

Hauptseminar P2P-Technologien

Betreuer:
Frank Kargl, Stefan Schlott und Jürgen Nagler-Ihle

Wintersemester 2003/2004
Abteilung Medieninformatik
Universität Ulm

Inhaltsverzeichnis

1	Free Haven	2
1.1	Das Projekt	2
1.1.1	Einleitung	2
1.1.2	Entwicklungsziele	3
1.1.3	Core Team	4
1.2	Überblick	5
1.2.1	Topologie	5
1.2.2	Exkurs - Mixnet	5
1.2.3	Kommunikation	6
1.2.4	Implementierung	7
1.3	Benutzerschnittstelle	7
1.3.1	Veröffentlichen	7
1.3.2	Empfangen	9
1.3.3	Widerrufen	9
1.4	Servnet	10
1.4.1	Reputation System	10
1.4.2	Handel	11
1.4.3	Buddy System	12
1.4.4	Neue und alte Server	14
1.5	Angriffe	15
1.5.1	Einstufung	15
1.5.2	Kommunikationskanal	15
1.5.3	Datenintegrität	16
1.5.4	Vetrauensnetzwerk	17
1.6	Schlusswort	18

Kapitel 1

Free Haven

Alexander Bernauer
alexander.bernauer@informatik.uni-ulm.de

Abstract: Das Ziel des Free Haven Projekts ist es, ein verteiltes, zuverlässiges und persistentes Datenspeicherungssystem für Publikationen zu entwickeln. Im Vordergrund steht dabei die Anonymität der Autoren, Leser und Server und der Schutz vor gezieltem Auffinden und Zerstören von Daten durch mächtige Angreifer. Mittels harter Kryptographie und dem Einsatz von Mixnets wird versucht, die Anforderungen dieses Peer-to-Peer Netzwerkes zu realisieren. Effizienz und Geschwindigkeit sind dabei untergeordnete Ziele des Designs. Eines der Probleme des Free Haven Projekts ist es, eine gewisse Verlässlichkeit der Teilnehmer zu erreichen, ohne dabei die Anonymität zu opfern. Doch es gibt noch einige andere offene Probleme, die gelöst werden müssen. Grund dafür sind v.a. die vielen potentiellen Gegner mit ihren reichhaltigen Angriffsmöglichkeiten, die man zu erwarten hat.

1.1 Das Projekt

1.1.1 Einleitung

Mit Napster wurden Peer-to-Peer (P2P) Netzwerke in der breiten Öffentlichkeit bekannt. Und seit seiner dezentralen Weiterentwicklung in Gnutella ist klar geworden, wie mächtig das Konzept ist. Weitere Protokolle wie Kazaa und eDonkey kamen auf und mittlerweile sind sie so weit entwickelt, dass man bequem seinem Filesharingserver eine Liste von Daten angibt und sie dann vielleicht eine Woche später besitzt. Doch haben bisher existierende P2P Netzwerke ein entscheidendes Problem: man ist nicht anonym. Durch das Verteilen von urheberrechtlich geschützten Daten macht man sich Gegner. Und diese haben Möglichkeiten, einen zu finden und zur Verantwortung zu ziehen.

Bald wurde klar, dass man P2P Netze braucht, in denen man anonym agieren kann. Jedoch nicht primär, um die Musikindustrie abzuhängen, sondern aus einem viel ernsteren Grund: Freiheit. Die Freiheit, seine Meinung äußern zu dürfen, Machthaber zu kritisieren, Missstände und Verschwörungen aufzudecken. Und das alles ohne die Gefahr, erwischt zu werden und dabei vielleicht

seine Zukunft oder gar sein Leben zu verlieren. Am aktuellen Beispiel China [9] wird klar, was gemeint ist. Das selbe Interesse haben diejenigen, die freie Informationen, ungefilterte Berichte von Ereignissen oder Meinungen politischer Querdenker beziehen möchten. Schlicht: es geht um die Informationsfreiheit.

Free Haven hat es sich zum Ziel gemacht, den Menschen Informationsfreiheit in ihrer bestmöglichen Form zu geben. Man ist sich sicher, dass Menschen ein Recht auf Informationsfreiheit haben. Doch man muss sich die Frage stellen lassen, ob das ganze auch moralisch in Ordnung ist. Ein System wie Free Haven kann z.B. für kriminelle Zwecke missbraucht werden. Die meistgenannten Beispiele sind hier Kinderpornographie und natürlich der Terrorismus. Interessant wird das vor allem wenn man bedenkt, dass es kein Zurück mehr gibt, sobald das Design fertiggestellt wurde und funktionstüchtig ist. Es ist die Natur von Free Haven, dass niemand, auch nicht die Erschaffer selber, es später aufspüren können. Wenn es einmal aktiv ist, kann es sich nur noch selber kontrollieren.

Free Haven weicht dieser Frage nicht aus, sondern beschäftigt sich vor allem in [4] intensiv damit. Interessierten sei Abschnitt 4 aus dieser Master Thesis von Roger Dingledine empfohlen. Zusammenfassend kommt er dort zum Schluss, dass die Möglichkeit für Menschen, sich frei, unzensiert und anonym äußern zu können, schwerer wiegt als die Tatsache, dass man das System für weniger tugendhafte Dinge verwenden könnte. Um diesen Schluss zu kräftigen muss man sagen, dass es in Free Haven einen Mechanismus gibt, so dass man nur so viel Daten im Netz speichern kann, wie man selber an Speicherkapazitäten anbietet. Es ist also unmöglich, auf Kosten anderer seine Daten zu verbreiten und v.a. ist es unmöglich, dass andere Inhalte verdrängt werden und damit das komplette Netz missbraucht wird.

1.1.2 Entwicklungsziele

By providing tools to enable safer and more reliable communication for organizations fighting for increased rights of individuals, as well as strengthening the capabilities of individuals to speak out anonymously about their situations, the members of the Free Haven Project hope to reinforce the rights of freedom of speech and freedom of information as integral parts of everyday life.

Dieses Zitat aus der Free Haven Projekt Seite [1] macht klar, worum es geht. Ein System zu entwickeln, welches den Menschen Redefreiheit und freien Zugang zu Informationen sichert. Damit ist man sich aber auch bewusst, wer die Gegner und zukünftigen Angreifer auf das System sein werden. Neben individuellen Personen erwartet man die Mächtigen gegen sich zu haben. Es wird Angriffe von Regierungen und ihren Geheimdiensten oder großen Firmen geben, um Autoren zu finden, Dokumente zu verändern oder zu zerstören, Leser zu verfolgen, Betreiber von Servern zu finden und die Zuverlässigkeit des Systems zu kompromittieren. Dementsprechend werden auf [2] folgende Anforderungen an das System definiert:

An oberster Stelle steht die **Anonymität** der Autoren, der Leser und der Free Haven Server. Unter dieser Bedingung muss es möglich sein, Dokumente im Netz zu **speichern** und wieder **empfangen** zu können. Das System soll eine Möglichkeit bieten, Dokumente **verfallen** zu lassen. Dazu bestimmt der Autor das Verfallsdatum, bis zu dessen Ablauf die Verfügbarkeit und Un-

veränderlichkeit garantiert wird. Das **Hinzufügen** neuer Server muss reibungslos und unkompliziert gehen. Schließlich braucht man noch einen Mechanismus, inaktive und tote Server zu **erkennen**, um sie aus weiteren Aktivitäten auszuschließen.

Zu diesen grundlegenden Mechanismen werden die folgenden Anforderungen an das Design gestellt: Das System muss **robust** sein. Das bedeutet, dass ein Ausfall von vielleicht bis zur Hälfte der Server keinen Verlust von Daten nach sich ziehen darf. Außerdem müssen die möglichen Schäden am Netz verursacht durch fehlerhafte und von Angreifern kontrollierte Server minimiert werden. Da die Erfahrung zeigt, dass mit dem Grad der Komplexität prinzipiell die Anzahl der möglichen Schwachstellen wächst, müssen die Protokolle **einfach** sein. Auch müssen sie realistischen technischen Erwartungen entsprechen. Ziel ist nicht eine akademische Formulierung eines optimalen Netzes, sondern die Implementierung eines bestmöglichen. Das Konzept soll **modular** sein, so dass ein Upgrade einzelner Komponenten reibungslos im Betrieb möglich ist. Des weiteren muss das System **dezentral** sein. Es darf für keinen Teil des Protokolls einzelne wenige Server mit spezieller Rolle geben. Solche Flaschenhälse wären attraktive Ziele für Angreifer. Außerdem würde die Leistungsfähigkeit dieser Server die Performance des gesamten Netzwerkes bestimmen. Jedem Betreiber eines Free Haven Servers will man freiräumen, wie paranoid er seinen Server konfiguriert und wie viel Speicherplatz er dem Servnet zur Verfügung stellen will. Das bedeutet, dass man das System **flexibel** designen muss. Man erhofft sich durch diese Freiheiten den Betrieb eines Servers attraktiver zu machen. Um wirkliche Sicherheit zu erlangen, müssen die Komponenten, auf die sich das System stützt, **open source** im Sinne der Free Software Foundation sein. Zu guter Letzt ist das System **inhaltsneutral**. Das bedeutet, dass alle Daten ohne Berücksichtigung der Inhalte gleich behandelt werden.

Effizienz spielt eine untergeordnete Rolle. Man erlaubt sich Zeit- und Datenoverhead, wenn man dadurch mehr Anonymität und Robustheit erreichen kann. Nur für den Fall, dass es auf die Entwicklungsziele keinen negativen Einfluss hat, wird man sich für eine effizientere Methode entscheiden. Jedoch ist man sich bewusst, dass man sich den Grenzen der technischen Möglichkeiten beugen muss.

1.1.3 Core Team

Das Core Team des Projektes wird unter [3] vorgestellt:

Roger Dingledine lebt als Kryptologe und Experte für Netzwerksicherheit in dem speziellen Bereich zwischen Theorie und Praxis. Er bevorzugt es die richtig harten Probleme anzugehen, so dass er eines Tages wirkliche Lösungen erschaffen kann. Sein derzeitiges Interesse gilt u.a. der anonymen Publikation und Kommunikation, dem Widerstand gegen Zensur, der Widerstandsfähigkeit dezentraler Netze gegen Angriffe und Reputationsprotokollen.

Michael J. Freedman macht zur Zeit seinen Doktor in Computerwissenschaften an der New York University, nachdem er seinen Master und Bachelor am MIT erlangt hat. Sein Forschungsinteresse konzentriert sich auf Computer und Netzwerk Sicherheit, verteilte Systeme und Kryptographie. Er versucht dabei Theorie und Systeme zu vereinen, um reale Probleme mit beweisbaren Lösungen zu beseitigen.

David Molnar hat 1993 begonnen, PGP zu verwenden. Es interessierte ihn herauszufinden, warum es funktioniert und er begann sich seit dem mit Kryptographie zu beschäftigen. Heute ist er Student und verfolgt weiterhin Sicherheitsfragen durch Vorlesungen, Newsgroups, Mailinglisten, Abhandlungen von Konferenzen und die DEF CON in seiner Heimatstadt Las Vegas. David ist ein ACM Student und Mitglied der Internationalen Vereinigung für Kryptographische Forschung.

1.2 Überblick

1.2.1 Topologie

Das Free Haven Netzwerk besteht aus einer beliebigen Anzahl von Servern, die in ihrer Gesamtheit als das Servnet bezeichnet werden. Jeder Server speichert Daten von anderen Servern um im Gegenzug seine Daten im Servnet speichern zu können. Jeder Server stellt eine autarke Einheit dar, die von ihrem Betreiber gewartet wird. Durch Free Haven wird nur das Kommunikationsprotokoll vorgeschrieben. Im Prinzip kann es dafür verschiedene Implementierungen geben. Das und vor allem die Konfiguration der Server liegen voll in der Hand ihrer Betreiber.

Ein Free Haven Server besteht aus drei Komponenten. Den Kern stellt das Haven Modul dar. Ein Comm Modul stellt die Verbindung zum Netzwerk her und ein Datenbankmodul bietet eine Schnittstelle zum Speichern relevanter Daten. Das Haven Modul beherbergt das für das Reputation System (Abschnitt 1.4.1) nötige Trustmodul zur Meinungsbildung über andere Server und das Handelsmodul, welches den Handel der Shares verwaltet (Abschnitt 1.4.2). Das Comm Modul ist spezifisch auf das Kommunikationsmedium zugeschnitten. Typischerweise kommen für die Kommunikation Mixnets zum Einsatz. Prinzipiell ist diese Wahl jedoch beliebig, so dass man in Zukunft einfach auf eventuell bessere Systeme umsteigen kann. Das Datenbankmodul besitzt keine besonderen Anforderungen und kann durch einen üblichen SQL Server realisiert werden. Das modulare Konzept erlaubt es, die drei Komponenten auf eigenen Rechner im gleichen Intranet zu betreiben. Es ist sogar möglich, dass ein Datenbankmodul mehrere Haven Module bedient.

1.2.2 Exkurs - Mixnet

Ein Mixnet stellt einen anonymen und verschlüsselten Kommunikationskanal für Teilnehmer eines Netzwerkes bereit. Es besteht aus einer Menge von Computern mit jeweils einem asymmetrischen Schlüsselpaar. Wenn ein Computer A einem Computer B eine Nachricht übermitteln möchte, wählt er für die Nachricht einen Pfad durch das Mixnet mit beliebigen Computern als Knoten. An die eigentlichen Daten der Nachricht hängt A die Adresse von B an. Dann verschlüsselt er dieses Paar mit dem öffentlichen Schlüssel des letzten Knotens auf dem Pfad. An diesen neuen Datenblock hängt A die Adresse des letzten Knotens an und verschlüsselt dieses Paar mit dem öffentlichen Schlüssel des vorletzten Knotens. So baut er Schicht für Schicht die Nachricht zusammen, bis er am Ende die Adresse des ersten Knotens an den Datenblock hängt und mit dessen öffentlichen Schlüssel verschlüsselt. Auf dem Weg durchs Mixnet entschlüsselt

jeder Knoten auf dem Pfad die Nachricht und verschickt den darin enthaltenen Datenteil an die angegebene Adresse weiter. Auf diese Art kennt nur der erste Knoten den Sender und nur der letzte Knoten den Empfänger. Alle Knoten dazwischen wissen überhaupt nichts. Prinzipiell ist ein Pfad durch ein Mixnet sicher, so bald mindestens ein beteiligter Knoten korrekt arbeitet.

Normalerweise muss der Computer A die Adresse des Computers B kennen, um ihm eine Nachricht zukommen zu lassen. Im Freehaven Servnet soll aber jeder Server anonym sein. Deshalb erstellt B einen sogenannten Reply Block. Im Prinzip handelt es sich dabei um eine fertige Nachricht, die durchs Mixnet zum Computer B führt, in die der Computer A nur noch die eigentlichen Daten einarbeiten und an einen ersten Computer im Mixnet schicken muss. Dieser Reply Block kann nun unter einem Pseudonym veröffentlicht werden, so dass jeder an B eine Nachricht schicken kann. Seine reale Adresse muss B dabei niemals preisgeben. Wenn nur der Sender anonym bleiben möchte, so hat er die Möglichkeit, zusammen mit der Nachricht an B einen Reply Block mit zu schicken. Diesen verwendet B dann einfach für seine Antwort ohne zu wissen, wer A eigentlich ist.

Es gibt viele Angriffsmöglichkeiten auf ein Mixnet. Die meisten bestehen aus statistischen Analysen des Datenverkehrs durch einen Knoten. Dabei wird versucht, eingehende und ausgehende Nachrichten miteinander in Beziehung zu bringen, um so den Pfad durch das Mixnet nachvollziehen zu können. Um einfache Angriffe dieser Art zu verhindern, muss jeder Knoten jede einzelne Nachricht für eine zufällige Zeit zurückhalten bevor er sie weitersendet. Deshalb haben Mixnets sehr hohe Latenzzeiten. Auch muss ein Knoten jede Nachricht nach dem Extrahieren der Adressen mit Zufallszahlen bis zur ursprünglichen Größe wieder auffüllen, um keine Zusammenhänge durch die Nachrichtengröße zu offenbaren. Mehr zu möglichen Angriffen und deren Gegenmaßnahmen gibt es im Abschnitt 1.5. Für jetzt ist es wichtig zu sehen, dass die Anonymität in einem Mixnet mit der Anzahl der Benutzer zunimmt. Grund dafür ist einfach die Vielzahl der möglichen Zusammenhänge zwischen ein- und ausgehenden Nachrichten pro Knoten. Damit steigt die zu analysierende Datenmenge für einen Angreifer an und erschwert oder verhindert gar die Analyse.

1.2.3 Kommunikation

Jeder Free Haven Server besitzt ein asymmetrisches Schlüsselpaar und einen oder mehrere Mixnet Reply Blöcke. Der öffentliche Schlüssel des Servers dient als Pseudonym zur Identifizierung innerhalb des Servnets. Die Benutzer des Netzwerkes, also die Autoren und die Leser, kontaktieren das Servnet über eine spezielle Benutzerschnittstelle. Deren Aussehen ist noch nicht genauer spezifiziert. Wahrscheinlich wird jeder Betreiber entscheiden können, ob er eine Benutzerschnittstelle anbieten möchte, oder nicht. Es ist noch nicht klar, über welche Kanäle diese Kommunikation statt finden wird. Zur Diskussion stehen öffentlich bekannte Reply Blocks von Servern, die bereit sind, Daten ins Netz einzuführen.

Prinzipiell ist es so, dass sich das Free Haven Design nicht darum kümmert, wie ein Benutzer einen Server zum Publizieren oder Empfangen von Daten finden kann. Auch stellt es keinen Mechanismus bereit, der einem eine Auflistung aller Dokumente im Servnet liefert, oder gar eine Suchanfrage ermöglicht. Es wurde diskutiert, ob man so einen Directory Service zur Verfügung stellen möchte, hat sich dann allerdings dagegen entschieden. Die Gründe sind, dass

sich Verzeichnisstrukturen ständig ändern, Daten in Free Haven aber per Design unveränderbar sind. Außerdem ist die Hauptaufgabe des Netzwerkes die sichere Speicherung und nicht die einfache Verfügbarkeit. Abgesehen von der Tatsache, dass dieser Service auf Grund der hohen Latenzzeiten der Kommunikationskanäle nur sehr langsam arbeiten könnte.

Die Anonymität des Free Haven Netzwerkes steht und fällt mit dem Kommunikationsmedium. Bisherige Mixnet Systeme haben sich als nicht ausreichend sicher und zuverlässig herausgestellt. Deshalb hat das Free Haven Team selber an einem Protokoll geforscht. Das neueste Ergebnis trägt den Namen Tor und wird in [7] vorgestellt. Seine Vorzüge gegenüber herkömmlichen Onion Routing Protokollen sind u.a. Staukontrolle, Directory Servers, Integritätsprüfungen, variables Verhalten beim Verlassen des Mixnets und ein praktisches Design für Rendezvous Punkte. Auch versucht man, Mixnets mit einem ähnlichen Reputation System auszustatten, wie man es ins Free Haven Design integrieren möchte. Dieses System bietet eine Möglichkeit, sich an Hand von Vergangenem eine Meinung über die Zuverlässigkeit eines Teilnehmers zu machen, um ihn ggf. zu meiden.

1.2.4 Implementierung

In den folgenden Abschnitten wird erläutert, wie die Entwicklungsziele umgesetzt werden sollen. Dazu muss man folgendes anmerken:

Das System besteht in seiner Gesamtheit aus vielen voneinander abhängigen und zusammenspielenden Bereichen. Das bedeutet insbesondere, dass es Bereiche gibt, von denen manche Eigenschaften in Designentscheidungen anderer Bereiche begründet liegen. Das macht es unmöglich, das System in einem linearen Aufbau zu erklären. Ich habe mich für die folgende Reihenfolge entschieden, weil sie mir zweckmäßig erschien. Stellen, an denen noch nicht behandelte Bereiche eine Rolle spielen, sind mit einem Vorwärtsverweis versehen. Verweise in bereits behandelte Abschnitte gibt es nicht. Deshalb empfiehlt es sich, die folgenden Abschnitte in der angegebenen Reihenfolge zu lesen.

Das Free Haven Projekt befindet sich noch in einem sehr frühen Stadium. Viele Entscheidungen sind noch nicht getroffen und viele Probleme noch nicht gelöst. Deshalb wird man hier keine Komplettbeschreibung einer Implementierung finden. Viel mehr werden hier die prinzipiell existierenden Bereiche und ihr Zusammenspiel erläutert. Die formulierten Entscheidungen sind widerspruchsfrei, erheben aber nicht den Anspruch der Vollständigkeit. An manchen Stellen werden bisher diskutierte Alternativen zu Designentscheidungen angesprochen da der Grund für eine Ablehnung in vielen Fällen nicht offensichtlich ist. Auch zeigt es, welche Szenarien und v.a. Angriffsmöglichkeiten man bedenken muss, wenn man ein System entwickeln will, dass den Mächtigsten stand hält.

1.3 Benutzerschnittstelle

1.3.1 Veröffentlichen

Ein Autor, der ein Dokument publizieren möchte, braucht zu allererst einen Free Haven Server, welcher bereit ist, sein Dokument aufzunehmen und ins Netz einzuführen. Wie bereits erwähnt, kümmert sich das Free Haven Design nicht darum, wie ein solcher Server gefunden werden kann.

Der Autor versieht sein Dokument mit einem Verfallsdatum. Optional kann er es noch verschlüsseln, muss dann allerdings eigene Wege finden, wie die Leser an den Schlüssel kommen. Wenn er z.B. nicht offiziell aber dennoch anonym publizieren möchte, käme eine Verschlüsselung und gezielte Verbreitung des Schlüssels in Frage. Je nach dem, welche Dateigrößen der Server akzeptiert, muss der Autor sein Dokument splitten oder gar auffüllen.

Der Server, der das neue Dokument ins Servnet einfügen möchte, zerlegt es mittels Rabins Algorithmus zur Zerstreuung von Informationen [8] in n Shares. Dabei genügen jeweils k beliebige Shares, um das Dokument zu rekonstruieren. Der Robustheitsfaktor k kann dabei beliebig gewählt werden. Ein großes k erhöht das Risiko des Verlustes, da schon nach wenigen verlorenen Shares das Dokument nicht rekonstruiert werden kann. Dem gegenüber bedeutet ein kleines k größere Shares, da in jedes Share mehr Informationen einfließen müssen.

Zur Signierung der Shares wird ein asymmetrischen Schlüsselpaar erstellt. Der öffentliche Schlüssel ist damit ein Pseudonym für das Dokument. Gebraucht wird es v.a. dann, wenn man das Dokument aus dem Servnet wieder beziehen möchte. Ein Share besteht schließlich aus folgenden Komponenten:

- öffentlicher Signaturschlüssel
- Nummer des Shares
- Nummer des Buddys (vgl. Abschnitt 1.4.3)
- Gesamtanzahl n der Shares
- Robustheitsfaktor k
- Ablaufdatum in GMT
- Signatur über die obigen Inhalte

Man könnte es auch dem Autor überlassen, die Shares zu erstellen und zu signieren. Man hätte damit auch die Frage beantwortet, wie man das Pseudonym des Dokumentes in Erfahrung bringt. Allerdings muss man dann dem Autor trauen, die Shares richtig erstellt zu haben. Zur Diskussion steht, dass man es dem Server überlässt, ob er nur Dokumente oder Shares oder beides akzeptiert. Vorausgesetzt natürlich, dass der Server überhaupt Einsendungen annimmt.

Der Server, der die neuen Shares nun besitzt, versucht diese durch Handel im Servnet zu verteilen (vgl. Abschnitt 1.4.2). Wichtig ist dabei, dass er ungefähr nur so viel Daten durch Handel loswerden kann, wie er im Gegenzug bereit ist anzunehmen. Das bedeutet insbesondere, dass eigensinnige oder böswillige Benutzer das Netz nur insoweit mit ihren Daten fluten können, wie der Server bereit ist anzunehmen. Da dessen Kapazitäten endlich sind, ist der Überflutung des Netzes durch unsinige Inhalte Grenzen gesetzt. Als Betreiber eines Servers mit Schnittstelle zum Publizieren könnte man sich eigene Mechanismen ausdenken, die einem vor einer Datenflut schützen. Da die Identität eines Free Haven Servers geschützt ist, können diese Mechanismen mit weniger Anonymität auskommen, da sich ein Autor quasi hinter dem Server versteckt. Vorausgesetzt natürlich, der Autor traut dem Server.

1.3.2 Empfangen

Um ein Dokument zu empfangen muss der Leser das Pseudonym des Dokumentes, also den öffentlichen Schlüssel der Signatur, kennen. Auch muss er einen Free Haven Server finden, der ihm eine Benutzerschnittstelle bereitstellt. Um beide Probleme kümmert sich das Free Haven Protokoll wie in Abschnitt 1.2.1 beschrieben nicht.

Der Leser erzeugt ein eigenes asymmetrisches Schlüsselpaar und einen Reply Block, der zu einem Rechner führt, welcher das Dokument empfangen soll. Seinen öffentlichen Schlüssel, das Pseudonym des Dokumentes und den Reply Block sendet er an den Free Haven Server. Dieser verteilt nun einen Request Broadcast im Servnet. Jeder Server, der ein Share mit passendem Pseudonym besitzt, verschlüsselt diesen mit dem öffentlichen Schlüssel des Lesers und sendet ihn über den Reply Block durchs Mixnet. Der empfangende Rechner entschlüsselt die Shares und rekonstruiert das Dokument, so bald genügend Shares empfangen wurden. Falls der Autor sein Dokument verschlüsselt hat, muss man es jetzt noch entschlüsseln, um schließlich die gewünschten Informationen zu erhalten.

Optional könnte man den Request Broadcast vom sendenden Server signieren lassen. Dadurch könnte man verhindern, dass einzelne Server mit ständigen Anfragen lahmgelegt werden würden. Dieser könnte nämlich an Hand der Signatur erstens entscheiden, ob er dem Server überhaupt genug vertraut um seine Anfrage zu beantworten und zweitens bei ständigen Anfragen eine Beschwerde ins Servnet ablassen (vgl. Abschnitt 1.4.1) und neue Anfragen vom Comm Modul wegfiltren lassen.

1.3.3 Widerrufen

Man könnte dem Autor die Möglichkeit geben, ein von ihm publiziertes Dokument zu widerrufen, und es damit aus dem Netz zu entfernen. Implementieren könnte man das folgendermaßen: Der Autor wählt eine zufällige Zahl als Kennung und speichert ihren kryptographischen Hashwert als Teil eines jeden Shares. Bei einem Widerruf nennt er das Pseudonym des Dokumentes und die Kennung. Jeder Server, der ein Dokument mit passendem Pseudonym besitzt berechnet den Hashwert der angegebenen Kennung und vergleicht ihn mit dem gespeicherten. Stimmen die Werte überein, so entfernt er das Share.

Jedoch gibt es viele Gründe, die gegen diese Möglichkeit sprechen. Wenn ein Autor für verschiedene Dokumente den selben Hash verwendet entsteht dadurch ein Verbindung zwischen den Dokumenten. Ein Angreifer könnte das ausnützen, um einem Autor ein zweites Dokument zu unterstellen. Außerdem weist ein Hash in einem Share einen Besitz aus, der andernfalls nicht existieren würde. Ein Autor kann durch den Besitz der Kennung beweisen, dass er der Urheber des Dokumentes ist. Das bedeutet aber andersherum, dass man diesen Beweis gegen den Autor verwenden kann, wenn die Kennung bei ihm gefunden wird. Dann kann es zu einem Durcheinander im Buddy System kommen (vgl. Abschnitt 1.4.3) wenn nur ein Teil der Server den Widerruf erhalten, da einzelne Shares gegenseitig aufeinander aufpassen. Ein Angreifer kann das explizit ausnützen, um Chaos ins das Vertrauensnetzwerk zu bringen. Am stärksten spricht allerdings folgende Überlegung gegen eine Widerrufsmöglichkeit: wenn es diese Möglichkeit gibt, wird ein Angreifer vielleicht versuchen, denjenigen zu finden, der die Macht darüber hat, und ihn eventuell mit Gewalt dazu zwingen,

das Dokument zu entfernen.

Aus den genannten Gründen hat man beschlossen, in Free Haven keine Möglichkeit anzubieten, ein Werk zu widerrufen.

1.4 Servnet

1.4.1 Reputation System

Dezentrale Netze haben das Problem, dass man nicht alle Knoten kontrollieren kann. Vielmehr muss man sich darauf verlassen, dass der andere dem Protokoll entsprechend handelt. Alternativ baut man Sicherheitsmechanismen ein, um Fehlverhalten zu erkennen. Dabei gibt es prinzipiell zwei Klassen: die eine bezieht sich auf Fehlverhalten, welches z.B. durch Softwarefehler, falsche Konfiguration oder menschliches Versagen auftritt. Der andere Fall sind Fehlverhalten, die absichtlich hervorgerufen werden. Unabhängig von der Motivation spricht man dann von einem Angriff. Die verschiedenen Angriffsmöglichkeiten werden im Abschnitt 1.5 genauer beschrieben.

In einem anonymen Netzwerk hat man zusätzlich das Problem, dass man seinen Gegenüber nicht kennt. Es gibt nur Pseudonyme, die eine gewisse Zuordnung erlauben. Doch was hintert einen Angreifer daran, einfach ein neues Pseudonym anzunehmen, so bald er entlarvt wurde? Und wie reagiert man, wenn man einen Angriff entlarvt? Es ist in den meisten Fällen unmöglich, den Angreifer zu ermitteln oder sonst irgendetwas zu unternehmen, um ihn daran zu hindern, wieder anzugreifen. Außerdem muss man sich bewusst machen, dass ein Angriff nicht immer eine aktive Handlung bedeutet. Ein böser Server könnte zum Beispiel Shares einfach unbemerkt verschwinden lassen.

Um die Auswirkungen böser und unzuverlässiger Server auf den Dienst minimal zu halten, arbeitet man bei Free Haven an einem Reputation System. Ein Server kann das Vertrauen anderer Server erlangen, indem er das Protokoll korrekt befolgt. Verhält er sich falsch dann wird sich ein anderer über ihn beschweren, was sein Vertrauen im Servnet reduzieren wird. Hohes Vertrauen bedeutet im Gegenzug, dass andere zum Handeln bereit sind oder auf Dokumentanforderungen reagieren.

Das Design sieht folgendes vor: Jeder Server merkt sich zu jedem ihm bekannten Server zwei Werte: Trust und Metatrust. Trust gibt an, wie sehr man dem anderen vertraut, dass er sich dem Protokoll entsprechend verhält. Ein hoher Trust Wert bedeutet also, dass man glaubt, der andere sei weder böse, noch kompromittiert, noch fehlerhaft. Zu diesem Glauben kommt man, in dem man viele gute Erfahrungen mit diesem Server macht. Metatrust gibt an, wie sehr man den Aussagen des anderen glaubt, also ob man z.B. aus Beschwerden von ihm über dritte Konsequenzen zieht, oder nicht. Um das Zusammenspiel dieser Werte zu verdeutlichen, hier ein Beispiel: einem Server mit hohem Trust aber wenig Metatrust werde ich viele Shares durch Handel anvertrauen. Allerdings sinkt mein Vertrauen in ihn sehr schnell, wenn andere sich über ihn beschweren.

Es ist allerdings dem Betreiber freigestellt, wie genau er mit diesem Mechanismus umgeht. Zum Beispiel könnte er Metatrust für alle prinzipiell auf Null setzen und somit Beschwerden generell ignorieren und nur auf seine eigene Erfahrungen vertrauen. Oder er vergibt generell einen hohen Metatrust um Beschwerden über Fehlverhalten die entsprechende Geltung zu verschaffen.

Dabei kann er auf schlichte Behauptungen anders reagieren als auf beweisbares Fehlverhalten. Als Beweis dienen zum Beispiel Quittungen, die beim Handeln ausgestellt werden (vgl. Abschnitt 1.4.2).

Allerdings ist es so, dass in vielen Fällen Beschwerden nicht überprüft werden können. So bleibt es immer eine Frage des Vertrauens, ob man einer Beschwerde glaubt, oder nicht. Das macht das System empfindlich gegenüber manchen Angriffen, wie zum Beispiel Rufmord einer Gruppe verschworener Server gegenüber einem Dritten. Das ist ein noch offenes Problem. Es wird darüber diskutiert, ob man vielleicht das komplette Design überarbeiten muss, um das Reputation System von Anfang an mit zu berücksichtigen. Denn man ist sich sicher, dass mit dem Reputation System auf dem richtigen Weg ist.

1.4.2 Handel

Shares werden im Servnet ständig zwischen den Servern ausgetauscht. Ein solcher Vorgang wird als Handel bezeichnet und ist ein grundlegender Bestandteil des Designs von Free Haven. Dafür gibt es folgende Gründe:

Wenn es keinen Verkehr durch Handel geben würde, dann könnte ein Angreifer, der das Netz überwacht feststellen, wann ein neues Dokument publiziert wurde. Denn der Server, der dem Autor die Benutzerschnittstelle anbietet, muss die Shares des neuen Dokumentes im Servnet verteilen. Der Angreifer könnte diesen Verkehr entdecken und hätte damit den Server entlarvt. Durch das ständige Handeln wird also ein Grundrauschen im Servnet erzeugt, in dem neue Publikationen nicht auffallen. Auch erhöht sich der Grad der Anonymität eines Mixnets mit der Dichte des Verkehrs.

Außerdem würde es Probleme mit langlebigen Shares geben, wenn ein Betreiber seinen Free Haven Server vom Netz nehmen möchte, aber keine Möglichkeit hat, die Shares ordnungsgemäß los zu werden. So kann er einfach die langlebigen Shares gegen kurzlebige austauschen, deren Ablauf abwarten und dann einfach den Server schließen. Das bedeutet, dass man ohne Probleme Shares mit sehr langen Lebenszeiten erlauben kann. Zusätzlich bietet der Handel die Möglichkeit Shares los zu werden, gegen deren Inhalte man ethische Einwände hat und deshalb damit nicht in Verbindung gebracht werden möchte. Der größte Vorteil des Handels ist es, dass man Angreifern keine statischen Ziele bietet und somit das Aufspüren von Shares äußerst erschwert. Wenn ein Share über Jahre hinweg auf dem selben Server verbleibt, hat ein Angreifer viel Zeit, den aufgezeichneten Netzverkehr, der durch die Publikation entstanden ist, zu analysieren, um das Mixnet zu brechen. Zu guter Letzt bietet der ständige Handel viele Verhaltensdaten, die das Reputation System auswerten und damit feiner arbeiten kann.

Ein Handel soll fair sein. Um einen Handel dahingehend beurteilen zu können, braucht man einen Maßstab. Free Haven sieht dafür das Produkt aus der Dauer bis zum Ablauf und der Größe eines Shares vor. Lokale Konfigurationen der Betreiber legen auf Grund dieser Basis fest, auf welche Handelsvorschläge eingegangen wird, und auf welche nicht. Wie oben erwähnt könnte ein Betreiber z.B. planen, den Server vom Netz zu nehmen, und konfiguriert ihn deshalb so, dass er große Shares akzeptiert wenn sie kurzlebig sind. Ein anderes Beispiel wäre ein Betreiber, der nicht viele Speicherkapazitäten zur Verfügung stellen möchte, und deshalb kleine wenn auch langlebige Shares bevorzugt. Zusätzlich könnte ein Server mit Absicht für ihn unvorteilhafte Angebote annehmen, um

sich dadurch einen guten Ruf im Reputation System zu erlangen. Vor allem für neue Server ist diese Möglichkeit attraktiv (vgl. Abschnitt 1.4.4).

Ein Handel beginnt mit einem Angebot. Ein Server A entscheidet sich dazu, ein Share zu handeln. Dazu sucht er sich aus der Liste der ihm bekannten Free Haven Server einen Handelspartner B. Diese Entscheidung basiert zum einen auf dem Vertrauen in den anderen Server und auf der Vorgabe, nicht immer mit dem selben Server zu tauschen. Er bietet B also das Share zusammen mit Hinweisen auf das gewünschte Gegenstück an. Ist B interessiert so sendet er ein Share zurück. Wenn dieses Share den Erwartungen entspricht, so wird der Erhalt bestätigt. B antwortet seinerseits mit einer Bestätigung. Nun benachrichtigen beide Server die Buddys des gerade verschickten Shares darüber, dass von nun an ein anderer Server verantwortlich für seinen Buddy ist.

Zusammen mit den Empfangsbestätigungen wird jeweils eine signierte Auflistung aller relevanten Daten als Quittung verschickt. Diese sind das Pseudonym, der Index, das Ablaufdatum und die Größe des weggegebenen und des empfangenen Shares, ein Zeitstempel und die beiden Pseudonyme der beteiligten Server. Durch die Aufnahme der Ablaufdaten der Shares in die Quittung erhält diese automatisch einen eigenen Gültigkeitszeitraum. Läuft dieser ab, kann die Quittung gelöscht werden. Bis dahin dient sie als Beweis. Zum einen dafür, dass man nicht mehr für ein Share verantwortlich ist, zum anderen gegen falsche Beschuldigungen. Zusätzlich enthält die Quittung alle nötigen Informationen für eine Weiterleitung von Buddy-Benachrichtigungen (vgl. Abschnitt 1.4.3).

Möchte B den Handel nicht eingehen, so kann er einfach nicht auf das Angebot antworten. Allerdings läuft er damit Gefahr sich eine schlechte Bewertung fürs Reputation System einzufangen. Besser wäre es also mit einer Ablehnung zu antworten. Diese könnte man alternativ mit einer Begründung versehen, damit vielleicht in Zukunft passende Angebote gemacht werden können.

Wenn B auf die Bestätigung nicht antwortet, so besitzt A keinen Beweis dafür, dass er das Share weggegeben hat. Um sich selbst vor Beschuldigungen zu schützen bleibt ihm nur noch die Möglichkeit, eine Beschwerde über B im Netz zu verteilen. Für den Fall das B unzuverlässig ist, ist es von Vorteil, wenn A eine Kopie des Shares behält. Das selbe gilt für B in Bezug auf A. Das erhöht zwar die Redundanz und den benötigten Speicherplatz fördert aber die Robustheit des Systems. Auf eine Dokumentanfrage antwortet dann jeder Server, der das Share besitzt, eventuell mit Vermerk, dass dieses Share nur eine Kopie ist. Allerdings bedeutet das, dass Shares sehr lange auf dem selben Server verweilen, was man eigentlich durch Handeln vermeiden wollte.

1.4.3 Buddy System

Es ist nicht verlässlich festzustellen, ob ein fehlerhafter oder "böser" Server Daten löscht. Dennoch möchte man eine Persistenz der Daten erreichen ohne dabei die Anonymität der Server zu gefährden. Es gibt verschiedene Ideen, wie man das realisieren könnte.

Man könnte den Autor des Dokumentes damit beauftragen, regelmäßig zu kontrollieren, ob noch genug Shares im Netz sind, um das Dokument zu rekonstruieren. Wenn nicht dann verbreitet er eine Beschwerde und publiziert ggf. erneut. Dieser Ansatz hat allerdings den großen Nachteil, dass man sich auf den Autor verlassen muss. Vielleicht ist er nicht mehr in der Lage, diese Wartung seines Dokumentes zu machen. Oder er macht sie nicht häufig genug. Eventuell

besitzt er das Original seines Dokumentes nicht mehr. Außerdem reagiert das System so äußerst träge auf böse Server.

Alternativ könnte man für jedes Share einen Server mit der Überwachung beauftragen. Wenn ein Share den Server wechselt, wird der Überwachungsserver von beiden informiert. Dieser Server wüsste dann, wer für das Verschwinden eines Shares verantwortlich ist und kann eine Beschwerde publizieren. So kann das System schnell und dynamisch auf böse Server reagieren. Jedoch bedeutet ein statischer Überwachungsserver für ein Share einen wunden Punkt für Angriffe und widerspricht der Idee, Aufgaben zu verteilen.

Man hat sich bei Free Haven deshalb für ein Buddy System entschieden. Das bedeutet, dass die Shares eines Dokumentes paarweise in Beziehung stehen. Dabei verwaltet jedes Share Informationen über den Aufenthaltsort seines sogenannten Buddys. Wenn ein Share den Server wechselt, dann wird der Buddy benachrichtigt. In regelmäßigen Abständen überprüft ein Share, ob sein Buddy noch vorhanden ist. Dieser Buddy-Check kann aber nicht von einer normalen Anfrage zur Dokumentübermittlung unterschieden werden, damit der Server des Buddys nicht nur auf Buddy-Checks positiv reagiert, einem Leser aber das Dokument verweigert. Ist der Buddy verloren, dann beschwert sich das Share darüber im Netz. Das Reputation System greift und der unzuverlässige Server wird dadurch bekannt gemacht und zukünftig gemieden.

Es stellt sich die Frage, wie man die Shares zueinander in Beziehung setzt. Eine einfache und naheliegende Idee ist es, in fortlaufender Folge jeweils immer zwei Shares zu Buddys zu machen. Damit wird aber folgender Angriff ermöglicht: ein böser Server weiß mit dem Besitz eines Shares den Aufenthaltsort des Buddys. Durch Handel könnte er versuchen, in Besitz des Buddys zu gelangen. Gelingt ihm das, kann er die beiden Shares einfach ohne weiteres aufsehen verschwinden lassen. Alternativ könnte ein Gruppe verschworener Server gemeinsam versuchen, an die Buddy zu kommen, um weniger Aufmerksamkeit zu erregen. Die augenscheinlich gute Idee, dass Share Nummer i Share Nummer $i + 1$ zum Buddy hat, stellt sich bei näherer Betrachtung als noch ungünstiger heraus. Dadurch kann die Gruppe verschworener Server sich ein komplettes Share nach dem obigen Verfahren besorgen und es verschwinden lassen. Es ist ein noch ungelöstes Problem, wie man die Shares am besten zueinander in Beziehung stellt. Man glaubt aber, mit dem System auf einem richtigen Weg zu sein.

Wenn man Shares dupliziert und jeweils die Kopien zu Buddys macht, so könnte ein Share, dessen Buddy verloren scheint, sich duplizieren und für die neue Kopie einen neuen Server suchen. Das würde die Persistenz der Daten stark fördern. Allerdings hat man sich gegen diese Möglichkeit entschieden. Der Grund dafür sind die Kommunikationskanäle. Die Mixnets besitzen nicht zu vernachlässigbare Latenzzeiten, und können Daten nicht zuverlässig transportieren. Kommt es dadurch zu einer temporären Aufspaltung des Netzes, so duplizieren sich alle Shares in beiden Teilen des Netzes, die ihren Buddy im jeweils anderen Teil besitzen. All diese Kopien müssen dann auf neue Server verteilt werden, was die Netzlast plötzlich stark ansteigen lässt. Wenn das zu weiteren Aufspaltungen des Netzes führt so wächst die Anzahl der Shares exponentiell und im Grunde grenzenlos. Das System bricht dadurch zusammen.

Die Latenzzeiten der Mixnets bringen noch ein Problem mit sich. Wenn ein Share seinen Server wechselt, und seinen Buddy darüber informieren möchte, so kann es sein, dass mittlerweile dieser Buddy auch seinen Server gewechselt hat, diese Information aber noch nicht beim Share angekommen ist. Um das in den

Griff zu bekommen, werden die Notifications der Buddys an den neuen Server weitergeleitet. Die dafür nötige Information besitzt der Server bereits, da er eine Quittung des Handels mit dem neuen Server hat.

1.4.4 Neue und alte Server

Im Abschnitt 1.4.2 wurde bereits angesprochen, wie man am Besten vorgehen sollte, wenn man einen Server aus dem Servnet nehmen möchte. Man handelt langlebige Shares gegen kurzlebige und wartet das längste Ablaufdatum ab. Damit wird gesichert, dass es durch das Fehlen des Servers zu keinem Datenverlust im Servnet kommt. Optional könnte der Server einen Broadcast versenden, um sein Ableben bekannt zu geben.

Fällt ein Server einfach aus oder nimmt ihn der Betreiber ohne Bekanntgabe vom Netz, so gibt es für das restliche Servnet keine Möglichkeit, dies zuverlässig festzustellen. Grund dafür ist das Mixnet, welches keine Fehlermeldungen zurücksenden kann. Wenn also Anfragen unbeantwortet bleiben, so weiß man nicht, ob sie das Mixnet verschluckt hat, der Adressat nicht antworten möchte oder der Adressat offline ist. Um das in der Griff zu bekommen, schlägt das Free Haven Design folgendes vor: Jeder Server zählt mit, wie viele Anfragen an einen Server unbeantwortet blieben. Ab einer bestimmten Anzahl wird dieser Server als inaktiv markiert und wird bei zukünftigen Broadcasts und Suche nach Handelspartnern übergangen. Falls eine Anfrage von einem inaktiven Server ankommt, so wird dieser wieder als aktiv markiert und der Zähler zurückgesetzt. Bleibt eine solche Anfrage aber über längere Zeit aus, so kann angenommen werden, dass der Server offline ist und er kann entsprechend aus der Datenbank entfernt werden.

Eines der wichtigsten Eigenschaften des Free Haven Designs ist die Möglichkeit, neue Server reibungslos ins Servnet aufzunehmen. Wenn sich jemand entschließt, einen Free Haven Server zu betreiben, soll das möglichst einfach funktionieren. Er installiert sich einige Softwarepakete - die er eventuell aus dem Free Haven Netz erhalten hat - und kann nach einer optionalen Konfiguration online gehen. Bevor er allerdings als Teil des Servnets funktionieren kann, braucht er einen Server, der ihn dem Servnet vorstellt. Analog zu der Frage, wie ein Autor einen Server findet, der sein Dokument publiziert, kümmert sich das Design wieder nicht darum, woher er einen solchen Introducer kennt. Und wieder bestimmt jeder Betreiber selber darüber, ob sein Server neue Server ins Netz einführt, oder nicht.

Der neue Server erstellt ein asymmetrisches Schlüsselpaar und schickt sein neues Pseudonym zusammen mit einem Reply Block dem Introducer. Dieser macht ihn nun im Servnet per Broadcast bekannt und schickt ihm die Pseudonyme und Reply Blocks aller Server, die er kennt. Ab jetzt kann es zu Handelsbeziehungen zwischen dem neuen Server und dem restlichen Servnet kommen. Wenn der neue Server einem anderen Server einen Handel anbietet, so wird dieser den Neuling in seine Liste der bekannten Server eintragen.

1.5 Angriffe

1.5.1 Einstufung

Es gibt viele potentielle Angriffsmöglichkeiten auf das System. Prinzipiell kann man diese in zwei Klassen einteilen: technische und nicht-technische. Gegen die Letzteren kann man ein System kaum schützen. Im Fachjargon auch als Social Engineering bekannt geht es dabei darum, das System in der Öffentlichkeit als schlecht darzustellen, das Vertrauen darin zu untergraben, Betreiber von Servern und Benutzer zu demotivieren oder durch Gesetzesfindung Leute abzuschrecken. Vermutlich werden Regierungen oder internationalen Unternehmen dabei die treibenden Kräfte dahinter sein. Es werden auch Länder mit liberaler Gesetzgebung unter Druck geraten, da in solchen Ländern vermutlich mehr Free Haven Server betrieben werden.

Gegen all diese Angriffszenarien können Systemdesigner wenig unternehmen. Wenn das System einmal läuft, liegt seine Zukunft an der Community. Da die Geschichte allerdings gezeigt hat, dass sich revolutionäres Gedankengut aller Art nicht durch Gewalt oder Manipulation unterbinden lässt, kann man davon ausgehen, dass Free Haven Zulauf finden wird.

Im Gegensatz dazu kann man gegen technische Angriffe Abwehrmechanismen ins Design mit aufnehmen. Manche Angriffe und die Abwehr dagegen wurden bereits in den letzten Abschnitten erläutert. Um aber zu zeigen, dass das noch lange nicht alles ist, werden jetzt exemplarisch weitere Angriffe aufgezeigt, mit denen man bei Free Haven rechnet. Sofern eine Abwehr bekannt ist, wird sie erklärt.

1.5.2 Kommunikationskanal

Unterbrechung des Kanals Ein böser Knoten lässt Pakete, die er eigentlich weiter schicken soll, einfach verschwinden. Daran kann man ihn nicht hindern. Was man aber machen kann ist, jeden Knoten regelmäßigen Tests zu unterziehen, die sich von normalen Aufgaben nicht unterscheiden lassen. Versagt ein Knoten bei einem Test, wird man ihm in Zukunft nicht mehr vertrauen und meiden.

Tracerouting Ein Knoten im Mixnet kennt die IP des vorherigen und des nächsten Knoten. Falls genügend Knoten dieses Wissen miteinander teilen ist es möglich, den Weg einer Nachricht durchs Mixnet zu verfolgen. Ausgehebelt wird das allerdings, so bald ein einziger Knoten dazwischen ist, der richtig arbeitet. Da sich die Nachricht durch die Entschlüsselung der nächsten Schicht verändert und die Weitersendung zufällig verzögert wird, kann ein Beobachter keinen Zusammenhang zwischen eingehenden und ausgehenden Nachrichten finden. Je mehr Traffic durch diesen Knoten geht, desto schwerer wird es, alle Möglichkeiten zu überwachen, wodurch sich die Anonymität des Mixnets verbessert.

Angriff auf die Nachrichtengröße Die Größe einer Nachricht wird sich normal nur um eine Konstante verändern, wenn sie einen Knoten passiert, da die knoteneigenen Informationen weg fallen. Dadurch kann man eingehende und ausgehende Nachrichten miteinander in Beziehung bringen und damit Wege

durchs Mixnet verfolgen. Deshalb muss man alle Nachrichten immer gleich groß halten, in dem man kleinere Nachrichten mit zufälligen Daten auffüllt.

Replay Attack Ein Angreifer speichert eine Nachricht bevor er sie weiter-sendet. Dann überflutet er den nächsten Knoten mit Kopien dieser Nachricht. Es wird dadurch zu einem starken Anstieg des Verkehrs am Ausgang des Kno-tens kommen, wobei die Adresse des übernächsten Knotens überdurchschnittlich häufig vorkommen wird. Mit einer Kopie einer solchen Nachricht attackiert man nun den nächsten Server auf die gleiche Weise. So kann man den Weg einer Nachricht durchs Mixnet verfolgen. Abhilfe schafft hier nur eine Nummerierung der Pakete in jeder Schicht. Wenn diese Nummerierung kryptographisch gegen Veränderung gesichert ist kann ein Knoten einfach Pakete verwerfen, die die gleiche Kennung tragen.

Markiertes Überfluten Normalerweise versteckt sich eine Nachricht beim Übergang durch einen Knoten zwischen all den anderen Nachrichten, die den selben Knoten passieren. Wenn man aber einen Knoten mit Paketen überflutet, die ein eindeutig erkennbares Merkmal haben, so kann man diese am Ausgang wieder herausfiltern. Zu einer normalen Nachricht gibt es dann effektiv weniger Nachrichten, die die Erkennung verschleiern. Es ist sehr schwer, sich gegen diesen Angriff mit einfachen Möglichkeiten zu verteidigen. Wirkliche Abhilfe schafft nur die Einführung von Authentifizierungen der Knoten. Jedoch darf dabei nicht die Anonymität gefährdet werden.

Benutzerverhalten Wenn ein Angreifer das Mixnet lange Zeit überwacht, kann man bestimmte Verhaltensmuster der Teilnehmer voneinander unterschei-den. Gegen diesen Angriff mit stochastischen Mitteln bleibt nur die Warnung an die Benutzer, Dinge nicht automatisiert abzuwickeln und ihr Verhalten oft zu ändern.

Anonymität Angriffe auf die Anonymität der Autoren, Leser und Server gibt es zahlreiche. Entscheidend für deren Abwehr ist das Mixnet. Doch es gibt auch Angriffe, gegen die nichts unternommen werden kann. Als Beispiel soll hier fol-gende Überlegung dienen: ein Angreifer bietet ein Dokument mit einer MIME-kodierten URL an und nützt die Tatsache aus, dass manche Anwenderprogram-me solche Links automatisch aufrufen. Damit erwischt er die Leser, die an dem Inhalt interessiert waren und kann sie nun genauer beobachten.

1.5.3 Datenintegrität

Zeitsprünge Wenn es einem Angreifer gelingt, die Systemzeit eines Servers zu kompromittieren, dann wird das Opfer viele Shares löschen, da ihr Verfalldatum scheinbar überschritten wurde. Wenn ein Server seine Systemzeit über einen Zeitserver im Netz stellt, kann man vielleicht einen falschen Server an die Stelle des richtigen setzen oder den Zeitserver selber kompromittieren. Es wäre deshalb ratsam, wenn ein Free Haven Server Zeitsynchronisationen nur dann akzeptiert, wenn diese nicht zu große Sprünge in der Zeit machen.

physikalische Angriffe Durch den Einsatz von Rabin's Algorithmus zur Verteilung von Informationen [8] macht es der Datenintegrität im Servnet nichts aus, wenn einzelne Server physikalisch zerstört oder auf andere Art vom Netz getrennt werden. Ein anderer Angriff wäre es, wenn man versucht, Betreiber von Servern für die Verbreitung illegalen Materials zur Verantwortung zu ziehen. Hier erhofft sich das Free Haven Team, dass der Betreiber nicht verurteilt werden kann, da er bedingt durch das Design zu keinem Zeitpunkt wissen kann, welche Daten sich auf seinem Server befinden. Man hofft weiterhin, dass es gemäß der Hackerethik Leute geben wird, denen das Ziel des Free Haven Projektes wichtig genug ist, um die Risiken und Belastungen auf sich zu nehmen. Auch glaubt man nicht, dass es gelingen wird, weltweit Gesetze durchzubringen, welche Free Haven illegal machen.

Denial of Service Wenn man einem Server mit Anfragen überflutet wird er die meisten Ressourcen für deren Bearbeitung verbrauchen und kann damit seiner eigentlichen Aufgabe nicht mehr nachkommen. Die einzige Möglichkeit gegen diesen Angriff besteht darin, dass das Kommunikationsmedium einen Schutz dagegen bietet. Doch zur Zeit bieten alle in Frage kommenden Techniken keinen solchen Schutz. Das ist ein offenes und ernstes Problem.

Shares ansammeln Eine Gruppe von Servern handelt so lange, bis sie genügend Shares eines Dokumentes besitzt, um das Dokument vollständig verschwinden lassen zu können. Man erwartet, dass das Servnet groß genug sein wird, damit ein zufälliges Zusammenkommen der Shares unwahrscheinlich ist. Das bedeutet, dass die bösen Server die bereits im Besitz befindlichen Shares nicht mehr hergeben. Wenn allerdings ein regelmäßiges Handeln der Shares vorge-schrieben wird und das Buddy System die Einhaltung überwacht, kann erzwungen werden, dass die gesammelten Shares wieder hergegeben werden müssen, um überhaupt neue Shares zu erlangen.

1.5.4 Vertrauensnetzwerk

Verrat Die einfachste Art, das Vertrauensnetzwerk zu umgehen ist es, ein ordentlicher Teilnehmer des Servnets zu werden und ab einem bestimmten Grad an Vertrauen damit beginnt, Shares vor ihrem Verfallsdatum zu löschen. Das kann offensichtlich nicht verhindert werden. Man kann aber dafür sorgen, dass dieser Angriff ohne große Folgen bleibt. Dafür sollen das Buddy System und das Reputation System sorgen. Ein Share erkennt, dass sein Buddy fehlt und beschwert sich im Servnet. Dadurch verliert der böse Server mit jedem gelöschten Share an Vertrauen und bekommt so mit der Zeit keine neuen Handelsangebote mehr.

Verschmutzung Ein Angreifer kann dem Servnet beitreten, und dann richtige Daten gegen Müll tauschen. Da er die richtigen Shares wieder zurückgeben muss so lange ein freier Buddy dazu existiert, wird sich mit der Zeit ein Gleichgewicht zwischen Müll im Servnet und richtigen Daten beim Angreifer einstellen. Wenn er über genügend Kapazitäten im Vergleich zur Größe des Servnets verfügt, kann der Angreifer dieses Gleichgewicht entscheidend zu seinen Gunsten verschieben. Damit wird es immer wahrscheinlicher, dass er in den Besitz

von Shares und ihren Buddys kommt, so dass er sie ohne Aufsehen löschen kann. Das kann so weit gehen, dass auf diesem Weg unbemerkt ganze Dokumente aus dem Servnet verschwinden. Dagegen hat das Free Haven Design keine Mittel. Allerdings ist fraglich, ob hier der Nutzen den Aufwand rechtfertigt, oder ob man nicht billigere Wege findet, den selben Schaden anzurichten.

Fallen Das System aus Beschwerden, Beweisen und Vertrauen ist noch nicht wasserdicht. Ein Angreifer kann mit Absicht eine Beschwerde provozieren, auf die er mit einem scheinbaren Beweis für seine Unschuld reagiert. Damit ist der Server, der sich beschwert hat, auf einmal der Schuldige und verliert Vertrauen. Im aktuellen Design gibt es dagegen keine Abwehr. Man ist sich bewusst, dass man an dieser Stelle nacharbeiten muss.

1.6 Schlusswort

Free Haven hat sich ein großes Ziel auf die Fahnen geschrieben und sich damit starke Gegner gemacht. Es scheint unmöglich zu sein, ein System zu bauen, das den gestellten Anforderungen gerecht wird und diesen Gegnern stand hält. Es gibt noch viele Probleme zu lösen und viele Dinge zu testen. Und gegen manche Angriffe ist noch keine Abwehr bekannt. Aber die Entwickler sind sich dennoch sicher, auf dem richtigen Weg zu sein.

Im Unterschied zu anderen Diensten soll Free Haven Anonymität sowohl für Autoren als auch für Leser und Server bieten. Das ist einmalig und verursacht viele Schwierigkeiten, die aus dem Weg geräumt werden müssen. Man erwartet nicht, dass Free Haven so bekannt sein wird wie Freenet. Zudem ist das System noch weit von seiner Fertigstellung entfernt. Aber im Gegensatz zu Netzen wie Freenet, wo ungewollter Verlust der Anonymität höchstens rechtliche Schritte für einen Benutzer bedeuten kann, will sich Free Haven um diejenigen kümmern, die es sich nicht leisten können, einen Fehler zu machen.

Meine persönliche Meinung ist, dass wir Dienste wie Free Haven brauchen um Informationsfreiheit zu garantieren. Es geht mir dabei nicht nur um Staaten, in denen man um sein Leben fürchten muss, wenn man sich politisch äußert. Auch in demokratischen Ländern sehe ich in Zukunft die Notwendigkeit eines freien Informationssystems. Wenn man z.B. die Internetzensur durch die Bezirksregierung Düsseldorf [10] oder die Telekommunikations-Überwachungsverordnung (TKÜV) [11] betrachtet, müsste der Grund klar werden. Das sind Anfänge in eine Richtung, die ich nicht als gut empfinde. Deshalb muss man sich überlegen, wie man sein Recht auf informelle Selbstbestimmung durchsetzen kann.

Literaturverzeichnis

- [1] <http://www.freehaven.net>
- [2] <http://www.freehaven.net/overview.html>
- [3] <http://www.freehaven.net/people.html>
- [4] Roger Dingledine:
The Free Haven Project: Design and Deployment of an Anonymous Secure
Data Haven.
MIT Master's Thesis, June 2000.
<http://www.freehaven.net/doc/freehaven.ps>
- [5] Roger Dingledine, Michael J. Freedman, David Molnar:
The Free Haven Project: Distributed Anonymous Storage Service.
July 2000 (LNCS 2009).
<http://www.freehaven.net/doc/berk/freehaven-berk.ps>
- [6] Roger Dingledine, Nick Mathewson, Paul Syverson:
Reputation in P2P Anonymity Systems
June 2003
<http://www.freehaven.net/doc/econp2p03.pdf>
- [7] Roger Dingledine, Nick Mathewson, Paul Syverson:
Tor: The Second-Generation Onion Router
November 2003.
<http://www.freehaven.net/tor/tor-design.pdf>
- [8] Michael O. Rabin:
Efficient dispersal of information for security, load balancing, and fault tolerance
April 1989
- [9] Erneut Internet-Dissident in China verurteilt
<http://www.heise.de/newsticker/data/wst-08.12.03-001/>
- [10] Oberverwaltungsgericht bestätigt Website-Sperrung
<http://www.heise.de/newsticker/data/jk-19.03.03-002/>
- [11] Telefonüberwachung kommt immer mehr in Schwung
<http://www.heise.de/newsticker/data/anm-03.05.03-000/>