# Freehaven und Publius: Anonyme und zensurresistente Verbreitung von Informationen

Christian Hausner
christian.hausner@fau.de

**Kurzzusammenfassung**

Dieses Dokument beschreibt Systeme zur zensurresistenten und anonymen Publikation von Informationen im Web. Ziel der vorgestellten Systeme ist es selbst Angreifern, die über große Mittel verfügen, das Zensieren oder Verändern von Informationen in solchem Grade zu erschweren, dass dies nahezu ausgeschlossen werden kann. Darüber hinaus soll die Anonymität des Autors, des Lesers und jeder beteiligten Partei gewahrt bleiben.

## 1 Motivation

Die Veröffentlichung neuer Ideen ist seit jeher eine Möglichkeit dem Wunsch nach Veränderungen Ausdruck zu verleihen. Die Urheber und Verbreiter neuer Ideen haben aus vielfachen Gründen ein Interesse daran ihre Identität geheim zu halten. Die Gegner ihrer Ideen, die i.A. über weit größere Mittel verfügen, werden Versuche unternehmen die Verbreitung zu verhindern oder die enthaltenen Informationen zu ihren Gunsten zu verändern. Sollte es möglich sein den Autor oder einen Verbreiter zu ermitteln, könnte auch dieser gezwungen werden Veränderungen vorzunehmen oder die Verbreitung negativ zu beeinflussen.

Das WorldWideWeb schien anfänglich geradezu prädestiniert dafür, Informationen anonym und zensurresistent zu verbreiten. Die Benutzer erkannten jedoch sehr schnell, dass dem nicht so ist. Jeder Transfer von Informationen, jedes Bereitstellen von Dateien, jede Aktion hinterlässt Spuren in den beteiligten Systemen. Deswegen wurden Informationsverbreitungssysteme entwickelt, die (in unterschiedlichem Ausmaß) den Beteiligten Anonymität zusichern und die Dokumente vor Veränderung und Löschung geschützt bereitstellen.

## 2 Designziele

Die wichtigste Aufgabe, die ein hier beschriebenes Informations-Verbreitungssystem erfüllen muss, ist die Wahrung der Anonymität aller beteiligen Parteien:

- Autor-Anonymität: Der Urheber eines Dokuments darf weder durch den Text selbst noch durch die Übertragung zum Herausgeber ermittelt werden können.
- Herausgeber-Anonymität: Die Anonymität des Verlegers - in vielen Fällen identisch mit dem Autor - darf durch die Bereitstellung des Dokuments nicht gefährdet werden.
- Leser-Anonymität: Die Identität eines potentiellen Lesers darf nicht Gefahr laufen aufgedeckt zu werden, wenn er ein gespeichertes Dokument aus dem System abruft.
- Server-Anonymität: Das Abrufen oder Bereitstellen von Informationen darf keinen Hinweis zulassen, welche Systeme beteiligt sind, auf welchem Knoten des Netzwerks die Inhalte gespeichert werden, noch wo sich diese Knoten befinden.

- Dokument-Anonymität: Einem Server bzw. dessen Administrator darf es nicht möglich sein auf die Inhalte der von ihm gespeicherten Dokumente zu schließen.

Ein weiteres primäres Designziel besteht darin, die gespeicherten Informationen vor Zensur oder Löschung geschützt bereitzustellen. Ein Informations-Verbreitungssystem sieht sich vielfältigen Versuchen der Veränderung/Löschung durch verschiedenste Gegner ausgesetzt. Regierungen, Firmen und Individuen können ein Interesse daran haben, die Publikation bestimmter Informationen zu unterbinden. Dies soll Angreifern in einem solchen Grade erschwert werden, dass es als nahezu unmöglich angesehen werden kann.

Überdies gibt es noch eine Reihe weiterer wünschenswerter Eigenschaften, wie beispielsweise Fehlertoleranz und Dezentralität. Diese müssen jedoch hinter den primären Anforderungen an ein Informationsverbreitungssystem der hier beschriebenen Art zurückstehen. Dies gilt auch für das Bestreben ein System möglichst effizient zu gestalten; Effizienz kann vernachlässigt werden, wenn dadurch Anonymität und Zensurresistenz in erheblichem Maße erhöht werden.

# 3 Publius

Das Publius-Informations-Verbreitungssytem wurde im Jahr 2000 von Marc Waldman (NYU) in Kooperation mit Aviel D. Rubin und Lorrie Faith Cranor (AT&T Labs-Research) erdacht.

## 3.1 Systemarchitektur

Das Publius-Informations-Verbreitungssystem besteht aus einer großen Anzahl möglichst weitverteilter Webserver, über die Informationen repliziert werden. Es existiert eine (statische) systemweitverfügbare Liste dieser Server, die jedem Client bekannt sein muss.

Zur Abwicklung der Kommunikation setzt Publius auf anonyme Proxy-Server. Diese nehmen bei der Kommunikation einen Platz als dritte Partei zwischen Client und Server ein. Ein anonymer Proxy nimmt eine Anfrage eines Client entgegen, entfernt alle Informationen über deren Herkunft und führt diese selbst aus. Anschließend werden die erhaltenen Informationen zurück an den Client gesendet.

## 3.2 Benutzung

### 3.2.1 Veröffentlichung

Der Herausgeber erzeugt einen symmetrischen Schlüssel, mit dessen Hilfe er die zu veröffentlichenden Inhalte chiffriert. Dieser Schlüssel wird anschließend mittels eines speziellen Algorithmus, dem Shamir-Secret-Sharing-Algorithmus [SHA79], aufgespaltet. Es werden n Bruchstücke erzeugt, von denen k (k<n) ausreichen, um den Schlüssel wiederherzustellen (typische Werte wären: n=30, k=3). Dabei ist jedes Bruchstück gleichberechtigt und lediglich die Anzahl ist für die Rekonstruktion entscheidend. Eine zu geringe Anzahl an Bruchstücken, erlaubt keinerlei Rückschluss auf die ursprünglichen Inhalte. Jedes dieser Bruchstücke, wird mit den verschlüsselten Inhalten kombiniert und ein MD5-Hash-Wert [RIV92] darüber gebildet. Aus diesem Wert wird mittels des “Publius-Wrap”-Algorithmus der Index eines Servers aus der Publius-Server-

Freehaven und Publius.fm: 20.06.02 23:06

Liste ermittelt. Dort wird in einem Verzeichnis, das den Namen des Hash-Werts bekommt, das Bruchstück unter dem Namen ’share’ und die verschlüsselten Inhalte unter dem Namen ’file’ abgelegt.

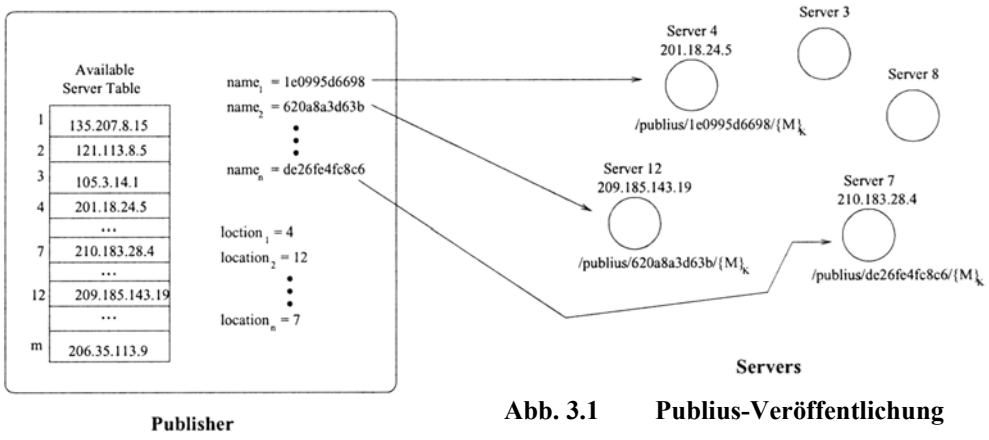


**Abb. 3.1 Publius-Veröffentlichung**

Um die Dokumente zugreifbar zu machen, wird bei der Publikation eine Publius-URL erzeugt, die aus folgenden Teilen besteht:

http://!publius!/options MD5_hash MD5_hash MD5_hash

```
http://!publius!/07 2LyMOBWJrDw=GTEaS2GlNNE=NIBsZlvUQP4=
sVfdKF7o/kl=EfUTWGQU7LX=OCk7tkhWTUe=GzWiJyio75b=QUiNhQWyUW2=
fZAX/MJnq67=y4enf3cLK/0=
```

### 3.2.2 Abrufen

Aus der Publius-URL extrahiert der Client die Hash-Werte/Verzeichnisnamen. Davon wählt er zufällig k aus und berechnet mittels der “Wrap”-Funktion den Listenindex des entsprechenden Servers. Dann lädt er k Bruchstücke des Schlüssels und eine Version der verschlüsselten Inhalte herunter. Der Schlüssel wird rekonstruiert und die Inhalte mit dessen Hilfe dechiffriert. Sollte auf den ausgewählten Servern das Schlüsselbruchstück oder die verschlüsselten Inhalte nicht verfügbar sein, bietet die Publius-URL alternative Speicherplätze, von denen ’file’ und ’share’ bezogen werden können.

### 3.2.3 Update

Das Publius-System ermöglicht dem Autor (passwortgeschützt) sein Dokument nachträglich zu verändern. Hierzu wird Datei ’password’ auf jedem der Server mitgespeichert, die einen Hash-Wert des Autor-Passworts enthält. Da Veränderungen im Dokument, bedingt durch die Art der Veröffentlichung, eine andere Publius-URL erzeugen, würde eine Anfrage niemals zum aktualisierten Dokument führen. Bei einem Update werden die Inhalte gelöscht und eine Datei ’update’ erstellt, die die neue Publius-URL enthält. Server überprüfen bei einer Anfrage das Vorhandensein einer solchen Datei und veranlassen eine Umleitung neuen URL.

Freehaven und Publius.fm: 20.06.02 23:06

### 3.2.4 Löschen

Das Löschen von Publius-Dokumenten wird ebenfalls über den Hash-Wert des Autor-Passworts kontrolliert. Übermittelt ein Autor eine Aufforderung zum Löschen, überprüft der Server, ob das Passwort zu dem gespeicherten Hash-Wert passt und entfernt ggf. das Dokument. Die Möglichkeit ein Dokument explizit löschen zu können, birgt die Gefahr, dass Angreifer Druck auf den Autor/Verleger ausüben, die Informationen zu löschen. Für Autoren/Verleger, die sich dieser Gefahr nicht aussetzen möchten, wurde eine “do not delete”-Option geschaffen die beim Veröffentlichungsprozess gesetzt werden kann.

## 3.3 Sicherheitsmechanismen

Das Veröffentlichungs-Prozedere erzeugt mehr als die benötigten Bruchstücke des Schlüssels und auf diesen vermeintlich überflüssigen Bruchstücken beruht die Fehlertoleranz und Zensurresistenz des Publius-Systems. Selbst bei Ausfall einiger Server oder Verlust der gespeicherten Schlüsselbruchstücke kann der Client genügend Fragmente empfangen, um diesen zu rekonstruieren und die Nachricht lesen zu können.

Bei Erzeugung der Bruchstücke wurden außerdem Hashwerte über das Paar aus Bruchstück und Informationen errechnet und in der URL gespeichert. Nach Erhalt einer ausreichenden Anzahl von Bruchstücken und Entschlüsselung einer Version der Informationen, berechnet der Client nochmals den Hashwert, analog zum Verfahren bei der Veröffentlichung. Sollte dieser neu berechnete Wert nicht mit dem in der URL gespeicherten Hashwert übereinstimmen wurden die Informationen verfälscht. Der Client hat dann die Möglichkeit eine andere Kombination von Schlüssel-Bruchstücken und Informationen zu versuchen. Die Anzahl der alternativen Speicherplätze ist implementierungsabhängig. Erste Versionen von Publius beschränkten sich browserbedingt auf eine maximale URL-Länge von 255 Zeichen, in die nur eine begrenzte Anzahl von Servern codiert werden konnte. Sollten trotz redundanter Speicherung keine k Fragmente des Schlüssels geholt werden können oder die korrekten Informationen auf keinem der Server abrufbar sein, gelten die Informationen als verloren.

## 3.4 Implementation

Die Kommunikation zwischen dem Client und den Publius-Servern findet mittels HTTP-Protokoll statt. Jede Publius-Operation ist an eine spezielle URL gekoppelt und die benötigten Parameter werden im Body eines HTTP-POST-Requests übertragen. Ergebnisse werden dann mittels eines Web-Browsers angezeigt. Um als Server teilzunehmen, wird ein CGI-Skript auf einem HTTP-Server gestartet, das die HTTP-POST-Anfragen des Client entgegennimmt. Als Client funktioniert ein spezieller HTTP-Proxy der transparent alle nicht Publius-Anfragen an die entsprechenden Server weiterleitet. Bei Erhalt einer Publius-Anfrage führt dieser Proxy alle nötigen Operationen durch, die zum Abrufen eines Publius-Dokuments notwendig sind.

# 4 Freehaven

Die Entwicklung des Freehaven-Systems begann 1999 als Forschungsprojekt der MIT-Studenten Roger Dingledine, Michael Freedman und David Molnar.

## 4.1 Systemarchitektur

Das Freehaven-Design besteht aus dem Publikations-System, das für die Lagerung und Bereitstellung der Dokumente verantwortlich zeichnet, und dem Kommunikations-System, das vertrauliche und anonyme Kommunikation zwischen den Beteiligten sicherstellen soll.

### 4.1.1 Kommunikationssystem

Freehaven setzt zur Realisierung der Kommunikation zwischen Servern und Clients auf die Verwendung von eMails. Diese haben im Hinblick auf Anonymität allerdings ein entscheidendes Problem: Im Mail-Header wird jeder Rechner vermerkt, den die Nachricht auf ihrem Weg vom Sender zum Empfänger passiert hat. Um die Zuordnung zu diskreten Benutzern zu verhindern, werden spezielle Remailer-Systeme eingesetzt.

Remailer entfernen die Mail-Header und senden die Nachricht (nach einer verschleiernden Latenz-Zeit) weiter zum Empfänger. Zur Erhöhung der Sicherheit wird die Übertragung in Form von Remailer-Ketten abgewickelt. Die Reihenfolge des Versands wird bereits vorher festgelegt (chaining). Der letzte Remailer der Kette kennt den Empfänger. Um zu erreichen, dass dieser den anderen verborgen bleibt, wird die Mail mit dem Public-Key des letzten Servers verschlüsselt. Wenn die Mail dort ankommt wird die Nachricht zurück in Klartext gewandelt und an die Zieladresse zugestellt. Analog dazu verfahren die übrigen Remailer der Kette, sodass eine regelrechte Kaskade von ineinander verschachtelten Verschlüsselungen entsteht.

Jeder Benutzer hat einen oder mehrere Reply-Blocks durch die eine Möglichkeit geschaffen wird, dem Benutzer eine Nachricht zu schicken ohne seine Identität zu kennen. Hierzu legt ein Client einen Weg zu sich durch das Remailer-Netzwerk fest und schachtelt diese Weginformationen, wie oben beschrieben, verschlüsselt ineinander. Soll es dem Adressaten einer eMail möglich sein eine Antwort an den Client zu verfassen, wird mit der Nachricht dessen Reply-Block übermittelt.

Freehaven benutzt implementierte Remailer-Systeme, wie MIX-Master Remailer-Networks oder Cypherpunks-Networks.[BLE00]

### 4.1.2 Servernetzwerk: servnet

Freehaven basiert auf einer Gemeinschaft von Servern, die Daten des Informationssystems bereitstellen. Eigene Daten werden anderen Servern anvertraut und im Gegenzug wird Speicherplatz für deren Inhalte angeboten. Jeder Server hat einen Reply-Block und ein Paar aus Private- und Public-Key. Darüber hinaus pflegt er eine Liste der anderen Teilnehmer des servnet, in der Reply-Blocks und deren Public-Keys gespeichert sind.

## 4.2 Benutzung von Freehaven

### 4.2.1 Veröffentlichung

Es gibt nur eine Möglichkeit Informationen im Freehaven-System zu veröffentlichen. Es muss Platz für Daten anderer Server bereitgestellt werden und im Gegenzug können die eigenen Daten an das System abgegeben werden.

Freehaven und Publius.fm: 20.06.02 23:06

Ein Autor hat zwei Möglichkeiten: entweder er wird selbst zum Verleger, d.h. er betreibt einen eigenen Freehaven-Server. Dort kann er Speicherplatz für andere Knoten des servnet bereitstellen und im Gegenzug seine Daten einbringen. Alternativ kann sich der Autor an einen Server mit einem öffentlich bekannten Reply-Block wenden, der bereit ist als 'Introducer' zu wirken.

Bei der Einspeisung in das Freehaven System werden folgende Schritte abgearbeitet:

- Die Informationen werden mittels eines Sharing-Algorithmus - Information Dispersal Algorithm [RAB89] - aufgeteilt. (n Bruchstücke, k ausreichend, um die Information wiederherzustellen)
- Der Autor erzeugt ein Schlüsselpaar aus einem Public- und einem Private-Key für dieses Dokument
- Mittels des Private-Key wird das Daten-Segment jedes Bruchstücks signiert
- Die Freehaven-Shares werden zusammengestellt

```
<share>
<PKdoc>cec41f889d75697304e89edbdddf243662d8c784</PKdoc>
<sharenum>1</sharenum>
<buddynum>0</buddynum>
<totalshares>100</totalshares>
<sufficientshares>60</sufficientshares>
<expiration>2002-07-11-22:25:24</expiration>
<data>ASCII-Daten</data>
<signature>bdf23f456999887234e89abbeeff34265438c134
</signature>
</share>
```

- Der Server tauscht die Shares dieses Dokuments dann mittels des 'Trading' (vgl.:4.3.1 Trading & Receipts) gegen Shares der anderen Server.

### 4.2.2 Abrufen

Der Public-Key des gewünschten Dokuments muss dem Leser bekannt sein, dies kann über Newsgroups oder Websites erreicht werden. Leser müssen entweder selbst einen Server betreiben oder sie kennen den Reply-Block eines Servers, der bereit ist ihre Anfrage durchzuführen.

Der Leser erstellt ein Paar aus Private-Key und Public-Key, sowie einen One-Time-Reply-Block, gültig für diese Transaktion. Der Freehaven-Server broadcastet nun eine Anfrage, die den Public-Key des gewünschten Dokuments enthält und darüber hinaus auch den Public-Key und Reply-Block des Lesers. Jeder Freehaven-Knoten, der die Anfrage erhält, prüft ob er Shares mit übereinstimmendem Public-Key hat. Bei Vorhandensein eines Share des angefragten Dokuments, verschlüsselt er dieses mit dem Public-Key des Lesers und verschickt es per eMail unter Verwendung des Reply-Blocks. Der Leser erhält nun (für ihn nicht ersichtlich woher) nach und nach Shares des gewünschten Dokuments zugesandt und kann diese, wenn genügend eintreffen, mit seinem Private-Key entschlüsseln und das Dokument wieder zusammensetzen.

### 4.2.3 Update

Das Freehaven-Design sieht aus Sicherheitsgründen keinerlei Aktualisierungsmöglichkeit für Dokumente vor. Selbst bei einer optionalen Möglichkeit Dokumente zu ändern würde eine aus Sicht der Freehaven-Designer untragbare Angriffsmöglichkeit geboten werden.

### 4.2.4 Löschen

Ein explizites Löschen ist im Hinblick auf Sicherheit ebenfalls nicht im Freehaven-System vorgesehen. Die Verweildauer von Dokumenten im System kann mittels eines Haltbarkeitsdatums gesteuert werden. Dieses setzt der Verleger bei Veröffentlichung des Dokuments fest. Server verpflichten sich nun ihre Anteile eines Dokumentes solange zu lagern bis dieses Haltbarkeitsdatum überschritten ist. Dabei ist zu beachten, dass der Wert so gewählt werden muss, dass auch ein Server gefunden wird, der sich bereit erklärt die Daten zu übernehmen.

## 4.3 Sicherheitsmechanismen

### 4.3.1 Trading & Receipts

Das servnet im Freehaven-Design ist dynamisch, d.h. Daten werden periodisch zwischen den beteiligten Servern gehandelt. Dieses Trading hat mehrere Gründe:

- Größeres Maß an Anonymität: Bittet ein Server um Abnahme eines Share, bedeutet dies nicht, dass er der Urheber der enthaltenen Information sein muss.
- Server können hinzukommen oder wegfallen: Ein Server kann hinzukommen, indem er Speicherplatz bereitstellt und anderen Servern ihre Shares abnimmt. Ein Austritt aus dem servnet kann durch Tauschen der eigenen Shares gegen kurzlebige Informationen und Warten bis zum Ablauf deren Haltbarkeitsdaten realisiert werden.
- Längere Haltbarkeitsdaten: Es wäre schwierig einen Server zu finden, der sich alleine verpflichtet ein Dokument für die nächsten Jahre bereitzustellen.
- Rücksicht auf ethische Bedenken von Server-Operatoren: Server-Operatoren die anhand der Public-Keys Dokumente innerhalb ihres Speicherplatzes entdecken, die sie nicht lagern möchten, können diese an andere Server abgeben. Organisationen können Listen mit Keys von Dokumenten, deren Veröffentlichung sie ablehnen, herausgeben und Administratoren können daraufhin ihren Datenbestand durchsuchen.
- Bewegtes Ziel: Die Sicherheit wird erhöht, weil Dokumente nie so lange am gleichen Ort liegen, dass Angreifern genügend Zeit geboten wird, die Anonymität zu kompromittieren.

Ein Server A, der Daten in das Freehaven-System einbringen möchte, wählt sich einen Server B aus und schickt ihm ein Angebot, dass eines seiner Shares enthält. Desweiteren wird die Größe und das Haltbarkeitsdatum eines Shares, das er willig ist im Austausch entgegenzunehmen, angegeben. Sollte B Interesse haben, sendet er im Austausch ein Share an den Server A. Abschließend senden sich die Server gegenseitig eine Bestätigung in Form eines Receipts zu, in der sie sich verpflichten das erhaltene Share gemäß dem Haltbarkeitsdatum bereitzustellen. Diese Quittung enthält Informationen über die beteiligten Server, die gehandelten Shares, einen Zeitstempel und eine Signatur des versendenden Servers. Eine Kopie dieser Quittungen wird jeweils an den Server (C & D) geschickt, der den Buddy des Shares speichert (siehe 4.3.3 Buddy-System).

Freehaven und Publius.fm: 20.06.02 23:06

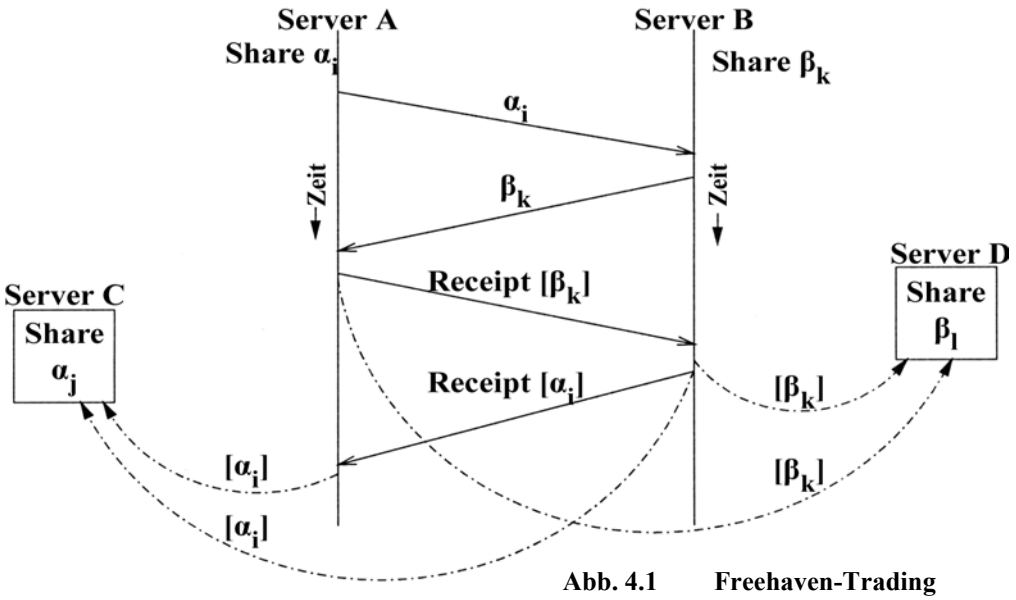


**Abb. 4.1 Freehaven-Trading**

### 4.3.2 Reputation-System

Das Vertrauen in Freehaven-Server wird über das “Reputation-System” beobachtet. Zwei Werte werden für jeden Server gepflegt:

- Reputation (Guter Ruf): Das Vertrauen in einen Server, dass er dem Freehaven-Protokoll korrekt Folge leisten wird.
- Credibility (Glaubwürdigkeit): Das Vertrauen in einen Server, dass seine Äußerungen korrekt waren.

Diese Werte werden durch Broadcasts bei erfolgreichem/fehlerhaftem Trading bzw. Verlust eines Buddies (vgl.:4.3.3 Buddy-System) angepasst. Ein mögliches Fehlverhalten können Server durch die ausgetauschten Receipts belegen. Sollten sich die Vertrauenswerte eines Servers in ein Mitglied des servnet signifikant ändern wird auch dies per Broadcast kundgetan.

### 4.3.3 Buddy-System

Um ein ehrliches Verhalten aller Server zu gewährleisten, überwachen die Server gegenseitig, dass niemand seine Daten vorzeitig löscht und somit das Dokument evtl. unwiederherstellbar wird. Bösartige Server müssen identifiziert und ausgeschlossen werden. Hierzu wurde das Buddy-System erdacht, dass eine Verbindung zwischen zwei Shares herstellt, die jeweils Informationen über den Ort des Partners enthält und pflegt. Wenn ein Share verlagert wird, erhält neben den beteiligten Servern auch der Server der den Buddy besitzt, die Receipts in denen der Handel beschrieben wird. Periodisch fragen Server an, ob die Buddies zu ihren Shares noch im Freehaven-System existieren. Sollte dies nicht mehr der Fall sein, muss eine Anomalie gemeldet und der Server, der das Bruchstück speichern sollte, bekannt gegeben werden. Somit wird den Mitgliedern des servnet ermöglicht die Vertrauenswerte für diesen Server, abhängig von deren aktuellen Werten für den sendenden und den beschuldigten Server, anzupassen

Freehaven und Publius.fm: 20.06.02 23:06

## 4.4 Implementation

Freehaven befindet sich noch in der Entwurfsphase. Es gibt Teilimplementierungen, die zur Überprüfung des Designs genutzt werden. Vor Verbreitung einer Freehaven-Implementation soll sichergestellt werden, dass der gewünschte Grad an Anonymität auch erreicht werden kann.

# 5 Vergleich

## 5.1 Systemdesign

### 5.1.1 Server-Netzwerk

Das Konzept von Freehaven und Publius bzgl. der beteiligten Server unterscheidet sich grundlegend. Freehaven setzt auf ein dynamisches Netzwerk, in dem Knoten hinzukommen oder wegfallen können. Auch Inhalte wechseln periodisch die Position, um die Angriffsmöglichkeiten zu verringern. Publius hingegen setzt auf ein statisches Netzwerk von Servern. Der Austritt eines Servers bedingt auch den Verlust eines kleinen Teils der Fehlertoleranz, da eine geringere Anzahl alternativer Speicherplätze zur Verfügung steht.

### 5.1.2 Kommunikationskanäle

Das Publius-System benutzt das HTTP-Protokoll zur Übertragung von Daten. Zur Wahrung der Anonymität der beteiligten Parteien werden Verbindungen über Anonymizer/anonyme Proxies aufgebaut. Freehaven setzt auf Kommunikation via eMail und benutzt zur Verschleierung der Identitäten bereits implementierte Remailer-Network-Systeme.

### 5.1.3 Update/Delete-Operationen

Publius bietet seinen Autoren optional die Möglichkeit Dokumente zu aktualisieren oder explizit aus dem System zu entfernen. Freehaven hingegen verzichtet vollständig auf diese Möglichkeiten, um die Sicherheit des Systems zu erhöhen und die Autoren/Verleger zu schützen.

### 5.1.4 Sicherheitsmechanismen

Beide Systeme verhindern eine unbemerkte Veränderung der Informationen. Publius speichert die Informationen verschlüsselt ab und verwendet Hashwerte, um eine Veränderung festzustellen. Freehaven benutzt den Information-Disperal-Algorithmus, um die Informationen aufzuteilen und signiert diese Datenbruchstücke mittels eines asymmetrischen Verfahrens, dessen Public-Key mit dem Share verteilt gespeichert wird.

Beide Systeme verlassen sich auf eine Verteilung über eine große Anzahl von Knoten des Netzwerks, um Datenverluste zu vermeiden. Fallen genügend Knoten aus, kann weder Freehaven noch Publius die Anfrage eines Clients erfolgreich bearbeiten. Bei typischen Werten mit n=30 und k=3 müssen 90% der Schlüssel- bzw. Datenbruchstücke verloren gehen, bis ein Abrufen durch einen Client nicht mehr möglich ist. Bei Publius bedeutet dies einen Ausfall von 90% der Server, während bei Freehaven eine geringere Ausfallquote genügen kann, da mehrere Shares eines Dokuments durch das Trading von einem einzelnen Server erworben werden können.

Freehaven bietet über die Verteilung hinaus, Mechanismen an, die fehlerhafte oder bösartige Server erkennen und von der Teilnahme ausschließen.

## 5.2 Analyse der Anonymität

Der Autor ist sowohl bei Freehaven als auch bei Publius stets anonym, sofern die Kommunikation zwischen ihm und dem Verleger über einen anonymen Kommunikationskanal abgewickelt wird und im Text keinerlei Hinweise auf die Identität hinterlassen worden sind.

Ein Publius Verleger ist ebenfalls durch den Gebrauch eines Anonymizers geschützt. Im Freehaven-Design werden Dokumente durch das Standard-Trading ins Netzwerk eingebracht, wodurch es nicht möglich ist zu sagen, ob ein anbietender Server auch der ursprüngliche Verleger oder nur ein Zwischenhändler der Informationen ist.

Die Anonymität des Lesers wird im Publius-Netzwerk nicht geschützt, dieser muss selbst für einen anonymen Kommunikationsweg sorgen. Freehaven schützt seine Clients durch die Verwendung von One-Time-Reply-Blocks innerhalb des benutzten Remailer-Netzwerks.

Publius-Server werden über eine statische Liste verwaltet und mittels der angeforderten URL lassen sich direkt die Server, die das Dokument lagern, bestimmen. Freehaven schützt seine Server mit anonymen Reply-Blocks und verbreitet Anfragen per Broadcast an alle Server.

Die Dokument-Anonymität ist bei den vorgestellten Systemen nur passiver Natur, da ein Server alleine nicht feststellen kann welche Inhalte er hostet. Durch aktive Teilnahme an den Systemen kann ein Administrator sehr wohl auf die von ihm gespeicherten Informationen schließen. Das Dokument wird im Freehaven-System geschützt, indem jeder Server zwar den Key zu Entschlüsselung besitzt, jedoch nur einen Teil der Informationen. Publius verteilt den Schlüssel, so dass ein Server alleine ebenfalls nicht beurteilen kann welche Informationen er gespeichert hat.

| Projekt | Autor | Verleger | Leser | Server | Dokument |
|---|---|---|---|---|---|
| **Publius** | eigenver-antwortlich | eigenver-antwortlich | eigenver-antwortlich | nein | passiv |
| **Freehaven** | eigenver-antwortlich | ja | ja | ja | passiv |

**Tab. 5.1 Anonymitätsvergleich**

## 5.3 Angriffs- und Abwehrmöglichkeiten

Systeme der beschriebenen Art sehen sich einer großen Zahl von Angreifern ausgesetzt, die sich in Zielen und Möglichkeiten stark unterscheiden. Neben technischen Attacken auf das System kann auch sozialer sowie rechtlicher/politischer Druck (abhängig von der lokalen Jurisdiktion) auf Beteiligte ausgeübt werden.

### 5.3.1 Zerstörung von Servern / gespeicherten Informationen

Publius und Freehaven setzen beide auf einen Verteilungsmechanismus, bei dem nur wenige Bruchstücke ausreichen, um die Informationen zu rekonstruieren. Erst nach Verlust einer sehr großen Zahl von Bruchstücken oder Servern gelten Informationen als zerstört. Gegenüber ausreichend potenten Angreifern bietet diese Verteilung allerdings keinen Schutz. Gegner dieser Art verfügen über genügend Mittel auch Knoten eines weitverteilten Servernetzwerks in ausreichender Anzahl zu zerstören. Gutsituierte Angreifer können ebenfalls genügend eigene Server in das Netzwerk einzubringen, um in der Lage zu sein die darauf gespeicherten Daten problemlos zu löschen.

### 5.3.2 Denial-of-Service

Keines der vorgestellten Informationsverbreitungssysteme sieht Möglichkeiten zum Schutz der Kommunikationswege oder der Erreichbarkeit der Server vor. Die Designer verlassen sich auf die Systeme, auf die zur Abwicklung der Kommunikation zurückgegriffen wird, bzw. auf die Administration der Server vor Ort.

### 5.3.3 Dataflodding

Publius implementiert keinerlei Mechanismen, um zu verhindern, dass ein Client Daten speichert, bis der verfügbare Speicherplatz aufgebraucht ist. Es wird über verschiedene Beschränkungen nachgedacht, die jedoch sämtlich keinen zuverlässigen Schutz bieten können.

Die Freehaven-Desginer verlassen sich auf den Schutz den der Trading-Mechanismus bietet, um zu verhindern, dass der verfügbare Speicherplatz sinnlos gefüllt wird. Hierbei muss immer eigener Speicherplatz bereitgestellt werden, ehe Informationen in das System eingebracht werden können. Dieses Mittel bietet bei ausreichend potenten Angreifern jedoch keinen Schutz vor einer Überfüllung des Systems, da diese in sehr großem Maße Speicherplatz zu Verfügung stellen und somit eigene (sinnlose) Daten einbringen können.

### 5.3.4 Angriffe auf Anonymität

Die Anonymität der Beteiligten kann durch einen geduldigen und mächtigen Angreifer kompromittiert werden. Dieser könnte intensive Beobachtungen anstellen und mittels fundierter Kenntnisse über Bandbreiten und Eigenarten des Internet Informationen über die beteiligten Knoten sammeln. Diese Kenntnisse können genutzt werden, um rechtlichen oder sozialen Druck auf beteiligte Personen auszuüben. Zwar befinden sich die Server idealerweise im Bereich vieler verschiedener Jurisdiktionen weitverteilt über den Globus, dennoch ist es nur eine Frage der Mittel, über die ein Angreifer verfügt, bis diese Verteilung keinen Schutz mehr bieten kann.

### 5.3.5 Angriffe spezielle Systemeigenschaften

Jedes der vorgestellten Systeme hat, bedingt durch das Design, spezielle Eigenschaften, die sich ein potentieller Angreifer zu Nutze machen kann.

Publius bietet mit der Möglichkeit Informationen zu aktualisieren Angriffpunkte für Manipulationen des update-Files. Angreifer können beispielsweise den Versuch unternehmen eigene ’update’-Dateien mit eigenen URLs zu erstellen. Sollte dies auf genügend Servern gelingen,

sodass ein Client, der willkürlich k Bruchstücke des Schlüssels herunterlädt, k gleiche Update-URLs erhält und somit zu den geänderten Informationen umgeleitet wird, wurde das Dokument erfolgreich zensiert.

Freehaven lässt unter anderem auch deswegen die Möglichkeit des Updates außen vor. Jedoch ist auch das Freehaven-Design nicht frei von systembedingten Ansatzpunkten für potentielle Angreifer. Server können mittels des Trading versuchen eine möglichst große Zahl von Shares anzusammeln, um diese anschließend zu vernichten, so dass das Dokument nicht wiederhergestellt werden kann. Dabei können bösartige Freehaven-Server so geschickt vorgehen, dass nicht einmal der Buddy-Mechanismus dies zu verhindern mag. Desweiteren können Server falsch Zeugnisablegen über andere Mitglieder des servnet. Somit kann das Vertrauen in diese Server geschwächt und die eigenen Server zusätzlich als Speicherplatz für Informationen favorisiert werden.

# Literaturverzeichnis


BLE00. H. Bleich, Selbstverdunkelung - Anonymes Mailen in der Praxis in c.t. 16/00, Heise Verlag, Hannover, 2000
http://www.heise.de/ct/

CRW00. L.F. Cranor, A.D. Rubin, M. Waldman, Publius: A robust, tamper-evident, censorship-resistant web publishing system, 2000.
http://cs1.cs.nyu.edu/~waldman/publius/

DFM00. R. Dingledine, M. J. Freedman, D. Molnar, The Free Haven Project: Distributed Anonymous Storage Service, 2000
http://www.freehaven.net/

DIG00. R. Digledine, The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven, 2000

ORA01. Andy Oram, et. al., Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology, O'Reilly and Associates, Sebastopol (CA), 2001

RAB89. M.O.Rabin, Efficient dispersal of information for security, load balancing, and fault tolerance, April 1989

RIV92. R. Rivest, The MD5 message digest algorithm, RFC 1321, April 1992

SHA79. A. Shamir, How to share a secret. Communications of the ACM, 22:612-613, November 1979