe vs. Failures
Abbildung A.6: DHT Größe vs. Search Time
A.2  Hidden Service Setup & Hidden Service Contact

Abbildung A.7: Hidden Service Setup vs. Tunnel Length

Abbildung A.8: Deskriptor Download & Hidden Service Contact