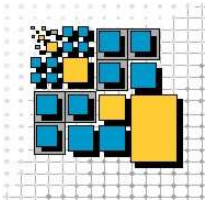

Otto-Friedrich-Universität Bamberg

Lehrstuhl für Praktische Informatik

Diplomarbeit

im Studiengang Wirtschaftspädagogik mit dem Schwerpunkt Wirtschaftsinformatik

an der Fakultät Wirtschaftsinformatik und Angewandte Informatik

Zum Thema:

Konzeption von Authentifizierungsmechanismen für anonyme Dienste und Realisierung eines ausgewählten Verfahrens im Anonymisierungsnetzwerk Tor

Vorgelegt von:

Knut Hildebrandt

Betreuer:
Karsten Loesing

Abgabedatum:
18.01.2007

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AT	Access Table
DNS	Domain-Name-System
GQ	Guillou Quisquater
HS	Hidden Service
HTTP	Hypertext Transfer Protocol
ID	Identification
I2P	Invisible Internet Project
IP	Internet Protocol
IPT	Introduction Point
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
NAT	Network Address Translator
NSD	Network Status Document
o.S.	ohne Seitenangabe
RAC	Relay_Access_Cell
RACE	Research and Development in Advanced Communication Technologies in Europe
RCC	Relay_Challenge_Cell
RCIAC	Relay_Command_Introduce_Ack_Cell
RD	Router Description
REIC	Relay_Establish_Intro_Cell
RIC	Relay_Introduce_Cell
RIEC	Relay_Intro_Established_Cell
RIPE	RACE Integrity Prmitives Evaluation
RPT	Rendezvous Point
RSA	Kryptografisches Verfahren nach Rivest, Shamir und Adleman
RSD	Rendezvous Service Descriptor
SEQDL	Signature of Equality of Discrete Logarithms
SHA	Secure Hash Algorithm
Sig	Signatur
SKDL	Signature of Knowledge of Discrete Logarithm
SKID	Secret-Key IDentification protocol
SOCKS	Sockets Secure
URL	Uniform Resource Locator
TAZ	Temporary Autonomous Zone
Tor	The Onion Routing
TS	Time Stamp
UML	Unified Modeling Language
XOR	Exclusive OR

1 Einleitung

Beim täglichen Surfen im weltweiten Netz hinterlässt jeder Nutzer bei jeder einzelnen Interaktion Spuren. Dies geschieht zum Beispiel in Form von Adressinformationen, mit der sich die Transaktionen unmittelbar auf das Individuum zurückverfolgen lassen. Von einer Privatsphäre, wie sie im täglichen Leben außerhalb des Internets beansprucht wird, kann hier keine Rede sein. Seit den ersten Bestrebungen, diese Spuren im Netz zu verwischen, sind etliche Jahre vergangen. Aktuell in der Entwicklung befindliche Anonymisierungsnetzwerke wie Tor (The Onion Routing) haben neben dem Aspekt des Schutzes der Privatsphäre eines Internetnutzers ebenfalls als Instrumente für die freie Meinungsäußerung Bedeutung erlangt und finden hier bereits weitläufig Verwendung. So nutzen beispielsweise Journalisten das Tor-Netzwerk, um sicher mit Dissidenten in China zu kommunizieren, was sonst nicht möglich wäre, ohne das Leben des Dissidenten zu bedrohen. Die in dieser Arbeit thematisierte Funktionalität der anonymen Dienste ermöglicht es weiterhin, dass Menschen ohne Angst vor einer teilweise weitreichenden Zensur oder Strafverfolgung Informationen der Öffentlichkeit zugänglich machen können. Aufgrund der Tor-Infrastruktur laufen diese Personen nicht Gefahr, dass später festgestellt werden kann, wer die Inhalte veröffentlicht hat bzw. dass der Standort des Dienstanbieters ausfindig gemacht und attackiert werden kann. Dass solche Infrastrukturen auch für die Verbreitung von rechtswidrigen Inhalten benutzt werden können, steht außer Frage und hat erst Anfang September 2006 beeindruckende Aktualität erlangt, als in Deutschland mehrere Server des Anonymisierungsnetzwerkes Tor von der Staatsanwaltschaft Konstanz wegen des Verdachtes der Verbreitung von kinderpornografischen Materials beschlagnahmt wurden.

Ziel dieser Arbeit ist es, Authentifizierungsmechanismen mit der angesprochenen Möglichkeit, in Anonymisierungsnetzwerken anonyme Diensten anzubieten, zu verbinden. Für den Kontext dieser Arbeit soll ein Anwendungsszenario aufgezeigt werden, das die Zielsetzung verdeutlichen soll. Hier könnte der Betrieb eines anonymen Dienstes in Form eines Blogs oder einer Wiki-Site als beispielhafte Anwendungsdomäne dienen. Bei dieser Kommunikationsmöglichkeit soll es angemeldeten Nutzern möglich sein, anonym Beiträge zu veröffentlichen und zu lesen. Des Weiteren sollen die Anbieter dieser Dienste in der Lage sein, eigene Identifizierungsmerkmale, hier insbesondere die IP-Adresse (Internet Protocol), die Rückschlüsse auf den Dienstbetreiber liefern können, zu verbergen. Dies ist notwendig, um den Dienstanbieter vor allem vor Maßnahmen der Zensur und vor strafrechtlicher Verfolgung zu schützen. Eine Nutzerauthentifizierung für den Dienstanbieter soll sicherstellen, dass lediglich zugangsberechtigte Personen den Service erreichen können. Eine Authentifizierung soll also nicht erst beim Dienstanbieter direkt, sondern schon auf Netzwerkebene erfolgen. Vom Dienstbetreiber nicht autorisierten Nutzern ist es durch diese frühe Überprüfung der Berechtigung erst gar nicht möglich, eine Verbindung zu dem anonymen Dienst aufzubauen. Bei bis dato vorhandenen Realisierungen zu anonymen Diensten (vgl. Hidden-Services [11] oder episites [8]) sind Nutzerauthentifizierungen bereits auf Netzwerkebene zwar geplant, eine konkrete Spezifikation oder Umsetzung fehlt jedoch bei allen Ansätzen. Diese Arbeit soll diesbezüglich eine Lücke füllen, indem verschiedene Authentifizierungsverfahren auf ihre mögliche Einsetzbarkeit bei anonymen Diensten näher untersucht werden und ein geeignetes Verfahren prototypisch umgesetzt wird.

Eine Authentifizierung von Nutzern bereits auf Netzwerkebene durch entsprechende Gateways bietet verschiedene Vorteile gegenüber einer direkten Authentifizierung durch den

Dienst selber, insbesondere in Bezug auf Angriffe auf die Anonymität (vgl. [12]) bzw. die Erreichbarkeit von Diensten (sog. denial-of-service-attacks). So konnten Syverson et al. zeigen (vgl. [12]), dass durch ein Analysieren des Netzwerkverkehrs die Anonymität des Dienstbetreibers dahingehend aufgehoben werden konnte, dass die IP-Adresse des Dienstes ermittelt werden konnte. Dies war bei verschiedenen Angriffsszenarien innerhalb weniger Stunden möglich. Da Internet-Dienste wie Blogs langfristig anonym im Internet verfügbar sein sollen, sind Vorkehrungen gegen diese Art von Angriffen zu treffen. Eine Nutzerauthentifizierung auf Netzwerkebene bietet hier die Möglichkeit, dass bereits an einem Gateway alle nicht-berechtigten Anfragen abgewiesen werden (entsprechende Router im Netzwerk übernehmen die Funktion einer Art Firewall im Netzwerk), was Verkehrsanalysen von Angreifern, die sich nicht authentifizieren können, ausschließt. Da für Verkehrsanalysen eine Vielzahl von Anfragen an den Dienst notwendig ist, die anschließend durch einen Angreifer korreliert werden, kann ein Dienstanbieter durch diese Konzentration von Anfragen *eines* Nutzers auch potentielle Angriffe von authentifizierten Nutzern erkennen und diese von der Nutzung des Dienstes ausschließen.

Zur Bearbeitung der vorgenannten Problemstellung ist diese Arbeit wie folgt gegliedert: Zunächst sollen grundlegende theoretische Konzepte erörtert werden, um eine gemeinsame Begriffsbasis zu schaffen. Insbesondere soll auf Anonymität, Authentifizierung und Autorisierung näher eingegangen werden. Anschließend wird ein Überblick über zur Verfügung stehende Prinzipien und Verfahren der Anonymisierung von Kommunikationsbeziehungen gegeben, ohne jedoch detailliert auf spezielle Umsetzungen einzugehen. Diese Übersicht soll in einem Vergleich von verschiedenen Systemen, die anonyme Dienste bereitstellen können, münden (vgl. Rewebber [6], Tor [11], I2P [8] und Tarzan [13]). Auch soll dargelegt werden, welche grundlegenden Techniken und Architekturen vorhanden sein müssen (vgl. [5], [14]), um bestehende Anonymisierungsnetzwerke so zu erweitern, dass sie die Bereitstellung von anonymisierten Diensten unterstützen. Verschiedene Authentifizierungsverfahren, wie einfache Passwortverfahren (vgl. [15] S. 388), Challenge-Response-Verfahren (vgl. [15] S. 397), Zero-Knowledge-Verfahren (vgl. [16] S. 186), Gruppen-Signaturen [17] und Ring-Signaturen [10], sollen des Weiteren dahingehend untersucht werden, inwieweit sie den Bedürfnissen der Nutzer und Anbieter von anonymen Diensten entsprechen und auch die aufgezeigten Vorteile einer Authentifizierung bereits auf Netzwerkebene bieten. Dazu wird zunächst ein Kriterienkatalog aufgestellt, anhand dessen eine Bewertung der Verfahren stattfinden soll. Ein Authentifizierungs-Verfahren, das im Tor-Netzwerk umgesetzt werden soll, wird daraufhin ausgewählt. Diese Umsetzung in Form einer Implementierung wird in Kapitel 6 dokumentiert. Abschließen soll diese wissenschaftliche Arbeit mit einem Fazit sowie einem Ausblick auf weiterführende Themengebiete, die mit der Fragestellung dieser Arbeit zusammenhängen.

2 Theoretische Grundlagen

In diesem Kapitel sollen wichtige theoretische Konzepte grundlegend erörtert und definiert werden, um eine gemeinsame Begriffsbasis zu schaffen. Zunächst sollen im Unterkapitel 2.1 auf Anonymität und Pseudonymität beschrieben werden und verschiedene Anonymitätsklassen unterschieden werden. Im darauf folgenden Unterkapitel 2.2 sollen die Begriffe Authentifizierung und Autorisierung zunächst definiert und anschließend voneinander abgegrenzt werden, da diese oft fälschlicherweise synonym verwendet werden. Dies ist wichtig, weil sich eine korrekte Verwendung als elementar in diesem Arbeitskontext herausgestellt hat. Da Anonymisierungstechniken, die in der Arbeit Anwendung finden, zum Großteil auf kryptografischen Primitiven beruhen, wird in diesem Grundlagenkapitel zum Abschluss auf diese näher eingegangen.

2.1 Anonymität, Pseudonymität, Unverknüpfbarkeit, Unbeobachtbarkeit

Bevor im nächsten Kapitel auf verschiedene Techniken eingegangen wird, mit denen Anonymität erzielt werden kann, soll in diesem Abschnitt zunächst erläutert werden, was unter Anonymität konkret verstanden werden soll. Hierbei soll insbesondere auf die von Pfitzmann und Köhntopp vorgeschlagene Terminologie zurückgegriffen werden (vgl. [1]).

Anonymität

In den Vorschlägen zur ISO/IEC 15408 (vgl. [18] S. 167) wird die Anonymität zunächst lediglich auf das Verbergen der Identität eines Nutzers bezogen:

„Anonymity ensures that a subject may use a resource or service without disclosing its user identity.“ ([18] S. 168).

Für den Rahmen dieser wissenschaftlichen Arbeit scheint die genannte Definition aus dem internationalen Standard zu begrenzt, da lediglich Zugriffe von Subjekten auf Ressourcen betrachtet werden. Andere Aspekte dieser Arbeit, beispielsweise das Anbieten von anonymen Diensten oder das Verbergen von Kommunikationsbeziehungen, können durch diese enge Sichtweise nicht abgebildet werden.

Sowohl Pfitzmann und Köhntopp als auch Demuth (vgl. [1] S. 6 und [3] S. 38 f.) fassen daher den Begriff der Anonymität weiter. Die erstgenannten Autoren sind der Auffassung, dass eine essentielle Voraussetzung, um Anonymität zu ermöglichen, das Vorhandensein einer geeigneten Umgebung mit den gleichen Attributen darstellt. Diese sogenannte Anonymitätsgruppe umfasst alle Subjekte, die im Hinblick auf die zu verbergenden Aktionen als geeignet erscheinen, d.h. die eine entsprechende Aktion potentiell ausführen können (z.B. Senden oder Empfangen einer Nachricht). Diese Anonymitätsgruppe variiert im Laufe der Zeit, lässt sich jedoch nicht erweitern, sondern lediglich einschränken. Ein Angreifer kann nämlich eindeutig erkennen, welche Elemente zu einer Anonymitätsgruppe später hinzugekommen sind und diese folglich ausschließen. Zusätzlich stellt Demuth fest, dass eine triviale Voraussetzung für das Vorhandensein von Anonymität ebenso sein muss, dass die Anzahl der Instanzen, die bei einem Ereignis vorhanden sind, größer sein muss

als die Anzahl der notwendigen Instanzen, um das Beobachtungsereignis (z.B. Senden einer Nachricht) auszulösen, da sich sonst die auslösende Instanz eindeutig identifizieren lässt. Somit sind die Anforderungen, die an eine Anonymitätsgruppe gestellt werden, eindeutig festgelegt. Auf diesen Annahmen beruhend soll im weiteren Verlauf der Arbeit in Anlehnung an Pfitzmann und Köhntopp die folgende Definition für Anonymität gelten:

„Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.“ ([1] S. 6).

Bei gleichbleibenden sonstigen Variablen kann angenommen werden, dass die Anonymität umso stärker ist, je größer die Anonymitätsgruppe ist. Die Stärke der Anonymität wird auch durch die Verteilung der Aktionswahrscheinlichkeiten der einzelnen Subjekte der Anonymitätsgruppe beeinflusst. Generell kann angenommen werden, dass die Anonymität umso stärker ist, je eher die Verteilung einer Gleichverteilung nahe kommt. Der konkreten Einfluss dieser beiden Faktoren auf die Stärke der Anonymität soll im Abschnitt zur „Quantifizierung der Anonymität“ näher erläutert werden. Es ist somit ersichtlich, dass das Konzept der Anonymität sehr stark von Umgebungsvariablen (Verteilungen, Attributen, Subjekten, Änderungen über den Zeitverlauf etc.) abhängig ist und diese Umgebungsvariablen bei einer Analyse der Anonymität auch Berücksichtigung finden müssen (vgl. [3] S. 38 f. und [1] S. 6).

Pseudonymität

Ein Element agiert dann pseudonym, wenn es bei einem Ereignis statt eines originären Identifizierungsmerkmals ein Pseudonym als Identifizierungsmerkmal verwendet. Pfitzmann und Köhntopp sprechen in diesem Zusammenhang dann von Pseudonymität, wenn Pseudonyme zur Identifizierung verwendet werden (vgl. [1] S. 14). Voraussetzung ist bei Pseudonymität jedoch nicht, dass der Gebrauch eines Pseudonyms zur Folge hat, dass der Nutzer zur Rechenschaft gezogen werden kann, so wie dies in der ISO/IEC 15408 gefordert wird (vgl. [18] S. 169 f.), ein Pseudonym also aufgedeckt werden kann.

Pseudonymität allein sagt zunächst noch nichts über den Grad der Anonymität aus. Zwei andere Eigenschaften der Pseudonymität sind hier weiter zu betrachten und bei einer späteren Analyse des Anonymitätsgrades mit zu berücksichtigen. Zunächst ist zu unterscheiden, ob und inwieweit die Zuordnung zwischen Pseudonym und Besitzer bekannt ist. Diese Zuordnung kann öffentlich sein, anfänglich nicht-öffentlich und anfänglich nicht-zugeordnet. Des Weiteren ist der Grad an Anonymität auch davon abhängig, in welchem Kontext das Pseudonym Verwendung findet und entsprechend, zwischen welchen Transaktionen Querbeziehungen hergestellt werden können. Abbildung 1 gibt eine Übersicht über verschiedene Kontexte, in denen Pseudonyme verwendet werden können und welche Abhängigkeiten zur Anonymität bestehen.

Wie aus der Abbildung ersichtlich wird, nimmt die Möglichkeit zur Verknüpfung von verschiedenen Transaktionen, die ein Subjekt tätigt, nach unten hin ab. Damit nimmt ebenfalls die erreichbare Anonymität zu. Während bei einem *person pseudonym* ein Pseudonym in allen Transaktionen des jeweiligen Subjektes benutzt wird und ein Substitut des Namens darstellt, wird das *transaction pseudonym* lediglich bei einer Transaktion benutzt und danach verworfen. Zwischen diesen beiden Extremen am oberen bzw. unteren Ende der Skala sind in der Mitte noch drei Stufen angesiedelt, die im begrenzten Umfang

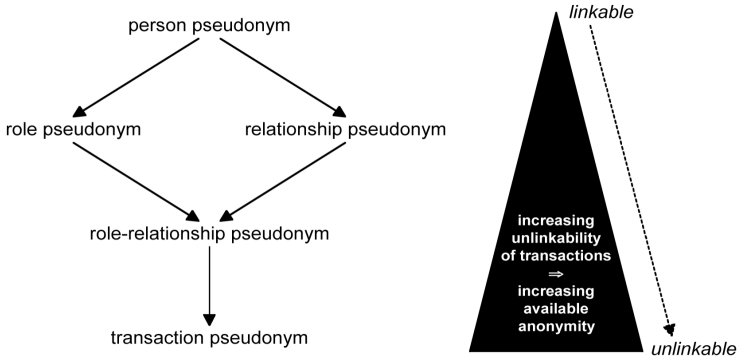

Abbildung 1: Abhängigkeit der Anonymität vom Kontext eines Pseudonyms (aus: [1] S. 20)

eine Verknüpfung von Transaktionen zulassen. Beim *role pseudonym* ist die Verwendung des Pseudonyms an eine bestimmte Rolle, die ein Subjekt in einer Transaktionen einnimmt (z.B. Käufer), gebunden. Bei verschiedenen Kommunikationspartnern wird u.U. das gleiche Pseudonym verwendet. Dies ist bei einem *relationship pseudonym* anders: Hier wird bei bei jedem Kommunikationspartner ein anderes Pseudonym verwendet. Tritt ein Subjekt in unterschiedlichen Rollen auf (z.B. als Käufer und Verkäufer), wird jedoch das gleiche Pseudonym verwendet. Bei einem *role-relationship pseudonym* wird sowohl bei Wechsel des Kommunikationspartners als auch beim Wechsel der Rolle, in dem ein Subjekt auftritt, das Pseudonym geändert.

Unverknüpfbarkeit

Während in den Vorschlägen zur ISO/IEC 15408 (vgl. [18] S. 173) die Unverknüpfbarkeit wieder lediglich auf Nutzeraktionen bezogen wird und es entsprechend Nutzern bzw. Subjekten innerhalb eines Systems nicht möglich sein soll zu unterscheiden, ob verschiedene Operationen von ein und dem selben Nutzer ausgeführt wurden, soll die in dieser Arbeit geltende Definition für Unverknüpfbarkeit von Elementen (Subjekte oder Objekte) weiter gefasst werden. Unverknüpfbar sind Elemente genau dann, wenn die Wahrscheinlichkeit, dass zwischen zwei Elementen eine Beziehung besteht, für einen Angreifer vor und nach einer Operation (z.B. Versenden von zwei Nachrichten) genau gleich ist. Das a-priori Wissen des Angreifers entspricht genau dem a-posteriori Wissen, durch die Operation lernt der Angreifer nichts hinzu (vgl. [1] S. 8).

Unbeobachtbarkeit

Während die Unverknüpfbarkeit das Verhältnis von verschiedenen Elementen zueinander beschreibt und dieses Verhältnis verbirgt, wird bei der Unbeobachtbarkeit das Element an sich verborgen. Ein Element ist dann unbeobachtbar, wenn es von anderen Elementen nicht unterschieden werden kann. Bezogen auf den Nachrichtenversand können Nachrichten nicht von sonstigem zufälligen Rauschen auf einem Kanal unterschieden werden. Einem Angreifer ist es also nicht möglich, ein Element zu extrahieren und zu erkennen (vgl. [1] S. 10 f.).

Anonymitätsklassen

Ausgehend von den vorherigen Betrachtungen zur Anonymität und Pseudonymität können anhand der Menge an Meta-Informationen (Informationen zu den Beteiligten), die bei Transaktionen preisgegeben werden, verschiedene Abstufungen für die Anonymität gemacht werden. Sowohl Flinn und Maurer als auch Goldberg (vgl. [19] S. 35 ff., vgl. [2] S. 39–48) haben Anonymitätsklassen für Transaktionen entwickelt, die größtenteils Übereinstimmungen, teilweise jedoch auch Differenzen aufweisen. Im Folgenden sollen beide Systematisierungen zu einem konsistenten System zusammengefasst werden. Die Abbildung 2 bzw. die Tabelle 1 geben zunächst einen Überblick über die verschiedenen Anonymitätsklassen der beiden Systematisierungen:

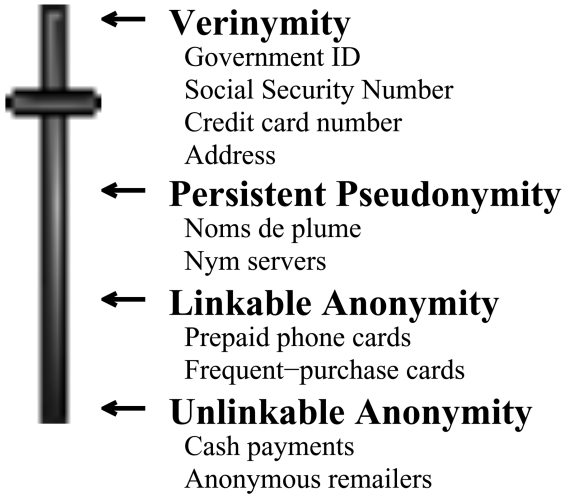

Abbildung 2: Nymity-Slider (aus: [2] S. 41)

Verinymität bedeutet, dass eine bestimmte Information ein Element aus einer Menge von möglichen Kandidaten eindeutig gegenüber allen anderen abgrenzt. Bei Verinymität sind verschiedene Transaktionen eines Subjekts verknüpfbar, und verwendete Identifizierungsmerkmale sind persistent. So ist der Name einer Person persistent und nicht beliebig änderbar. Werden Änderungen vorgenommen, sind diese nachvollziehbar (vgl. [2] S. 41 ff.). Flinn und Maurer unterscheiden in dieser Klasse der Anonymität noch einmal zwischen Super-Identifikation und Identifikation, während Goldberg diese Unterscheidung nicht trifft. Bei der Super-Identifikation ist es nicht möglich, ein Identifikationsmerkmal zu fälschen oder nachzumachen, während dies bei der normalen Identifikation möglich ist. Super-Identifikation kann beispielsweise durch persönliche Identifikation oder biometrische Merkmale erfolgen (vgl. [19] S. 36 f.).

Am unteren Ende der Skala befindet sich die (nicht verknüpfbare) Anonymität. Transaktionen lassen keinen Rückschluss auf die Identität eines Elementes zu, es werden keinerlei Identifikationsmerkmale bzw. Authentifizierungsinformationen preisgegeben. Nicht verknüpfbar bedeutet in diesem Zusammenhang, dass von zwei Transaktionen nicht bestimmt werden kann, ob diese von ein und dem selbem Element durchgeführt wurden. Hierbei ist anzumerken, dass in Abhängigkeit von verschiedenen Systemparametern (Verteilungen etc.), trotz Vorherrschen von nicht verknüpfbarer Anonymität, verschiedenen Ausmaße an Anonymität erreicht werden können. Hierauf wird im Abschnitt „Quantifizierung der Anonymität“ näher eingegangen.

Anonymitätslevel	Beschreibung
Level 5	Super-Identifizierung
Level 4	Identifizierung
Level 3	Kontrollierte Pseudonymität
Level 2	Unkontrollierte Pseudonymität
Level 1	Anonyme Identifizierung
Level 0	Anonymität

Tabelle 1: Anonymitätslevel nach Flinn und Maurer

Zwischen diesen beiden Extremen befinden sich auf der Skala von Goldberg zwei und auf der von Flinn und Maurer drei Zwischenstufen. Bei dem Level unterhalb der Verinymität spricht Goldberg von persistenter Pseudonymität. Bei persistenter Pseudonymität ist eine eindeutige Identifizierung nicht möglich (vorausgesetzt, die Zuordnung zwischen Pseudonym und Besitzer ist nicht-öffentlich). Verschiedene Transaktionen, die unter einem Pseudonym getätigt wurden, können jedoch u.U. verknüpft werden. Der Grad der Verknüpfbarkeit und damit auch die Anonymität ist, wie in Abbildung 1 bereits verdeutlicht, abhängig vom Kontext, in dem das Pseudonym verwendet wird. Eine Persistenz, wie bei der Verinymität, geht durch das beliebige Ändern von Pseudonymen und das Benutzen von mehreren Pseudonymen gleichzeitig verloren (vgl. [2] S. 43 ff.). Flinn und Maurer unterscheiden bei der persistenten Pseudonymität oder kurz Pseudonymität, ob Pseudonyme unter bestimmten Umständen von einer Kontrollinstanz aufgedeckt werden können (kontrollierte Pseudonymität) oder ob dies nicht möglich ist und die Identität eines Nutzers beispielsweise geschützt ist (unkontrollierte Pseudonymität) (vgl. [19] S. 38).

Oberhalb der (unverknüpfbare) Anonymität spricht Goldberg von verknüpfbarer Anonymität, da eine Transaktion zwar keine Identifizierungsmerkmale preisgibt, aber verschiedene Transaktionen in Beziehung zueinander gesetzt werden können. Der Unterschied zur vorherigen Stufe besteht darin, dass unterschiedliche Elemente Transaktionen ausführen können, die dann verkettet werden. Ein Beispiel ist der Kauf einer Telefonkarte: Die Identität des Benutzers wird zu keinem Zeitpunkt preisgegeben, verschiedene Anrufe können in Verbindung gesetzt werden, jedoch kann nicht sichergestellt werden, dass immer die gleiche Identität die Karte benutzt hat (vgl. [2] S. 45 ff.). Flinn und Maurer sprechen von anonymer Identifizierung (vgl. [19] S. 38).

Quantifizierung der Anonymität

Während im vorherigen Abschnitt auf unterschiedliche Anonymitätsklassen eingegangen wurde, soll in diesem Abschnitt ein Modell zur Messung des Grades an Anonymität, der innerhalb eines Systems erreicht werden kann, vorgestellt werden.

Berthold et al. (vgl. [20] S. 3) machen den Grad der Anonymität zunächst lediglich von der Größe der Anonymitätsgruppe abhängig. Das Vorhandensein von nicht-verknüpfbarer Anonymität stellt jedoch in diesem Zusammenhang eine notwendige Anforderung an das System dar. Sie gehen in ihren Arbeiten zur Anonymität in Mixnetzwerken von einer Gleichverteilung von möglichen Interaktionswahrscheinlichkeiten der Systemteilnehmer

aus. Somit gilt für die Anonymität (A):

$$A_{Berthold} = log_2(N) \tag{1}$$

wobei N die Anzahl der Systemteilnehmer angibt. Die Anonymität nach Berthold nimmt somit die Entropie von N bei Gleichverteilung als Maßzahl an. Die Entropie wird in der Informationstheorie als die Menge an Informationen, die in einer Nachricht stecken, definiert. Sie gibt die minimale Anzahl an Bits an, die zum Darstellen aller möglichen Ausprägungen der Nachricht (hier: N) benötigt werden (vgl. [9] S. 233 f. und [21] S. 51). Díaz et al. (vgl. [22] S. 4 ff.) gehen bei der Betrachtung differenzierter vor. Es ist anzunehmen, dass der Idealfall der Gleichverteilung der Interaktionswahrscheinlichkeiten nicht vorliegt und ein Angreifer durch verschiedene Angriffe (im Kommunikationskontext durch Verkehrsanalysen und Angriffe auf die Zeitmuster und die Länge von Nachrichten) den Subjekten / Objekten individuelle Wahrscheinlichkeiten zuordnen kann. Außerdem ist zu beachten, dass lediglich die Subjekte / Objekte in die Betrachtung mit einfließen dürfen, die nicht mit dem Angreifer zusammenarbeiten. Die Entropie ist in diesem Fall ein geeignetes Maß, um die Information, die in einer Verteilung beinhaltet ist, zu messen. $H(X)$ stellt also die Entropie der einzelnen Wahrscheinlichkeiten (p_i) von einem System, nachdem ein Angriff auf das System stattgefunden hat, dar:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{2}$$

Das Maß an Anonymität eines Systems wird somit wie folgt bestimmt:

$$A_{Diaz} = \frac{H(X)}{H_M} \tag{3}$$

wobei $H_M = \log_2(N)$ die maximale Entropie des Systems bei Gleichverteilung der Interaktionswahrscheinlichkeiten angibt. Dieses Anonymitätsmaß nach Díaz wird also normalisiert und der Aspekt der Größe einer Anonymitätsgruppe (N) bleibt dadurch unberücksichtigt. Lediglich die Verteilung der Interaktionswahrscheinlichkeiten bestimmt das Ausmaß der erreichten Anonymität. Aus der obigen Formel folgt, dass für A_{Diaz} gilt: $0 \leq A_{Diaz} \leq 1$:

- $A_{Diaz} = 0$ gilt, wenn eine Aktion in einem System einem Systemteilnehmer mit einer Wahrscheinlichkeit von 1 eindeutig zuordenbar ist.
- $A_{Diaz} = 1$ gilt, wenn jeder Systemteilnehmer die Aktion mit der gleichen Wahrscheinlichkeit initiiert haben kann ($p_i = 1/N$).

Anonymität im Kontext von Kommunikationsbeziehungen

Bezogen auf den Kontext von Kommunikationsbeziehungen, der auch dieser Arbeit zugrunde liegt, lässt sich das allgemeine Modell der Anonymität noch näher spezifizieren:

Zunächst ist zwischen unidirektionaler und bidirektionaler Kommunikation zwischen zwei Kommunikationspartnern zu unterscheiden. Während bei der unidirektionalen Kommunikation lediglich eine Nachricht gesendet wird, ohne dass eine Antwort erwartet wird, ist dies bei der bidirektionalen Kommunikation nicht der Fall. Der Empfänger der ersten

Nachricht antwortet seinerseits auf die empfangene Nachricht und wird somit zum Sender, die Rollen der Partner werden also getauscht. Generell kann sowohl bei unidirektionaler als auch bei bidirektionaler Kommunikation zwischen unilateraler Sender- bzw. Empfängeranonymität und bilateraler Anonymität unterschieden werden. Bei einem Vorliegen von Senderanonymität, die z.B. durch zwischengeschaltete Proxies erzielt werden kann, ist es dem Empfänger einer Nachricht nicht möglich, den entsprechenden Sender zu identifizieren. Bei einer bidirektionalen Kommunikation wird die Identität des Senders durch ein verwendetes Pseudonym dem Empfänger gegenüber verborgen, um ein Antworten auf die Nachricht zu ermöglichen. Wenn nicht-transaktionsbezogene Pseudonyme Verwendung finden, tritt der Sender nur pseudonym auf. Bezüglich des Sendens *einer* Nachricht handelt der Sender in jedem Fall anonym (vgl. [3] S. 41 f.). Pfitzmann und Köhntopp verstehen unter Senderanonymität, dass eine Nachricht keinem Sender zuordenbar ist und einem bestimmten Sender keinerlei Nachrichten zugeordnet werden können (vgl. [1] S. 9).

Bei Empfängeranonymität ist dies umgekehrt: Während der Empfänger der Nachricht den Sender eindeutig identifizieren kann, ist eine eindeutige Identifizierung des Empfängers einer Nachricht durch den Sender nicht möglich. Der Sender kommuniziert lediglich z.B. mit einem Proxy, der ein vom Empfänger verwendetes Pseudonym in die reale Adresse umsetzt. Dieser Vorgang bleibt dem Sender der Nachricht verborgen (vgl. [3] S. 42 f.). Pfitzmann und Köhntopp definieren Senderanonymität gleichbedeutend wieder über die Verknüpfbarkeit von Nachricht und Empfänger einer Nachricht (vgl. [1] S. 9).

Bei bidirektionaler Kommunikation kann eine unilaterale Anonymität, sei es Empfänger- oder Senderanonymität, lediglich für einen Kommunikationspartner aufrecht erhalten werden, da bereits mit der ersten Nachricht der Sender bzw. der Empfänger (Vorliegen von Senderanonymität bzw. Empfängeranonymität) die Identität des Kommunikationspartners erfährt. Bilaterale Anonymität bedeutet, dass weder Empfänger noch Sender einer Nachricht den entsprechenden Kommunikationspartner identifizieren können. Bei unidirektionaler Kommunikation ist dies sehr einfach möglich, indem der Sender bei vorliegender Empfängeranonymität auf das Mitschicken von Identifizierungsmerkmalen (z.B. Adresse) verzichtet. Falls nicht-transaktionsbezogene Pseudonyme verwendet werden, kann bei bidirektionaler Kommunikation nur bezüglich der Vorgänge „Senden *einer* Nachricht“ und „Empfangen *einer* Nachricht“ bilaterale Anonymität erzielt werden, da die Teilnehmer auf Grund der Erfordernis, auf Nachrichten zu antworten, lediglich pseudonym auftreten können und entsprechende Pseudonyme mit übermitteln (vgl. [3] S. 43 ff.). Abbildung 3 verdeutlicht die bidirektionale Kommunikationsbeziehung mit bilateraler Anonymität.

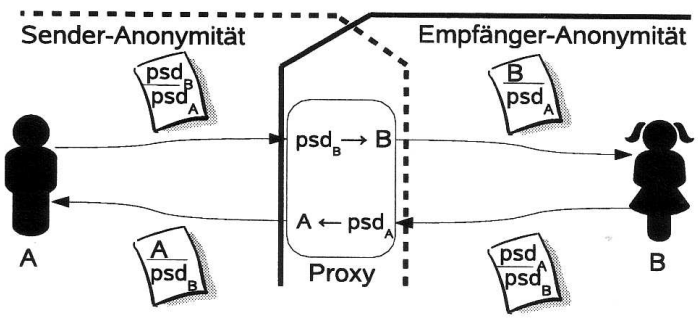

Abbildung 3: Bidirektionale Kommunikationsbeziehung mit Sender- und Empfängeranonymität (aus: [3] S. 45)

Neben Sender- und Empfängeranonymität, die nach Pfitzmann und Köhntopp die Unverknüpfbarkeit zwischen Nachrichten und Kommunikationspartnern beschreibt, lässt sich eine Verbindungsanonymität definieren, die es ausschließt, dass bestimmt werden kann, welche Kommunikationspartner miteinander interagieren. Es lassen sich also Sender und Empfänger einer Nachricht nicht in Beziehung setzen (vgl. [1] S. 9 f.).

2.2 Zugangskontrolle, Identifizierung, Authentifizierung, Autorisierung

In diesem Unterkapitel soll nun näher auf den Bereich der Zugangskontrolle eingegangen werden. Beim kontrollierten Zugriff auf Ressourcen erfolgt die Zugangskontrolle in einem zweistufigen Prozess: Zunächst geschieht die Identifizierung und Verifikation (Authentifizierung) von autorisierten Subjekten, bevor eine Autorisierung des Zugriffs erfolgen kann. Im Kommunikationskontext wird Autorisierung und Zugangskontrolle (access control) oft synonym verwendet, was zu unklaren definitorischen Abgrenzungen führen kann. Autorisierung und Zugangskontrolle sollen in dieser wissenschaftlichen Arbeit daher nicht synonym verwendet werden. Neben der Identifizierung, Authentifizierung und Autorisierung stellt die sog. Protokollierung (Accounting) einen Bestandteil der Zugangskontrolle dar. Hierbei werden alle Interaktionen, die bei einem Zugriff auf eine Ressource veranlasst wurden, protokolliert. Auf diesen Teilbereich soll hier nicht näher eingegangen werden.

Identifizierung und Authentifizierung

Woo und Lam zufolge (vgl. [23] S. 43) gibt es drei unterschiedliche Authentifizierungstypen, die zunächst zu unterscheiden sind:

- message content authentication – verifiziert, dass gesendeter und empfangener Nachrichteninhalt übereinstimmen.
- message origin authentication – verifiziert, dass der Sender einer Nachricht und die Absenderangabe übereinstimmen.
- general identity authentication – verifiziert, dass ein Subjekt Besitzer einer vorgegebenen Identität ist.

Im Folgenden soll in dieser Arbeit beim Verwenden des Begriffs Authentifizierung generell der dritte Authentifizierungsanlass verstanden werden.

In einem System, bei dem der Zugriff beschränkt ist, muss ein Subjekt zunächst die eigene Authentizität (Echtheit und Glaubwürdigkeit) nachweisen, bevor über einen Zugriff auf eine Ressource entschieden werden kann. Dies erfolgt durch Maßnahmen der Identifizierung und der Authentifizierung (vgl. [24] S. 5). In einem ersten Schritt wird also das Subjekt identifiziert, was bedeutet, dass eine vorgegebene Identität (z.B. ein Nutzername) dem System präsentiert wird. In einem zweiten Schritt erfolgt nun die Verifikation bzw. Authentifizierung der Identität (vgl. [25] S. 9). Auch in der ISO/IEC 15408 wird davon ausgegangen, dass eine geläufige Sicherheitsanforderung nicht nur die Identifizierung eines Subjekts darstellt, sondern in einem zweiten Schritt eine Verifikation zu erfolgen hat. Dies

sollte durch private Informationen erfolgen, die lediglich vom Besitzer der Identität präsentiert werden können (vgl. [18] S. 151). Benantar und Opplinger (vgl. [26] S. 10 f. und [27] S. 17 f.) unterscheiden dabei drei Arten von privaten Informationen, die ein Subjekt zur Authentifizierung präsentieren kann:

- Präsentation von etwas, was das Subjekt weiß – diese Art der Authentifizierung nutzt ausgetauschte Geheimnisse zur Verifizierung, z.B. Passwörter, personal identification numbers (PINs), transaction authentication numbers (TANs).
- Präsentation von etwas, was das Subjekt besitzt – diese Art der Authentifizierung nutzt sog. Token, in denen Berechtigungsmerkmale (credentials) gespeichert sind, z.B. Schlüssel, Schlüsselkarten, Smart Cards.
- Präsentatiuon von etwas, was das Subjekt ist – diese Art der Authentifizierung verifiziert ein Subjekt anhand unterscheidbarer biometrischer Merkmale, z.B. Fingerabdruck, Spracherkennung oder Unterschrift.

Autorisierung

Um einen Zugriff auf eine geschützte Ressource zu beschränken, müssen unterschiedliche Zugriffsrechte für Subjekte und Subjektgruppen festgelegt werden. Sogenannte Zugriffsrichtlinien (Security Policies) bestimmen, welcher Nutzer innerhalb eines Systems auf welche Ressourcen zugreifen darf und in welcher Art und Weise er dies tun kann. Man unterscheidet bei Security Policies allgemein zwischen zwei Arten (vgl. [26] S. 23 ff.):

- Discretionary Policy – Bei dieser besitzerzentrierten Zugriffsrichtlinie hat der Besitzer einer Ressource die volle Kontrolle darüber, wer außer ihm in welcher Weise auf die Ressource zugreifen kann.
- Mandatory Policy – Bei dieser Zugriffsrichtlinie wird das Besitz-Prinzip nicht verwendet. Zugriffsinformationen werden von einer zentralen Stelle und nicht durch einen Besitzer einer Ressource zugewiesen. Einzelne Systemteilnehmer haben keine Kontrolle über die Verteilung von Zugriffsrechten.

Beide unterschiedlichen Arten der Zugriffsrichtlinien können durch verschiedene Zugriffsmechanismen, z.B. durch eine Zugriffskontrollliste (Access Control List - eine Liste die zugriffsberechtigte Subjekte aufführt), realisiert werden.

Bei der Autorisierung wird, entsprechend der zugewiesenen Zugriffsrechte der Security Policies, dem Subjekt ein Zugriff auf die geschützten Ressourcen gewährt. Besitzt also ein Subjekt eine Zugriffsberechtigung, so ist dieses Subjekt für den Zugriff auf die Ressource autorisiert (vgl. [24] S. 2). Bevor eine Autorisierung eines Systemteilnehmers erfolgen kann, muss jedoch zunächst eine erfolgreiche Identifizierung und Authentifizierung eines Subjektes, wie im vorherigen Abschnitt beschrieben, erfolgt sein. Da die Autorisierung im Speziellen nicht Gegenstand dieser wissenschaftlichen Arbeit ist, soll nicht vertiefend auf diesen Teilbereich der Zugangskontrolle eingegangen werden.

Nachdem die theoretischen Grundlagen in diesem Kapitel beleuchtet wurden, soll im nächsten Kapitel eine Klassifikation von Anonymisierungstechniken stattfinden.

2.3 Kryptografische Primitive

„Cryptography is the science of keeping secrets secret“ ([28] S. 1). Die Sicherheit und somit der Vertraulichkeitsschutz aller modernen Verschlüsselungsmethoden oder auch Verschlüsselungsalgorithmen basieren heutzutage einzig und allein auf Geheimnissen oder Schlüsseln. Kein moderner Algorithmus basiert auf der Sicherheit und Geheimhaltung des Verfahrens an sich. Dieses Designprinzip moderner kryptografischer Verfahren geht auf Arbeiten von Auguste Kerckhoffs aus dem Jahr 1883 zurück und ist daher als Kerckhoffs-Prinzip bekannt. Heute werden kryptografische Algorithmen veröffentlicht, um Sicherheitsanalysen durch eine breite Öffentlichkeit zu ermöglicht.

Man unterscheidet bei den schlüsselbasierten kryptografischen Algorithmen zwei grundlegende Verfahren: symmetrische Verfahren und asymmetrische Verfahren (vgl. [9] S. 3). Auf beide Prinzipien soll im Folgenden näher eingegangen werden.

Symmetrische Verfahren

Bei den symmetrischen Verfahren sind die Schlüssel, die für die Verschlüsselung und für die Entschlüsselung einer Nachricht eingesetzt werden, entweder gleich oder sehr einfach voneinander zu errechnen. Bei den meisten heute verwendeten Verfahren sind beide Schlüssel gleich. Die Sicherheit des Verfahrens wird, wie oben bereits beschrieben, durch geheimgehaltene Schlüssel erzeugt, während die Verschlüsselungs- und Entschlüsselungsalgorithmen öffentlich bekannt sind. Die besagten Schlüssel müssen, bevor eine vertrauliche Kommunikation stattfinden kann, zunächst zwischen den Kommunikationspartnern ausgetauscht werden. Da ein Angreifer, der in den Besitz des Schlüssels gelangt, alle Nachrichten, ebenso wie die Kommunikationspartner, entschlüsseln kann und somit Kenntnis vom Inhalt der Nachricht erlangen kann, muss ein Austausch von Schlüsseln vertraulich erfolgen (vgl. [9] S. 4). Dies geschieht oft über die im nächsten Abschnitt beschriebenen asymmetrischen Verfahren. Ein Vorteil der symmetrischen Verfahren besteht darin, dass im Gegensatz zu den asymmetrischen Verfahren sehr schnelle, hardwarenahe Implementierungen für fast alle Algorithmen realisiert werden können. Bei den symmetrischen Verschlüsselungsverfahren kann zwischen Block- und Stromchiffrierverfahren unterschieden werden. Beim Blockchiffrierverfahren wird die Nachricht zunächst in gleichgroße Blöcke aufgeteilt und anschließend blockweise verschlüsselt. Beim zweiten Verfahren werden die Nachrichten Bit für Bit verschlüsselt, wodurch die Nachrichten als Ganzes verschlüsselt werden können (vgl. [15] S. 16 ff.).

Asymmetrische Verfahren

Bei asymmetrischen Verfahren (oft auch public-key Verfahren genannt) sind die Schlüssel für die Verschlüsslung und die Entschlüsselung von Nachrichten nicht gleich. Jeder Kommunikationspartner besitzt ein Schlüsselpaar: einen privaten Schlüssel zum Entschlüsseln, der geheimgehalten wird, und einen öffentlichen Schlüssel zum Verschlüsseln, der allen Kommunikationspartnern bekannt ist. Während bei einem symmetrischen Verfahren ein gemeinsamer Schlüssel zwischen den Kommunikationspartner verwendet wird, der zunächst vertraulich ausgetauscht werden muss, ist dies bei asymmetrischen Verfahren nicht der Fall. Um eine vertrauliche Kommunikation zu ermöglichen, veröffentlicht der

Empfänger einer Nachricht den eigenen öffentlichen Schlüssel. Ein Kommunikationspartner verschlüsselt mit diesem Schlüssel die vertrauliche Nachricht. Ein Entschlüsseln der Nachricht ist nun nur mit dem zugehörigen (geheimgehaltenen) privaten Schlüssel des Nachrichtenempfängers möglich (vgl. [9] S. 4 f.).

Die Sicherheit des Verfahrens liegt insbesondere darin begründet, dass aus einem öffentlichen Schlüssel nicht oder nicht in angemessener Zeit der zugehörige private Schlüssel errechnet werden kann. Vice versa ist dies meist sehr einfach möglich. Außerdem ist es praktisch nicht möglich, aus einer abgefangenen chiffrierten Nachricht zusammen mit dem zugehörigen öffentlichen Schlüssel die resultierende Originalnachricht zu errechnen. Die Familie der mathematischen Funktionen, die diese Eigenschaften bereitstellen, werden Einwegfunktionen genannt. Zusätzlich zu der gewünschten Einweg-Eigenschaft soll es möglich sein, über eine sogenannte „Hintertür“ sehr effizient die Originalnachricht zu ermitteln, falls eine geheimgehaltene Information (in diesem Fall der private Schlüssel) bekannt ist. Mathematische Funktionen, die diese Eigenschaft zusätzlich besitzen, werden Trapdoor-Einwegfunktionen genannt und liegen den asymmetrischen Verfahren zugrunde. Asymmetrische Verfahren werden auf Grund ihrer meist sehr rechenaufwendigen Operationen lediglich zum Schlüsselaustausch für symmetrische Verfahren angewandt. Die eigentliche Verschlüsslung der Nachrichten erfolgt dann über effizientere symmetrische Verfahren (vgl. [28] S. 23 ff.).

Eine weitere Anwendung von asymmetrischen Verschlüsselungsverfahren ist das Erzeugen von digitalen Signaturen. Um eine digitale Signatur einer Nachricht zu erzeugen, wendet ein Kommunikationspartner den eigenen privaten Schlüssel auf die Originalnachricht an. Der entstehende Schlüsseltext stellt die digitale Signatur dar, die beweist, dass die Nachricht von einer bestimmten Person stammt. Eine Überprüfung der Signatur kann durch Entschlüsseln mit dem passenden öffentlichen Schlüssel durch eine beliebige Person erfolgen. Da eine korrekte digitale Signatur nur mit dem geheimgehaltenen privaten Schlüssel erzeugt werden kann, ist diese ein Beweis dafür, dass auch nur der Halter des passenden privaten Schlüssel die Signatur erzeugt haben kann und die Nachricht von diesem Halter stammt. In der Praxis werden digitale Signaturen meist nicht auf Originaldokumente angewendet, da die zugrunde liegenden Algorithmen meist sehr aufwändig sind. Vielmehr werde Signaturen oft in Verbindung mit sicheren Hashfunktionen (vgl. nächster Abschnitt) eingesetzt, wobei lediglich der Hashwert des Dokuments signiert wird.

Hashfunktionen

Neben symmetrischen und asymmetrischen Verschlüsselungsverfahren spielen insbesondere sichere Hashfunktionen eine wichtige Rolle im Kontext der Kryptografie. Eine Hashfunktion besitzt einen Input beliebiger Länge und erzeugt daraus einen sog. Hashwert, der eine feste Länge n hat.

$$h : \{0,1\}^* \longrightarrow \{0,1\}^n, m \longmapsto h(m) \qquad \text{aus: [28] S. 39} \tag{4}$$

Eine Funktion $h(x)$, die als Hashfunktion Verwendung findet, wird oft auch als Einwegfunktion bezeichnet, da es nicht möglich ist, aus dem Hashwert $h(m)$ Aussagen über Länge oder Inhalt der Originalnachricht m zu treffen bzw. diese zu rekonstruieren. Weitere Anforderungen an eine sichere Hashfunktion sind, dass es weder möglich sein soll, bei einem gegebenen Hashwert einen Eingabewert zu konstruieren, der den gleichen Hashwert be-

sitzt, noch die Möglichkeit bestehen soll, zwei Originalnachrichten zu konstruieren, die einen gleichen Hashwert besitzen.

Da die Länge der Eingabe für eine Hashfunktion prinzipiell nicht begrenzt ist und die Ausgabe auf eine fixe Größe n limitiert ist, entfallen auf einen Hashwert beliebig viele Eingabewerte. Eine Hashfunktion ist also in keiner Weise kollisionsfrei oder injektiv (d.h. es gibt mehrere Eingabewerte x, x', für die gilt: $h(x) = h(x')$). Trotz dieser vorhandenen Kollisionen werden Hashfunktionen eingesetzt, um einen sog. digitalen Fingerprint einer Nachricht zu erzeugen. Damit kann beispielsweise überprüft werden, ob an einer Nachricht oder einer Datei Modifikationen vorgenommen wurden. Diese können dadurch erkannt werden, dass der Fingerprint der modifizierten Nachricht nicht mehr mit Fingerprint des Originals übereinstimmt. Außerdem werden Hashfunktionen im Zusammenhang mit digitalen Signaturen benutzt, um Nachrichten, die signiert werden sollen, zu verkürzen und somit eine kürzere und effizientere Signatur zu erzeugen (vgl. [28] S. 39 und [29] S. 21).

3 Klassifikation von Anonymisierungstechniken

In diesem Kapitel soll auf wesentliche Anonymisierungstechniken eingegangen werden, mit denen für Empfänger und Sender unterschiedliche Anonymitätsklassen erreicht werden können. An Anonymisierungstechniken ist zwischen Broadcasting, Mixnetzwerken, überlagerndem Senden (DC-Netze) und einer Weiterleitung von Nachrichten über Proxies zu unterscheiden, wobei Anonymisierungstechniken von konkreten Anonymisierungssystemen abgegrenzt werden sollen. Ziel dieser Betrachtung ist daher nicht, unterschiedliche Systeme zu vergleichen, sondern vielmehr eine Klassifizierung der zugrunde liegenden Prinzipien aufzustellen und eine Einordnung der erreichbaren Anonymität (Sender-, Empfänger oder Verbindungsanonymität) vorzunehmen.

3.1 Proxies

Ein Proxy ist ein „Mittelsmann“, der Nachrichten zwischen Kommunikationspartnern weiterleitet. Neben der einfachen Weiterleitung der Nachricht an den Empfänger ersetzt der Proxy die Absenderkennung durch ein entsprechendes Pseudonym. Durch die Verknüpfung von Pseudonym und Kennung des Absenders beim Proxy ist es für einen Nachrichtenempfänger auch möglich, über den Proxy auf eine Nachricht zu antworten. Proxies bieten, selbst wenn mehrere in Reihe verwendet werden, so wie dies bei einem probabilistischen Routingkonzept von Crowds geschieht (vgl. [30]), lediglich Schutz gegen einen lokalen Angreifer. Ein globaler Angreifer, der die Möglichkeit hat, sowohl Ein- aus auch Ausgänge eines Proxies zu beobachten, kann sehr einfach eine Verbindung zwischen ein- und ausgehenden Nachrichten herstellen. Durch ein solches Überbrücken des Proxies ist die Anonymität komplett aufgehoben.

3.2 Broadcasting und implizite Adressierung

Das Broadcasting, als einfacher Mechanismus zum Schutz der Empfängeranonymität, geht auf die Arbeiten von Farber, Larson und Karger (vgl. [31] und [32]) zurück. Senderanonymität kann durch dieses System nicht erreicht werden. Beim Broadcasting erhalten alle Nutzer eines Systems alle Nachrichten der anderen Teilnehmer, und die Empfänger können lokal entscheiden, ob die Nachrichten für sie bestimmt sind. Ein Angreifer kann somit nicht unterscheiden, ob eine Nachricht für den entsprechenden Nutzer bestimmt ist oder dieser die Nachricht lediglich verwirft. Bei vertraulichen Daten können Nachrichteninhalte durch entsprechende kryptografische Sicherungsverfahren geschützt werden.

Damit ein Empfänger erkennen kann, ob eine Nachricht für ihn bestimmt ist, muss diese Nachricht adressiert sein. Dies sollte implizit erfolgen, um die Anonymität des Empfängers zu wahren. Implizite Adressen sind Merkmale oder Bitketten, an denen lediglich der intendierte Empfänger einer Nachricht erkennt, dass diese für ihn bestimmt ist. Erfolgt eine implizite Adressierung offen, d.h. für Dritte erkennbar, ist u.U. ein Verknüpfen der Nachrichten möglich, wenn die Adressierung nicht bei jeder Nachricht gewechselt wird. Somit wird keine (unverknüpfbare) Anonymität erreicht. Bei verdeckter impliziter Adressierung können Adressen von Nachrichten von Dritten nicht auf Gleichheit getestet werden, d.h. ein Verknüpfen von Nachrichten ist nicht möglich. Realisiert werden kann dies durch kryp-

tografische Operationen, bei denen nur der intendierte Empfänger die Nachricht korrekt entschlüsseln kann (vgl. [4] S. 93 f. und [33] S. 628 f.).

3.3 Überlagerndes Senden - DC-Netze

Während beim Broadcast die Anonymität des Empfängers in einer Kommunikationsbeziehung geschützt ist, indem eine Nachricht an alle Gruppenmitglieder gesendet wird, besteht beim DC-Netz ein Schutz der Anonymität des Senders. Weder ein Mitglied innerhalb der Gruppe noch ein Dritter können bei einem Nachrichtenaustausch bestimmen, wer Initiator der Kommunikation war. Die Technik, die Anonymität durch ein überlagertes Senden erreicht, geht auf David Chaum (vgl. [34]) zurück. Ein Beispiel, das das Funktionsprinzip sehr anschaulich beschreibt, ist dem Originaltext von Chaum zu entnehmen.

In einem Kommunikationsnetz funktioniert das DC-Prinzip wie folgt: In einem getakteten, vollständig verbundenen Netzwerk haben alle Kommunikationspartner zunächst paarweise für jede Kommunikationsrunde Schlüssel ausgetauscht. Dabei müssen die Schlüssel die gleiche Länge wie die Nachrichten besitzen. In jeder Runde (Takt) kann ein Teilnehmer eine Nachricht senden, alle anderen Teilnehmer senden Leernachrichten (Folge von Null-Bits). Bevor die Nachrichten (Null-Bits und echte Botschaften) gesendet werden können, werden diese mit den ausgetauschten Schlüsseln XOR verknüpft. Für einen potentiellen Angreifer, der nicht die geheimen ausgetauschten Schlüssel kennt, sehen alle Nachrichten, sowohl Leerbotschaften als auch echte Botschaften wie Zufallszahlen aus. Er kann nicht zwischen den beiden Nachrichtentypen unterscheiden. Durch eine anschließende Überlagerung der gesendeten Nachrichten zu einer Summe heben sich alle paarweisen Schlüssel auf, da jeder Schlüssel genau zweimal in der globalen Summe verwendet wurde. Wenn lediglich eine echte Botschaft und sonst Leerbotschaften gesendet wurden, tritt somit die echte Botschaft zum Vorschein. Abbildung 4 verdeutlicht das Prinzip anhand eines Beispiels mit drei beteiligten Kommunikationspartnern.

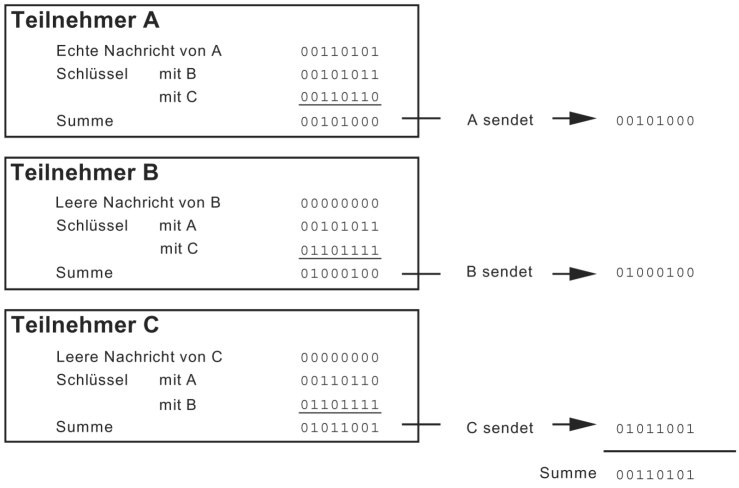

Abbildung 4: Überlagerndes Senden im DC-Netz (aus: [4] S. 98)

Durch das überlagernde Senden aller Teilnehmer innerhalb einer Gruppe wird eine perfekte Anonymität des Senders erreicht. Es ist nicht möglich zu unterscheiden, welcher Teilnehmer eine Leerbotschaft und welcher eine echte Botschaft gesendet hat. Unabhängig

davon kann durch das in Kapitel 3.2 angesprochene Broadcasting Empfängeranonymität erreicht werden. (vgl. [4] S. 97 f.).

Verschiedene Einschränkungen und Nachteile sind bei der Verwendung von DC-Netzen erkennbar, die auch in unterschiedlichen Angriffen auf das System münden. Zunächst ist zu erwähnen, dass eine Synchronisierung der Takte erfolgen muss und in einem Takt lediglich ein Teilnehmer senden kann. Dies macht die Verwendung von Mehrfachzugriffsverfahren notwendig. Außerdem ist das System anfällig für bösartige Teilnehmer, die durch das Senden von beliebigen Bitsequenzen und manipulierten Nachrichten eine Kommunikation vollständig unterbinden und somit sehr einfach einen Denial-of-Service-Angriff (Angriff auf die Verfügbarkeit des Systems) auf das System ausüben können. Letztendlich unmöglich scheint eine skalierbare Implementierung des Systems. Bei einer (n-zu-n Topologie müssen pro Takt $\mathcal{O}(n^2)$ Schlüssel und Nachrichten zwischen den Teilnehmern ausgetauscht werden. Eine Reduzierung der Nachrichtenanzahl ist theoretisch möglich, wenn statt einer n-zu-n Topologie eine Ringtopologie eingesetzt würde. Dies hat jedoch den entscheidenden Nachteil im Bezug auf die bereitgestellte Anonymität: Während bei einer n-zu-n Topologie $n-1$ Teilnehmer kooperieren müssen, um einen Sender zu enttarnen, ist dies bei einer Ringtopologie bereits durch zwei zusammenarbeitende Teilnehmer möglich (vgl. [34] S. 70).

3.4 Mix-Netze

In diesem Unterkapitel soll die von David Chaum bereits 1981 (vgl. [35]) vorgeschlagene Anonymisierungstechnik der Mix-Netze vorgestellt werden. Auf Grund der Bedeutung, die das Mixprinzip in den heute vorgefundenen Anonymisierungssystemen einnimmt, soll diese Technik detaillierter beleuchtet werden als die beiden vorangegangenen Prinzipien der Anonymisierung. Da die Anonymisierungstechnik im Besonderen auf der asymmetrischen Verschlüsselung beruht (vgl. [35] S. 84), soll zunächst eine Einführung in die kryptografische Primitive geben werden, bevor im Anschluss daran auf die allgemeine Funktionsweise und verschiedene Topologien von Mixnetzwerken eingegangen wird. In einem weiteren Schritt werden erreichbare Schutzziele sowie insbesondere auch das Angreifermodell, das Mixnetzen zugrunde liegt, betrachtet. In einem letzten Abschnitt wird auf das Onion- sowie das Garlic-Routing, zwei Abwandlungen des Mixprinzips, eingegangen, da die in Kapitel 4 vorgestellten Anonymisierungssysteme Tor und I2P dieses Prinzip einsetzen, um Anonymität zu erlangen.

3.4.1 Funktionsweise eines Mixnetzwerkes

Ziel eines Mixnetzes ist es, den Nachrichtenaustausch (Kommunikationsbeziehung) zwischen Sender und Empfänger einer Nachricht durch das Senden von Nachrichten über spezielle Knoten innerhalb des Netzwerkes, den sog. Mixen, zu verbergen. Ein einzelner Mix hat dabei lediglich die Aufgabe, die eingehenden Nachrichten modifiziert weiterzuleiten und somit eine Verkettung zwischen eingehender und ausgehender Nachricht zu unterbinden. Um dies zu leisten, muss ein Mix, der nach dem Originalprinzip von Chaum arbeitet, folgende Operationen vornehmen (vgl. [36] S. 94 f.):

- Sammeln und Speichern von eingehenden Nachrichten, bis ausreichend viele unterschiedliche Nachrichten von unterschiedlichen Sendern vorhanden sind.
- Umkodieren der Nachrichten. Dies geschieht durch Anwenden von asymmetrischen Verschlüsselungsverfahren. Die Umkodierung ist notwendig, um ein Verknüpfen von eingehenden und ausgehenden Nachrichten zu verhindern.
- Verändern der Reihenfolge der Nachrichten und Ausgabe in einem Schub.
- Prüfen, ob eine eingehende Nachricht bereits gemixt wurde. Hierdurch werden Angriffe durch Wiederholungen verhindert, da eine gleiche eingehende Nachricht wieder zu einer gleichen Ausgabe führen würde und eine Verkettung möglich wäre.
- Einfügen von sog. Padding, um eine gleiche Nachrichtenlänge zu erhalten. Dadurch wird trotz Umkodierung eine Verkettung auf Grund der spezifischen Nachrichtenlängen verhindert.

Die Arbeitsweise eine Mix-Netzwerkes ist in der Abbildung 5 veranschaulicht:

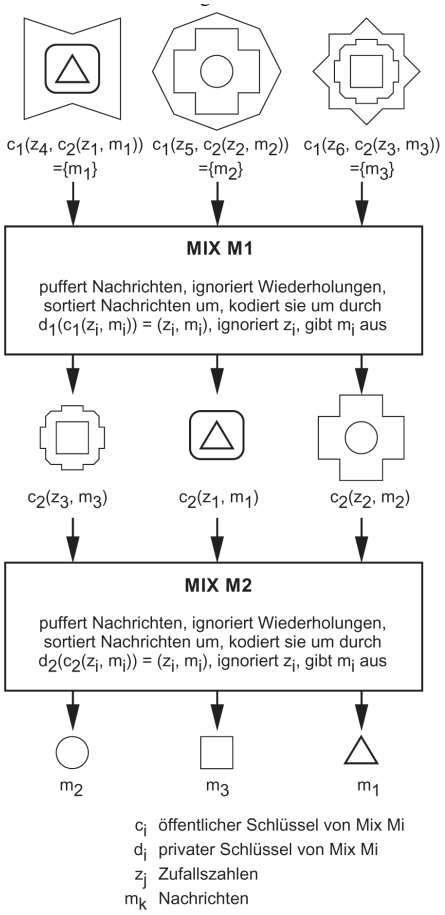

Abbildung 5: Funktionsweise von Mixknoten (aus: [4] S. 95)

Das Umkodieren der eingehenden Nachrichten wird durch asymmetrische Kryptosysteme erreicht (durch ein Entschlüsseln der eingehenden Nachrichten mit Hilfe des privaten Schlüssels des Mixes) und stellt die Hauptaufgabe eines Mixes dar. Zwar ist es prinzipiell ausreichend, dass eine Nachricht lediglich über *einen* vertrauenswürdigen Mixknoten geleitet wird, um Unbeobachtbarkeit zu gewährleisten, jedoch sollte eine Nachricht immer

über mehrere Mixe geleitet werden, da sonst ein einzelner Mix sowohl Empfänger als auch ursprünglichen Sender der Nachricht kennt. Falls dieser Mix nun kompromittiert ist, kann dieser eine Verkettung der Nachrichten herstellen und die Anonymität aufheben. Grundsätzlich sind beim Einsatz von mehreren Mixen zwei wesentliche Topologien denkbar, wie Mixknoten angeordnet werden können: Mixsequenzen oder Mixkaskaden. Bei Mixsequenzen wählt ein Nutzer des Mixnetzwerkes eine Menge von Mixknoten aus der zur Verfügung stehenden Gesamtheit frei aus und legt zusätzlich die Reihenfolge fest, in der diese Knoten durchlaufen werden sollen. Bei der Verwendung von Mixkaskaden ist die Wahlfreiheit des Nutzers eingeschränkt: Mixkaskaden sind vordefinierte Folgen von Mixinstanzen - der Nutzer kann hierbei keine einzelnen Mixe mehr auswählen, sondern benutzt feste Routen. Eine hybride Lösung kombiniert die Eigenschaften beider Verfahren, indem ein Nutzer eine Sequenz aus Kaskaden zur Anonymisierung auswählen kann (vgl. [3] S. 57 ff.).

Voraussetzung für den Einsatz von mehreren Mixinstanzen ist, dass die Originalnachricht des Absenders entsprechend aufbereitet wird. Dazu wird diese mit den öffentlichen Schlüsseln c_i der durchlaufenen Mixe M_i rekursiv verschlüsselt:

$$m_i = c_i(z_i, A_{Mi+1}, m_{i+1}) \qquad \text{mit } i = n...1 \qquad \text{aus: [4] S. 96} \tag{5}$$

wobei m_i die Nachricht darstellt, die der i-te Mix erhält und m_{n+1} die Nachricht, die der Empfänger erhält. A_{Mi} bezeichnet die Adresse des nächsten Mixes bzw. bei $i = n + 1$ die Adresse des Empfängers. z_i sind Zufallszahlen, die einen Angreifer davon abhalten sollen, eine ausgehende Nachricht einfach erneut mit dem öffentlichen Schlüssel des Mixes zu chiffrieren und somit eine Zuordnung zwischen eingehender und ausgehender Nachricht herstellen zu können. Sie werden von den Mixen nach dem Entschlüsseln entfernt. Die benötigten öffentlichen Schlüssel werden meist durch eine zentrale Instanz verteilt. Da die einzelnen Schichten für die jeweiligen zwischengeschalteten Mixe verschlüsselt sind, kann jeder Mix lediglich *seine* Schicht entschlüsseln und die im Aussehen veränderte Nachricht an den nächsten Mix weiterleiten. Einem außenstehenden Dritten ist es nicht möglich, den Entschlüsselungsvorgang nachzuvollziehen, da die dafür notwendigen privaten Schlüssel selbstverständlich geheim gehalten werden (vgl. [33] S. 629). Da der letzte Mixknoten, der in einem Adresspaket definiert wurde, die Nachricht unverschlüsselt an den eigentlichen Empfänger A_{Mn+1} weiterleitet, bietet der Einsatz von Verschlüsselungsverfahren hier keinen Schutz der Vertraulichkeit der Nachrichten.

Durch den Einsatz von Mixsystemen ist es ebenfalls möglich, Empfängeranonymität zu gewährleisten. Dies ist beispielsweise notwendig, wenn der Empfänger einer anonymen Nachricht dem Sender auf seine Nachricht antworten möchte. Zu diesem Zweck werden sog. anonyme Rückadressen verwendet. Diese Rückadressen werden, wie leicht ersichtlich ist, ähnlich dem obigen Schema rekursiv zusammengesetzt.:

$$r_i = c_i(k_i, A_{Mi+1}, r_{i+1}) \qquad \text{mit } i = n...1 \qquad \text{aus: [4] S. 96} \tag{6}$$

Wenn ein Empfänger nun eine Nachricht anonym erhalten möchte, veröffentlicht er eine anonyme Rückadresse, also ein mehrfach chiffriertes Adresspaket (r_1). Der Sender kann dieses Adresspaket zusammen mit der eigentlichen Nachricht an den ersten klarschriftlich angegebenen Mixknoten (A_{M1}) senden. Die einzelnen Mixinstanzen entschlüsseln mit den privaten Schlüsseln nacheinander die Schichten des Pakets und leiten die verbleibenden Schichten zusammen mit der mit dem mitgelieferten symmetrischen Schlüssel k_i chiffrierten Nachricht weiter an die nächste Adresse A_{Mi+1}. Zum Schluss erhält der Empfänger A_{Mn+1} die mehrfach chiffrierte Nachricht zusammen mit dem sog. Adresskennzeichen

(r_{n+1}), an dem er erkennt, welche Schlüssel er zum Entpacken der Nachricht verwenden muss. Schritt für Schritt kann er mit den symmetrischen Schlüsseln k_i die chiffrierte Nachricht entpacken. Zu beachten ist bei der Verwendung von anonymen Rückadressen jedoch, dass jede anonyme Rückadresse lediglich einmal verwendet werden darf, um eine Unverkettbarkeit von Nachrichten und somit die Anonymität des Empfängers zu gewährleisten (vgl. [4] S. 96 f. und [3] S. 61-66). Dies macht eine naive Verwendung von anonymen Rückadressen wenig praktikabel.

3.4.2 Erreichbare Schutzziele

Nachdem detailliert auf die Arbeits- und Wirkungsweise von Mixnetzwerken eingegangen wurde, soll im weiteren Verlauf spezifiziert werden, welche Schutzziele mit Mixnetzwerken erreicht werden können. Dazu ist es notwendig, zunächst aufzuzeigen, welches Angreifermodell den Betrachtungen zugrunde gelegt wird. Eine Untersuchung von möglichen Angriffen auf Mixnetzwerke soll nicht Gegenstand dieser Arbeit sein. Eine detaillierte Analyse ist in den Ausführungen von Berthold, Pfitzmann und Standtke zu finden (vgl. [20]). Zunächst ist bei einem Angreifermodell die maximale Stärke eines potentiellen Angreifers zu spezifizieren. Berthold et al. (vgl. [20] S. 30) gehen von einem globalen Angreifer aus. Nach Demuth ([3] S. 33) werden diesem globalen Angreifer die folgenden Möglichkeiten zugeschrieben: „Ein globaler Angreifer:

- ist in der Lage, auf dem Übermittlungswege auf beliebig viele Stellen bzw. beliebig viele Übermittlungsknoten zuzugreifen,
- kennt jeden Teilnehmer der Kommunikationsbeziehungen, ohne explizit einer Beziehung zwei Teilnehmer zuordnen zu können und
- sieht jede Aktivität aller Teilnehmer.“

Serjantov (vgl. [21] S. 32) unterscheidet beim Angreifer weiterhin danach, ob er Netzwerkverkehr modifizieren (aktiver Angreifer) oder lediglich Verkehr beobachten kann (passiver Angreifer). Außerdem ist zu unterscheiden, ob in einem Netzwerk kompromittierte Mixe vorhanden sind, auf die ein Angreifer zurückgreifen kann. Bei diesen Knoten besitzt der Angreifer die Möglichkeit, eingehende und ausgehende Nachrichten in Verbindung zu setzen. Sowohl Berthold et al. als auch Demuth gehen bei ihren Betrachtungen davon aus, dass alle Mixe bis auf einen auf einem Pfad zwischen Sender und Empfänger kompromittiert sein können. Weiterhin spezifizieren beide Autoren, dass lediglich ein kleiner Anteil der Teilnehmer eines Mix-Systems mit dem Angreifer zusammenarbeitet, da sonst eine Anonymität nicht erreichbar ist (vgl. [3] S. 56 und [20] S. 30).

Nachdem die unterschiedlichen Eigenschaften eines im Folgenden hier geltenden Angreifermodells für Mixnetzwerke beleuchtet wurden, lassen sich an Anlehnung an Demuth ([3] S. 56) die nachstehenden drei Schutzziele für Mixnetzwerke formulieren:

1. *Unbeobachtbarkeit der Kommunikationsbeziehung zwischen Sender und Empfänger*- Grundsätzlich sollte es einem globalen Angreifer unmöglich sein zu erkennen, dass zwei Teilnehmer eines Mixnetzwerkes miteinander in Kommunikationsbeziehung stehen.

2. *Sender- und Empfängeranonymität*- wie bereits im vorherigen Abschnitt erläutert, sollte durch das Mix-Netzwerk die Anonymität des Senders oder des Empfängers (durch Zuhilfenahme von anonymen Rückadressen) gewahrt werden.

3. *Vertraulichkeit der Nachrichten*- der Inhalt der Nachrichten sollte lediglich dem Sender und dem Empfänger zugänglich sein. Allen Dritten (Mixknoten oder Beobachter) sollte ein Zugriff auf den Nachrichteninhalt verwehrt bleiben.

3.4.3 Onion-Routing und Garlic-Routing

In diesem letzten Abschnitt des Kapitels wird auf Erweiterungen des Mixkonzeptes eingegangen, da diese die Grundlagen für die Anonymisierungssysteme bilden, die im nächsten Kapitel näher beleuchtet werden. Konkret soll hier zunächst das Onion-Routing näher betrachtet werden, das auf Arbeiten von Goldschlag, Reed und Syverson aus dem Jahr 1999 zurückgeht (vgl. [37]). Das Garlic-Routing, welches zuerst von Dingledine in 2000 vorgeschlagen wurde (vgl. [38]), stellt eine weitere Modifikation des Onion-Routings dar. Beide Systeme beruhen auf dem Grundprinzip eines Mixnetzwerkes nach Chaum, wobei sie im Gegensatz zu Chaum-Mixen, die für *high-latency* (nicht verzögerungssensitiven) Verkehr ausgelegt waren, auch für *low-latency* (verzögerungssensitiven) Verkehr wie z.B. Webanwendungen geeignet sind. Da beide Systeme viele gemeinsame Eigenschaften und Arbeitsweisen besitzen, sollen sie im weiteren Verlauf dieses Abschnittes zunächst zusammen betrachtet werden, ehe zum Ende des Abschnittes, nachdem die grundsätzlichen Arbeitsweise von beiden Verfahren beschrieben wurde, eine Abgrenzung vorgenommen werden soll. In diesem Zusammenhang sollen im Folgenden aus Gründen der besseren Lesbarkeit zunächst unter dem Begriff Onion-Routing beide echtzeitfähigen Mechanismen (Onion- und Garlic-Routing) verstanden werden.

Durch den gravierenden Unterschied im Anwendungsszenario zwischen Mixen, die auf der Grundidee von Chaum beruhen (keine Echtzeitfähigkeit gefordert) und echtzeitfähigen Routingverfahren, sind vielfältige Modifikationen des von Chaum vorgeschlagenen Grundkonzeptes notwendig, die teilweise auch auf Kosten der erreichbaren Anonymität gehen. Syverson et al. (vgl. [39] S. 97 f.) stellen zunächst als schwerwiegendste Modifikation heraus, dass ein Onion-Router kein Sammeln und Speichern von eingehenden Nachrichten vornimmt, so wie dies als Aufgaben eines Mixes nach Chaum in Kapitel 3.4.1 aufgeführt wurde. Ein Sammeln und Speichern ist auf Grund der geforderten Echtzeitfähigkeit nicht möglich und schwächt somit den Schutz, der durch Onion-Routing erreicht werden kann. Eine großes Verkehrsaufkommen in einem Onion-Routing-System kann diesen verringerten Schutz, der ein Verknüpfen von eingehenden und ausgehenden Nachrichten an einem Router ermöglichen kann, teilweise aufwiegen. Ein weiterer Unterschied zum ursprünglichen Modell besteht darin, dass beim Onion-Routing ein verbindungsorientiertes Routing stattfindet. Das bedeutet, dass Verbindungen zunächst, bevor Nachrichten transportiert werden können, aufgebaut und nach einem Nachrichtenaustausch diese Verbindungen wieder abgebaut werden müssen. Beim Verbindungsaufbau werden sog. Onions vom Initiator versandt, die eine Strecke durch das Netzwerk festlegen. Im Aufbau entsprechen diese Pa-

kete weitestgehend den rekursiven Nachrichten von Chaum-Mixen, die im Abschnitt 3.4.1 beschrieben wurden. In der Onion sind neben Routinginformationen auch symmetrische Schlüssel für den weiteren Datenaustausch bereitgestellt. Jeder Router in der Verbindung merkt sich bei Empfang einer Onion, woher er die Onion bekommen hat und wohin er die entpackte Nachricht anhand der Routinginformationen gesendet hat. Dieser Verknüpfung weist er eine Pfad-ID zu. Ist ein Pfad initialisiert, können die eigentlichen Datenpakete versandt werden. Dazu werden diese mit den symmetrischen Schlüsseln der einzelnen Router rekursiv vom Sender verschlüsselt. Es werden keine Routinginformationen mehr beigefügt, sondern es erfolgt ein Weiterleiten der Nachrichten anhand der zugewiesenen Pfad-ID. Sendet der Initiator eine Nachricht, so verschlüsselt er die Datenpakete rekursiv mit den symmetrischen Schlüsseln der einzelnen Router, die bereits bei der Pfadinitialisierung ausgetauscht wurden. Jeder Router entfernt bei Empfang der Nachricht seine Verschlüsselungsschicht von der Nachricht, sodass der Empfänger die Nachricht klarschriftlich erhält. Ein Antworten ist ebenfalls möglich. Dazu sendet der Empfänger der ersten Nachricht klarschriftlich zusammen mit der empfangenen Pfad-ID eine Nachricht an den letzten Router. Die Nachricht wird dann in umgekehrter Richtung von allen Routern im Pfad verschlüsselt. Der Initiator kann durch die symmetrischen Schüssel die Originalnachricht wieder sichtbar machen. So werden asymmetrische Verschlüsselungsverfahren lediglich beim Aufbau der Pfade benötigt, was die Echtzeitfähigkeit des Verfahrens auf Grund des geringeren Rechenaufwands von symmetrischen Verfahren begünstigt. Durch symmetrische bzw. asymmetrische Verschlüsselung und Einsatz von sog. Padding erscheinen alle Nachrichten entlang einer Verbindung verschieden. Einem Angreifer oder selbst einem korruptem Onion-Router ist es nicht möglich, den Weg eines Nachrichtenpaketes zu verfolgen.

Und doch ist ein echtzeitfähiges Anonymisierungsnetzwerk einer Reihe von Angriffen ausgesetzt, die nicht-echtzeitfähige Netzwerke leicht verhindern können und die die Anonymität des Systems stark gefährden können. Rennhard und Plattner (vgl. [40] S. 256 f.) beschreiben recht anschaulich, warum es so schwierig ist, in echtzeitfähigen Anonymisierungsnetzwerken die Anonymität aufrechtzuerhalten. Zwei sehr mächtige Instrumente, die der Angreifer besitzt, um die Anonymität zu untergraben, sind zum einen die Verkehrsanalyse (passiver Angreifer) und zum anderen das Kontrollieren von einer gewissen Anzahl von Knoten (aktiver Angreifer). Bei der Verkehrsanalyse versucht ein Angreifer, eingehende und ausgehende Pakete auf Grund der Nachrichtenlänge oder eines Zeitmusters zu korrelieren. Während durch einheitliche Nachrichtenlängen (durch Hinzufügen von sog. Padding) der erste Schwachpunkt sehr einfach behoben werden kann, gestaltet sich das Unterbinden der zweiten Korrelationsmöglichkeit auf Grund der Zeitmuster sehr schwierig. Eine Korrelation kann dabei an verschiedenen Routern stattfinden oder auch direkt bei den Benutzern des Systems (Ende-zu-Ende Verkehrsanalyse). Eine Möglichkeit ist das Einfügen von Cover-Traffic (künstlicher Verkehr, der durch das System selbst erzeugt wird) und das Sammeln von Nachrichtenpaketen bei den Mixen, was jedoch bewirkt, dass die Latenzzeit des Systems immens steigt und die nutzbare Bandbreite belastet wird. Auch können langfristige Zeitmuster eines Nutzerverhaltens (online/offline) Korrelationsmöglichkeiten eröffnen, die durch Cover-Traffic und Sammeln von Nachrichten nicht zu beheben sind. Durch das Kontrollieren von mehreren Knoten des Netzwerkes ist es einem globalen aktiven Angreifer noch einfacher möglich, zeitliche Muster im Verkehrsaufkommen zu erkennen, selbst wenn Cover-Traffic eingesetzt wird. Rennhard und Plattner kommen letztendlich zu dem Schluss, dass ein *globaler* Angreifer, vor dem ein Mixnetzwerk nach Chaum schützen kann, *jedes* abgewandelte, echtzeitfähige System brechen kann.

So ist auch zu erklären, dass die beim Onion-Routing formulierten Schutzziele diesbezüglich moderater ausfallen als die der nicht-verzögerungssensitiven Chaum-Mixe. Reed, Syverson und Goldschlag (vgl. [41] S. 44) definieren als eines der Ziele des Onion-Routings den Schutz vor Verkehrsanalysen, gehen dabei aber nicht explizit auf die mögliche zeitliche Verkettung von Nachrichten ein. Bei der Beschreibung des Angreifermodells gehen die Autoren von folgender Einschränkung aus:

„We assume roving attackers that can monitor part, but not all, of the network at a time“([41] S. 49).

Dies bedeutet, dass ein globaler Angreifer nicht in den Schutzzielen eingeschlossen wird. Im Rahmen des Tor-Projektes, das auf Onion-Routing basiert, wird in Bezug auf den Angreifer deutlich formuliert, dass globale Angreifer nicht Gegenstand der Betrachtung und der Schutzziele sind. Weiterhin wird konstatiert, dass auf Grund der Unterstützung von interaktiver Kommunikation eine Vielzahl von Attacken auf die Anonymität des Systems möglich sind, die auf originäre Mixnetzwerke so nicht stattfinden können. Es ist also klar ersichtlich, dass theoretisch ein höherer Grad an Anonymität erreichbar ist, jedoch dies letztendlich meist nur auf Kosten der Latenzzeit realisiert werden kann. Somit sind Echtzeitfähigkeit des Systems und bereitgestellte Anonymität gegeneinander abzuwägen (vgl. [42] S. 2 f.).

Nachdem eine Abgrenzung von Garlic-Routing und Onion-Routing zu originären Mixnetzwerken stattgefunden hat, soll nun aufgezeigt werden, welche Unterschiede zwischen Onion-Routing und Garlic-Routing auszumachen sind. Garlic-Routing und Onion-Routing unterscheiden sich lediglich in einem Detail. Bei beiden Konzepten werden die Routing-Informationen in verschlüsselten Schichten verborgen und somit erhält der jeweilige Router durch Entschlüsseln der nächsten Schicht die benötigten Routing-Informationen. Während beim Onion-Routing ein einzelner Routingpfad vorgegeben wird, werden beim Garlic-Routing mehrere mögliche nächste Routen vorgegeben. Eine Schicht enthält somit mehrere Routingpakete, ähnlich wie die Knolle eines Knoblauchs mehrere Zehen pro Schicht enthält. Der Router kann also eine „Knolle“ auswählen, die einen möglichen Pfad zwischen dem aktuellen Router und dem Empfänger der Nachricht darstellt und entsprechend einen der möglichen Pfade bestimmen (vgl. [38] S. 74). Mit diesem Konzept sind sowohl Vor- als auch Nachteile verbunden. Durch die Redundanz der möglichen Pfade wird die Ausfallsicherheit und Robustheit des Systems erhöht, da bei möglichen Ausfällen von Routern andere Pfade gewählt werden können. Nachteilig wirken sich der erhöhte Platz- und Bandbreitenbedarf sowie der zusätzliche Aufwand für das Verschlüsseln der redundanten Pfadinformationen aus.

4 Anonymisierungsnetzwerke zum Anbieten von anonymen Diensten

„Although recent research provides many techniques for anonymous web-browsing, anonymous internet services, i.e. to run a web server or file server without revealing ones identity, have received little or no attention.“ ([14] S. 1). Auch Scarlata et al. teilen diese Meinung. Sie sind der Ansicht, dass ein Fokus der bisherigen Arbeiten im Bereich der Anonymisierungsnetzwerke auf der Bereitstellung von Anonymität für den Initiator einer Kommunikationsverbindung lag (vgl. [5] S. 272). In diesem Abschnitt sollen nun unterschiedliche Anonymisierungsnetzwerke, die das Anbieten von anonymen Diensten unterstützen, näher betrachtet werden. Gegenstand der Betrachtung sind konkret die Anonymisierungsnetzwerke Rewebber (vgl. [6]), Tor (vgl. [11]) sowie die beiden Peer-to-Peer Architekturen I2P (vgl. [8]) und Tarzan vgl. [13]), da es mit diesen Netzwerken auch möglich erscheint, das zu Beginn dieser Arbeit beschriebene Anwendungsszenario umzusetzen. Außerdem sollen die erwähnten Systeme dahingehend untersucht werden, inwieweit Authentifizierungsverfahren bereits umgesetzt wurden bzw. eine Realisierung möglich erscheint. Abschließend soll in einem Fazit ein System für eine Umsetzung im Kontext dieser Arbeit ausgewählt werden. So genannte „content distribution systems“ wie Freenet (vgl. [43]) oder Publius (vgl. [44]), die das anonyme Veröffentlichen von Dokumenten erlauben und diese im Netzwerk speichern, sollen nicht Gegenstand der Betrachtung sein. Das Design dieser Systeme lässt eine interaktive Kommunikation und ein flexibles Anbieten von dynamischen Inhalten, wie dies das eingangs beschriebene Anwendungsszenario eines Blogs erfordert, nicht zu.

Bevor nun auf die oben aufgeführten konkreten Systeme näher eingegangen wird, sollen in einem einleitenden Abschnitt zunächst die generelle Infrastruktur und die Kernkomponenten von Anonymisierungsnetzwerken, die das Anbieten von anonymen Diensten ermöglichen, beleuchtet werden.

4.1 Infrastruktur für das Anbieten von anonymen Diensten

Scarlata et al. (vgl. [5] S. 275) gehen bei ihren Ausführungen für Empfängeranonymität davon aus, dass zunächst jedes Protokoll, das eine Anonymität des Initiators gewährleisten kann, auch für die Etablierung von Empfängeranonymität geeignet ist. Die Autoren sehen zwei grundsätzliche Möglichkeiten, um ein solches originäres Protokoll zu erweitern: Unter Zuhilfenahme von Proxies oder durch Verwendung von Muliticasting. Auch Landsiedel et al. (vgl. [14] S. 2) nutzen den proxybasierten Ansatz, um eine Infrastruktur für anonyme Internetdienste zu beschreiben. Abbildung 6 verdeutlicht die vier wesentlichen Schritte, die notwendig sind, damit ein Dienstanbieter beim proxybasierten Ansatz anonyme Anfragen erhalten kann.

Im ersten Schritt baut der Dienstanbieter zu einem ihm bekannten Proxy unter Verwendung des originären Anonymisierungsprotokolls, welches dem Initiator einer Verbindung Anonymität gewährt, eine Verbindung auf. Der Proxy erhält ein Pseudonym, das dem Dienst zugeordnet werden kann, sowie Pfadinformationen, wie der Service erreichbar ist. In einem zweiten Schritt sendet ein Initiator der Serviceanfrage Datenpakete an den Proxy. Dies kann ebenfalls über eine anonyme Verbindung geschehen, um die eigene Identität

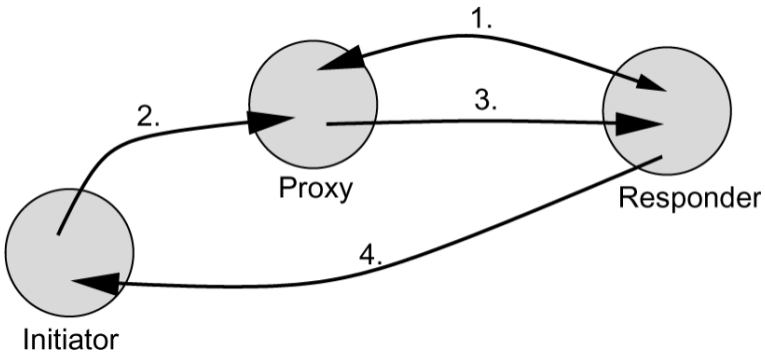

Abbildung 6: Verwendung eines Proxies für die Empfängeranonymität (aus: [5] S. 275)

zu verbergen. Der Header des Datenpakets enthält das Pseudonym, das der Proxy dem Dienst zugeordnet hat. Der Proxy entpackt das Paket und leitet den Datenanteil über den vordefinierten Pfad weiter an den Empfänger der Nachricht. Muss die Identität des Dienstnutzer nicht geschützt werden, sollte im weiteren Verlauf der Datenverkehr nicht mehr über den Proxy geleitet werden, um eine mögliche Verzögerung zu vermeiden. Der Service kann hierzu über das Anonymisierungsprotokoll, das Initiatoranonymität gewährt, eine Verbindung direkt zum Dienstnutzer aufbauen. Soll die Identität des Dienstnutzers hingegen verborgen bleiben, muss der gesamte Datenverkehr über den Proxy transportiert werden. Die Verwendung eines Proxies setzt lediglich voraus, dass diesem insofern vertraut werden kann, dass er keine Pakete verwirft und somit die Unerreichbarkeit des Services erzwingt. Bezüglich sonstiger Sicherheitsaspekte muss einem Proxy kein weiteres Vertrauen geschenkt werden.

Der Multicastansatz benötigt keinen zwischengeschalteten Proxy. Auch hier können alle Protokolle, die einem Initiator einer Verbindung Anonymität gewährleisten, Verwendung finden. Anstatt eine Proxyadresse und ein Pseudonym zu veröffentlichen, gibt der Dienstanbieter eine Multicastadresse und eine Zufallszahl weiter. Die Zufallszahl gewährleistet, dass Dienste Anfragen eindeutig zuordnen können, falls mehrere Dienste Anfragen an eine Multicastadresse abhören. Der Nachteil dieser Variante ist, dass durch das Multicasting in der Regel mehr Verkehr im Netzwerk erzeugt wird, was insbesondere bei verzögerungssensitiven Diensten kritisch sein kann. Der Vorteil dieser Variante liegt darin, dass kein Proxy benötigt wird, dem ein Mindestmaß an Vertrauen entgegen gebracht werden muss (vgl. [5] S. 274 f.).

Eine Implementierung eines Authentifizierungsmechanismus und einer nachfolgenden Autorisierung des Zugriffes sehen die beiden hier beschriebenen allgemeinen Infrastrukturvorschläge von Landsiedel et al. und Scarlata et al. nicht vor. Bei einer proxybasierten Struktur scheint es jedoch sehr einfach möglich, eine Authentifizierung bereits auf Netzwerkebene durchzuführen. Der zwischengeschaltete Proxy kann bei Anfragen bereits gewährleisten, dass lediglich autorisierte Nutzer Zugriff auf die Ressource erhalten. Hierzu ist es jedoch notwendig, dass in einer vorgeschalteten Vereinbarung zwischen dem Proxy und dem Dienstanbieter geklärt wird, wie eine Authentifizierung und ggf. eine anschließende Autorisierung durchzuführen sind. Dazu sind insbesondere u.U. auch Zugangsberechtigungsdaten, z.B. Access Control Lists, auszutauschen. Auf dieser Grundlage kann ein Proxy eine Art Firewall schon auf Netzwerkebene darstellen und ungewollte Zugriffe auf einen Service unterbinden. Auch ist es ihm möglich, durch Setzen von entsprechenden Policies einen Dienst davor zu schützen, dass er durch eine übermäßige Anzahl von Zugriffen außer Gefecht gesetzt wird. Bei einer Infrastruktur, die auf der Verwendung einer

Multicastadresse beruht, ist die Implementierung einer Zugangskontrolle auf Netzwerkebene nicht möglich. Da keine dritte Instanz (beispielsweise ein Proxy) zwischengeschaltet wird, können Zugriffe auf einen Dienst nicht schon im Vorhinein „gefiltert“ werden. Das Abwehren von sog. Denial-of-Service Angriffen, die durch ein erhöhtes Anfragevolumen die Verfügbarkeit eines Dienstes stören wollen, ist mit dieser Infrastruktur nicht möglich. Vielmehr kann durch einen Angriff auf eine Multicastadresse die Verfügbarkeit aller Dienste, die diese Adresse verwenden, gestört werden.

Neben dem vereinfachten Schema nach Scarlata et al. ist es Landsiedel et al. zufolge zusätzlich erforderlich, dass es unmöglich ist, die Identität eines Dienstes „zu übernehmen“, d.h. Anfragen, die an einen bestimmten Dienst gerichtet sind, abzufangen und auf diese entsprechend zu reagieren. Dies kann durch asymmetrische Verschlüsselung verhindert werden. Weiterhin besteht die Erfordernis, dass ein Dienstnutzer die Informationen, wie ein Dienst erreichbar ist, erhält. Dazu ist eine zentrale Instanz, ein sogenannter „Discovery Service“, notwendig. Hier kann ein Dienstnutzer über eine eindeutige Dienstkennung, die außerhalb des Protokolls, z.B. über traditionelle Internetseiten, verbreitet wird, alle notwendigen Verbindungsinformationen (z.B. über welche Proxies der Dienst erreichbar ist, welcher public key zu verwenden ist, etc.) abrufen. Da Landsiedel et al. in ihrem Vorschlag für eine Infrastruktur als Anonymisierungsnetzwerk ein Mixnetzwerk einsetzen, sollte der Discovery Service zusätzlich alle verfügbaren Mixknoten veröffentlichen, damit eine dynamische Auswahl der Mixknoten anhand dieser Publikation erfolgen kann. Landsiedel et al. gehen weiterhin davon aus, dass der Discovery Service eine Art anonyme Rückadresse für jeden Dienst speichert (vgl. [14] S. 2 f.). Dabei sollten jedoch die Einschränkungen im Zusammenhang mit der Verwendung von anonymen Rückadressen, die bereits im Kapitel 2.1 dieser Arbeit beschrieben wurden, beachtet werden.

4.2 Rewebber Netzwerk

Das Rewebber Netzwerk geht auf Arbeiten von Goldberg und Wagner aus dem Jahre 1998 zurück (vgl. [6]). Eine aktuelle Implementierung des Projektes kann unter der aufgeführten Internetadresse[1] nicht mehr ausgemacht werden. Es ist anzunehmen, dass die Arbeit an dem Projekt eingestellt wurde. Trotzdem soll an dieser Stelle kurz auf die zugrunde liegende Infrastruktur eingegangen werden. Beim Design von Rewebber in 1998 wurde sich zunächst sehr stark an einem E-Mail-Szenario orientiert, so wie dies schon bei Chaum 1981 als Anwendungsszenario vorherrschte(vgl. [35]). Der Unterschied zwischen Remailern im Bereich des E-Mail-Verkehrs und einer Architektur, die anonyme Dienste ermöglicht, ist insbesondere der Umstand, dass anonyme Dienste Interaktivität erfordern. Außerdem herrscht beim E-Mail-Verkehr eine Push-Technologie vor (die Daten werden von der Quelle zum Empfänger „geschoben“), während bei anonymen Diensten eine Pull-Architektur zu erkennen ist. Bei Rewebber war das Ziel, ein Pendant zu den schon etablierten Remailern im Emailverkehr auch im HTTP-Verkehr (Hypertext Transfer Protocol) zu konstruieren.

Um dies zu erreichen, sollte ähnlich den Remail-Servern eine Reihe von Knoten in unterschiedlichen Regionen der Welt stationiert werden, die als HTTP-Proxy fungieren. Diese sog. Rewebber bilden ein Rewebber Netzwerk. *„Each node is essentially an HTTP proxy which understands „nested“ URLs [Uniform Resource Locator, der Autor] (i.e. URLs of the form* `http://proxy.com/ http://realsite.com/`*)[...]“*([6] S. 6). Diese Proxies lei-

[1] http://www.isaac.cs.berkeley.edu/taz/ [Stand: 2006.09.27]

ten die Anfrage an die angehängte Adresse weiter. Da diese reine Verschachtelung die Empfängeradresse nicht verbirgt, wird diese mit dem öffentlichen Schlüssel des Rewebber-Knotens chiffriert, sodass lediglich dieser die Adresse lesen kann. Zusätzlich wird das Dokument, welches von einem Dienstnutzer abgerufen wird, nicht im Klartext, sondern verschlüsselt auf dem Server des Dienstanbieters gespeichert, um einen direkten Abruf durch Suchmaschinen etc. zu verhindern. Eine Verschlüsselung erfolgt über effizientere symmetrische Verfahren. Die Dechiffrierung nimmt der Rewebber-Knoten vor. Den symmetrischen Schlüssel erhält er zusammen mit der Anfrage des Clients mit seinem öffentlichen Schlüssel geschützt. Ein weiterer Vorteil des verschlüsselten Speicherns der Dokumente ist die Tatsache, dass auf diese Weise durch Zufügen von sog. Padding einheitliche Dokumentengrößen erzeugt werden können, was eine Verkehrsanalyse erschwert.

Um einem einzelnen Rewebber-Knoten nicht zu ermöglichen, sowohl die Herkunft eines Dokuments als auch den Dienstanbieter zu kennen und somit allein die Anonymität des Dienstes aufdecken zu können, werden mehrere Rewebbber in Reihe geschaltet. Dazu werden Adressen mehrfach verschachtelt, sodass der erste angefragte Rewebber-Knoten die Anfrage nicht direkt an den Dienstanbieter, sondern zunächst an einen weiteren Rewebber-Knoten weiterleitet. Dies kann mehrere Male geschehen, sodass der Knoten, der den Dienstanbieter kennt und der Knoten, der den Inhalt des Dokumentes kennt, nicht zusammenfallen. Abbildung 7 verdeutlicht den Aufbau der verschachtelten Adresse. A, B, C sind die Adressen der jeweiligen Rewebber, K_A, K_B, K_C sind die symmetrischen Schlüssel, mit denen das Dokument verschlüsselt ist und mit denen die einzelnen Rewebber-Knoten das Dokument nacheinander entschlüsseln. Um den Inhalt eines Dokumentes und den zugehörigen Anbieter in Verbindung zu bringen, müssten in diesem Fall A, B und C korrupt sein und zusammenarbeiten (vgl. [6] S. 7 f.).

Abbildung 7: Schematische Darstellung der schichtweisen Verschlüsselung einer Adresse im Rewebber-Netzwerk (aus: [6] S. 7)

Das hier gezeigte Vorgehen der verschachtelten Adressen entspricht sehr stark dem im Kapitel 2.1 angesprochenen Prinzip der anonymen Rückadressen, denn die verschachtelte URL stellt nichts anderes dar als eine anonyme Rückadresse, und die einzelnen Rewebber-Knoten entsprechen in ihrer Funktionsweise sehr stark einem Mixknoten. Hier ergibt sich jedoch folgendes Problem: Durch mehrmaliges Verwenden der gleichen anonymen Rückadresse (verschachtelten URL) kann ein Dienstnutzer den Weg durch das Rewebber-Netzwerk verfolgen, da eindeutig eine Verknüpfung zwischen der eingehenden und der ausgehenden Adresse hergestellt werden kann. Eine Anonymität des Dienstanbieters ist folglich nur dann gewährleistet, wenn jede verschachtelte URL (anonyme Rückadresse) lediglich einmal Verwendung findet. Dies mach eine Anwendung des Systems unpraktikabel. Sowohl I2P als auch Tor verwenden nicht das Prinzip der anonymen Rückadressen, sondern leiten die Nachrichten mit Hilfe von Pfadinformationen weiter bis zum Dienstanbieter.

Als weiteres Problem bezüglich der Praktikabilität stellt sich heraus, dass die mehrfach verschachtelte URL (inklusive der jeweiligen symmetrischen Schlüssel) mehrere hundert Zeichen lang ist. Abhilfe schaffen hier eigene Namensdienste, sog. TAZ-Server (Temporary Autonomous Zone), die eine Adressumsetzung von kurzen Internetadressen mit dem Suffix `.taz` auf die verschachtelten URLs vornehmen. Dabei übernehmen die TAZ-Server keinerlei sicherheitsrelevanten Operationen, sondern stellen nur eine reine Adressumsetzung wie

traditionelle DNS-Server (Domain-Name-System) bereit. Insbesondere ist es ihnen nicht möglich, die verschachtelten URLs weiter zu entschlüsseln (vgl. [6] S. 10). Die TAZ-Server übernehmen eine ähnliche Funktion wie der im Abschnitt 4.1 beschriebene Discovery Service.

Authentifizierungsmechanismen, die Gegenstand dieser wissenschaftlichen Arbeit sind, wurden im bisherigen Design vom Rewebber-Netzwerk nicht berücksichtigt. Auf Grund der dargebotenen Struktur, die einem proxybasierten Ansatz sehr nahe kommt, scheint es jedoch einfach möglich, eine Authentifizierung, wie sie im vorherigen Kapitel allgemein für eine Infrastruktur beschrieben wurde, auch durch die Rewebber-Knoten durchzuführen.

4.3 Tor - The Onion Routing

In diesem Abschnitt soll auf das Tor-Onion-Routing-Projekt (Tor)[2] eingegangen werden, bei dem anonyme Dienste als sog. Location-Hidden-Services oder kurz Hidden Services ermöglicht werden. Sie erlauben das Anbieten von Diensten, ohne die eigene Internet-Protokoll-Adresse (IP-Adresse) zu veröffentlichen.

Tor beruht, wie der Name bereits verdeutlicht, auf dem im Kapitel 3.4.3 beschriebenen Prinzip des Onion-Routings. Um u.a. auch anonyme Dienste ermöglichen zu können, ist das originäre Anonymisierungsprinzip weiterentwickelt worden, sodass sich Tor als eine Art zweite Generation des Onion Routings bezeichnen lässt. In Bezug auf das Ermöglichen von Empfängeranonymität tritt eine Änderung in den Vordergrund, welche die Sicherheit und Praktikabilität des Verfahrens wesentlich erhöht. Obwohl Goldschlag et al. in ihren Ausführungen zum Onion Routing auch die Möglichkeit sehen, über sog. „Reply Onions“, die Ähnlichkeit mit den mehrfach erwähnten anonymen Rückadressen besitzen, Empfängeranonymität zu erzeugen (vgl. [45] S. 145 f.), besitzt dieses Verfahren zwei wesentliche Nachteile. Zum einen ist es nur möglich, die Reply Onions einmalig zu verwenden, um Replay-Attacken zu verhindern, zum anderen ist eine sog. forward-security des Verfahrens nicht gegeben. Dies bedeutet, dass durch einen Schlüsselbruch in der Zukunft durch einen Angreifer ein Pfad durch das Tornetzwerk nachvollzogen werden kann. Da Tor statt einer mehrfach verschlüsselten Datenstruktur (Onion), die einem Client zum Routing durch das Netzwerk übergeben wird, einen inkrementellen Pfadinitialisierungsmechanismus verwendet (sog. telescoping), werden beide genannten Angriffe unterbunden. Dabei wird das Routing von Anfragen nicht durch einzelne Onionstrukturen festgelegt, sondern ein Pfad und die temporären Schlüssel werden direkt durch den Dienstanbieter mit den einzelnen Routern nacheinander initialisiert (vgl. [11] S. 2). Der Client muss lediglich den ersten Router des Pfades, den sog. Introduction Point, kennen. Diese eher dynamische, empfängerzentrierte Pfaderzeugung besitzt ggü. der statischen Variante über die Onion den weiteren Vorteil, dass ein Empfänger auf Kommunikationsfehler im Pfad sehr schnell reagieren und entsprechend neue Pfade auslegen kann.

Im Folgenden sollen nun, basierend auf der Tor Rendezvous Spezifikation (vgl. [46] o.S.) und der Veröffentlichung von Dingledine et al. (vgl. [11] S. 8 f.), die Schritte, die zur Einrichtung eines Hidden Service und anschließenden Anfrage durch einen Client notwendig sind, näher beschrieben werden. Abbildung 8, die angelehnt ist an einen Konferenzbeitrag anlässlich der Konferenz „What The Hack“ in 2005, verdeutlicht die einzelnen Schritte.

[2] http://tor.eff.org [Stand: 2006.09.27]

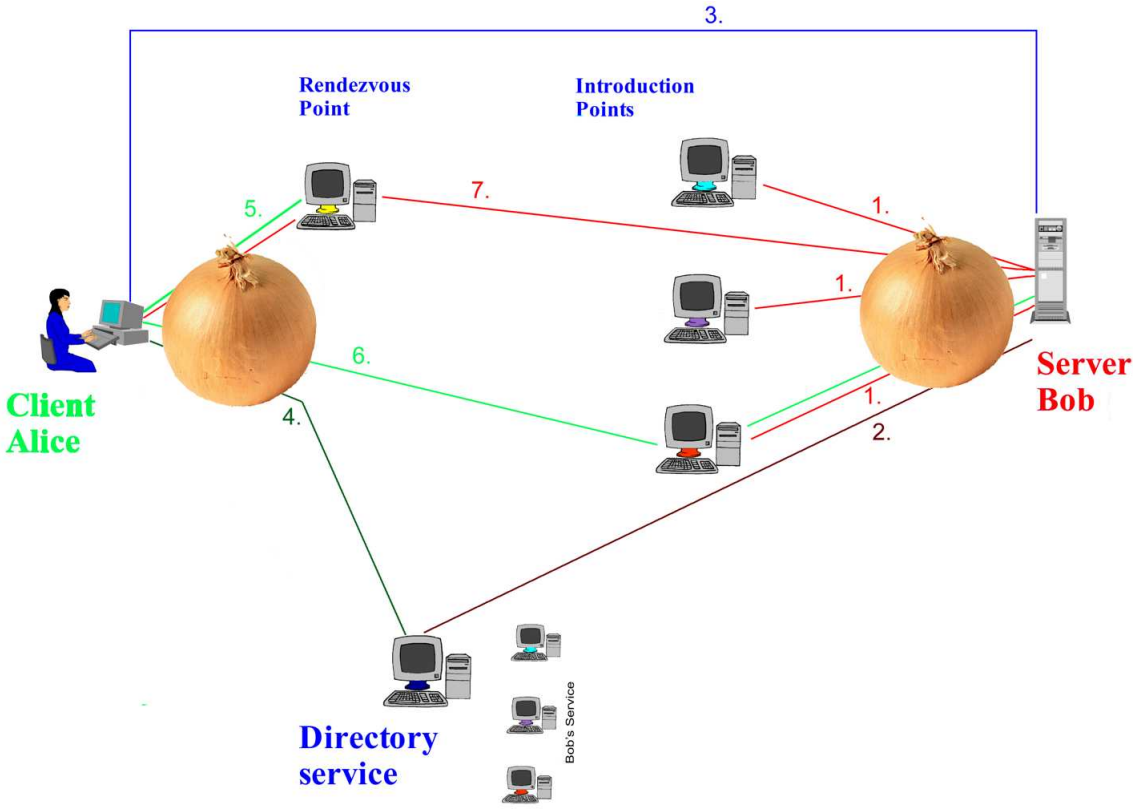

Abbildung 8: Darstellung des Ablaufes der Einrichtung und Anfrage an einen Hidden Service in Tor (angelehnt an: [7] o.S.)

1. Der Dienstanbieter (Bob) generiert zunächst ein asymmetrisches Schlüsselpaar sowie einen Hashwert des öffentlichen Schlüssels, der den Service eindeutig identifiziert. Dann werden mehrere Router als Introduction Point aus einer Liste von verfügbaren Routern ausgewählt (diese Liste wird durch das Directory verwaltet) und ein Pfad inkrementell zu jedem Introduction Point aufgebaut. Diese warten fortan auf Anfragen für den Service. Es ist notwendig, mehrere Introduction Point auszuwählen, da die Verfügbarkeit des Services gestört werden kann, wenn einzelne Introduction Point nicht erreichbar sind.

2. Bob legt eine Liste der Introduction Points sowie weitere notwendige Informationen (z.B. öffentlicher Schlüssel von Bob und Zeitstempel) bei dem Directory Service als sogenannten „Rendezvous Service Descriptor“ ab. Diese Informationen werden signiert mit dem privaten Signierschlüssel von Bob. Eine Verbindung zum Directory Service sollte via Tor erfolgen, um Bobs Adresse auch hier zu schützen. Der Hidden Service ist nun von außen erreichbar.

3. Der anfragende Client (Alice) erhält außerhalb des Tor-Netzwerkes eine Adresse (`[x.y.]z.onion`), die den Dienst eindeutig identifiziert. Dies kann beispielsweise über eine anonyme Veröffentlichung der Onion-Adresse auf einer anderen Webseite erfolgen.

4. Alice fragt den Directory Service an, um die notwendigen Informationen (Introduction Points, Schlüsselinformationen etc.) in Form des Rendezvous Service Descriptors zum zugehörigen Dienst zu erhalten. Eine Anfrage an den Directory Service sollte auch hier über Tor erfolgen, wenn Alice anonym bleiben will.

5. Im nächsten Schritt wählt Alice, ebenso wie Bob, einen Router aus einer Liste von verfügbaren Routern aus und baut einen Pfad inkrementell zu dem als Rendezvous Point fungierenden Router auf. Datenpakete, die an dem Rendezvous Point für Alice eintreffen, werden über den initialisierten Pfad weitergeleitet.

6. Über das Tor-Netzwerk baut Alice eine Verbindung zu einem der von Bob angegebenen Introduction Point auf. Alice agiert im Schutz des Tornetzwerkes anonym. Der Introduction Point leitet die Anfrage auf dem bereits initialisierten Pfad weiter an Bob. Eine Authentifizierung / Autorisierung könnte an dieser Stelle erfolgen, wird jedoch bisher nicht unterstützt.

7. Bob erhält die Anfrage von Alice über das Tor-Netzwerk und kann nun entscheiden, ob er eine Verbindung zu Alice über den Rendezvous Point auf Grund der von Alice gelieferten Informationen aufbauen will. An dieser Stelle wäre ebenfalls eine Authentifizierung / Autorisierung von Alice möglich. Der Rendezvous Point leitet das Datenpaket von Bob weiter an Alice, die über den so aufgebauten Pfad mit Bob anonym kommunizieren kann.

Nachdem nun die Funktionsweise von Hidden Services (Einrichtung und Anfrage) in Tor näher beleuchtet wurde, soll abschließend ein Augenmerk auf die Architekturziele des Hidden Service-Konzeptes (vgl. [11] S. 8 f.) und inwieweit diese bisher umgesetzt wurden gelegt werden. Da Internetdienste meist langfristig unter der gleichen Adresse erreichbar sein müssen, ist die *Robustheit* des Systems unabdingbar. Dazu ist es wichtig, dass ein Hidden Service migrierbar ist und somit langfristig unter der gleichen Onion-Adresse erreichbar ist. Gleichzeitig muss dabei die Anonymität des Anbieters langfristig gewahrt bleiben. Während eine Migration im Tor-Netzwerk sehr einfach erreichbar ist, scheint das Wahren einer langfristigen Anonymität im Hinblick auf mögliche Verkehrsanalysen in Tor (vgl. [47]) und im Speziellen bei Hidden Services (vgl. [12]) bisher noch nicht komplett möglich zu sein. *Anwendungstransparenz* als Entwurfsziel soll erreichen, dass sowohl auf Client- als auch auf Dienstanbieterseite die Integration in eine bestehende Anwendungsumgebung mit möglichst geringen Änderungen durchführbar ist. Auf Clientseite wird dies durch einen SOCKS-Proxy (Sockets Secure) erreicht, der alle Anfragen weiterleitet. Über die Onion-Adresse werden alle notwendigen Informationen an den Tor-Proxy auf Clientseite weitergeleitet. Alle Anwendungen, die das SOCKS-Protokoll unterstützen, können somit ohne große Änderungen Hidden Services benutzen. Auf Dienstanbieterseite werden die Anfragen vom Tor-Proxy direkt an den jeweiligen Port, auf dem der Dienst erreichbar ist weitergeleitet. Änderungen an der Konfiguration des Dienstes selber sind nicht notwendig. Dadurch, dass ein Introduction Point keine Verbindung zu einem Client aufnimmt und auch keinerlei Daten an den Client sendet, sondern diese Verbindung über den Rendezvous Point aufgebaut wird, ist das Ziel der *Verleumdungsresistenz* erreicht. Ein Introduction Point wird nicht für den Inhalt, den der Service anbietet (z.B. verbotene Internetseiten), verantwortlich gemacht, da eine Antwort des Dienstes nicht über den Introduction Point erfolgt. Diese Architektur der doppelten Umleitung über Introduction Point und Rendezvous Point unterscheidet Tor von den anderen hier untersuchten Architekturen und überlässt es dem Dienst selbst, auf eine Anfrage zu reagieren.

Ein letztes Designziel von Tor zielt auf die Möglichkeit einer *Zugangskontrolle* für den Dienstanbieter ab. Dadurch soll dem Anbieter eines Dienstes ermöglicht werden, unterschiedliche Dienstgüteklassen bereit zu stellen, bzw. soll ein Überfluten des Dienstes und damit ein Angriff auf die Verfügbarkeit unterbunden werden. Während die Ziele

bereits 2004 klar definiert wurden, fehlt es an einer Spezifizierung und Umsetzung von Authentifizierungs- und Autorisierungsmechanismen vollends. Vorgeschlagen werden sog. „authorization token“, mit denen sowohl auf Ebene der Introduction Points als auch auf Ebene des Dienstanbieters eine Zugangskontrolle stattfinden soll. Dazu soll die Onion-Adresse um zwei Elemente erweitert werden. Eine qualifizierte Adresse hat dann die Form `x.y.z.onion`. `x` beschreibt dabei das Token für den Zugriffsschutz auf Ebene des Introduction Point, `y` das Token auf Ebene des Dienstes und `z` identifiziert eindeutig den Service anhand eines Hashwertes des öffentlichen Schlüssels. Bisher ist ausschließlich die eindeutige Identifizierung des Dienstes umgesetzt. In den entsprechenden Spezifikationen ist zwar ein Hinweis auf die Zugangskontrolle gegeben, näher erläutert ist hierzu jedoch nichts (vgl. [46] o.S.). Daher kann eine Analyse und Bewertung der geplanten Zugangskontrolle an dieser Stelle nicht stattfinden. Vielmehr soll im nächsten Kapitel diese Lücke in der Spezifikation gefüllt werden, indem verschiedene Authentifizierungsverfahren daraufhin untersucht werden, inwieweit ein Einsatz im Tornetzwerk realisierbar ist.

4.4 Peer-to-Peer Systeme

Während die bisher betrachteten Systeme Tor und Rewebber auf einer Client-Server-Architektur basieren, sollen in diesem Unterkapitel die zwei Systeme Tarzan und I2P betrachtet werden, die auf einer Peer-to-Peer-Architektur beruhen. Der Unterschied zu bisher vorgestellten Systemen besteht insbesondere darin, dass alle Systemnutzer (sowohl Nachrichtenempfänger als auch Sender) Teil des Anonymisierungsnetzes (sog. Peers) sind und sowohl anonym Nachrichten senden und empfangen können als auch Nachrichten anderer Peers weiterleiten. Dieses Prinzip wurde bei Tor teilweise umgesetzt, da Clients auch als Onion-Router auftreten *können.* Weiterhin wird bei den beiden vorgestellten Peer-to-Peer-Architekturen keinerlei zentrales Management oder zentrale Kontrollinstanz benötigt, wie dies beispielsweise bei Tor durch die zentralen Directory Server der Fall ist.

Tarzan

Tarzan[3], das auf Arbeiten von Freedman et al. (vgl. [13] und [48]) beruht, benutzt, wie alle bisher vorgestellten Verfahren auch, das in Kapitel 3.4 beschriebene Mixprinzip, um Anonymität zu erzeugen. Dabei werden jedoch nicht festdefinierte Router oder Kaskaden von Routern durch die Nutzer als Mixinstanzen ausgewählt, sondern es erfolgt eine Auswahl aus einem Pool von Tarzan-Peers. Dadurch werden sog. Ende-zu-Ende-Angriffe, die insbesondere bei der beschriebenen Tor-Architektur möglich sind, stark erschwert, da nicht bestimmt werden kann, ob ein Vorgängerknoten der Initiator der Verbindung ist oder lediglich die Anfrage weitergeleitet hat.

Der typische Ablauf einer anonymen Verbindung erfolgt in drei wesentlichen Schritten: Zunächst wählt der Initiator der Verbindung aus einer Liste verfügbarer Knoten im Netzwerk (diese Liste ist anders als bei Tor nicht zentral gespeichert, sondern wird dezentral verwaltet) eine Menge von Weiterleitungsknoten aus. Im Folgenden wird ein sog. Tunnel durch das Netzwerk aufgebaut, wobei der Verkehr über die ausgewählten Knoten geleitet wird. Dieser Tunnelmechanismus ähnelt sehr stark dem Aufbau eines Pfades in Tor. Zum Schluss werden die Datenpakete entlang dieses Pfades weitergeleitet, der letzte Knoten

[3] http://pdos.csail.mit.edu/tarzan/index.html [Stand: 2006.09.27]

dieses Pfades leitet das Paket an den eigentlichen Empfänger (z.B. ein Webserver) unverschlüsselt weiter. Dieser letzte Knoten, der sog. Network Address Translator (NAT) stellt also den Kontakt mit dem Empfänger außerhalb des Tarzan-Netzes her und nimmt auch Antworten entgegen. Jeder Peer-Knoten kann als NAT fungieren (vgl. [13] S. 195 und [48], S. 121 f.).

Bisher wurde beschrieben, wie ein Szenario aussehen kann, bei dem der Empfänger einer Nachricht sich außerhalb des Tarzan-Netzwerkes befindet und der Sender ein Peer-Knoten ist. Der umgekehrte Fall, das Erzeugen von Empfängeranonymität, ist ebenfalls sehr einfach möglich. Hierzu baut ein Dienstanbieter, wie oben beschrieben, einen Tunnel bis zu einem NAT auf. Die Adresse des NAT kann nun als pseudonyme Dienstadresse veröffentlicht werden. Treffen Anfragen bei dem NAT ein, werden diese über den zuvor aufgebauten Tunnel an den Dienstanbieter weitergeleitet (vgl. [13] S. 196). Der NAT fungiert in dieser Weise wie der in Kapitel 4.1 beschriebene Proxy. Die hier getroffenen Aussagen bezüglich des Einsatzes von Authentifizierungsverfahren treffen also auch auf Tarzan zu. Ein mögliches Problem ist, dass der NAT für den Inhalt, den ein anonymer Dienst anbietet, verantwortlich gemacht werden könnte, da ein Client zunächst den NAT als Anbieter des Services identifiziert. Dies wird bei Tor mit Hilfe der doppelten Umleitung über einen Rendezvous Point vermieden. Der Vorteil des NAT-basierten Ansatzes besteht im Wesentlichen darin, dass ein Client nicht Teil des Tarzan-Netzwerkes sein muss, um einen anonymen Dienst nutzen zu können, wie dies bei Tor der Fall ist.

I2P - Invisible Internet Project

I2P (Invisible Internet Project) ist ebenso wie Tarzan ein Peer-to-Peer-Ansatz, der auf dem Grundprinzip eines Mixnetzwerkes beruht. Die folgenden Betrachtungen der Funktionsweise sind der Dokumentation auf der Webseite des Projektes entnommen (vgl. [8]). Weitere veröffentlichte Arbeiten über I2P konnten nicht ausgemacht werden. Das System beruht auf dem in Kapitel 3.4.3 beschriebenen Prinzip des Garlic-Routings, um Nachrichten innerhalb des Netzwerkes anonym weiterzuleiten. Ebenso wie Tor und Tarzan werden dazu Tunnel oder Pfade initialisiert, d.h. es werden Peer-Knoten ausgewählt, über die die schichtweise verschlüsselten Nachrichten weitergeleitet werden. Ein Tunnelaufbau erfolgt ähnlich der Vorgehensweise bei Tor, wenn auch hier die Initialisierung in einem einzelnen Schritt durch eine Nachricht erfolgt und spezielle Auswahlmechanismen für die Peer-Knoten eingesetzt werden. Im Gegensatz zu den beiden vorgenannten Systemen setzt I2P jedoch auf unidirektionale Kanäle, d.h. es werden für einen Peer-Knoten jeweils eingehende und ausgehende Tunnel aufgebaut, je nachdem, ob in diesem Tunnel Nachrichten zu einem Peer geliefert werden oder ob sie von einen Peer gesendet werden. Abbildung 9 verdeutlicht die Architektur von I2P.

Um nun auf einen Dienst über das I2P-Netzwerk anonym zugreifen zu können, müssen sowohl der Dienstnutzer als auch der Dienstanbieter als Knoten im I2P-Netzwerk integriert sein. Ein Zugreifen auf den Dienst von außerhalb des Netzwerkes, wie dies bei Tarzan über die NAT möglich ist, ist nicht vorgesehen. Jeder Peer-Knoten im Netzwerk besitzt wie abgebildet eine Anzahl von eingehenden und ausgehenden Tunneln, wobei der erste Router bei einem eingehenden Pfad als Gateway eine Proxyfunktionalität übernimmt. Die Tunnelgateways werden in einer Netzwerkdatenbank in einem sog. „LeaseSet“ veröffentlicht. So ist es jedem Peer möglich, über einen eingehenden Tunnel einen anderen Peer zu erreichen. Zusätzlich zu den Tunnelendpunkten werden im LeaseSet noch Schlüssel-

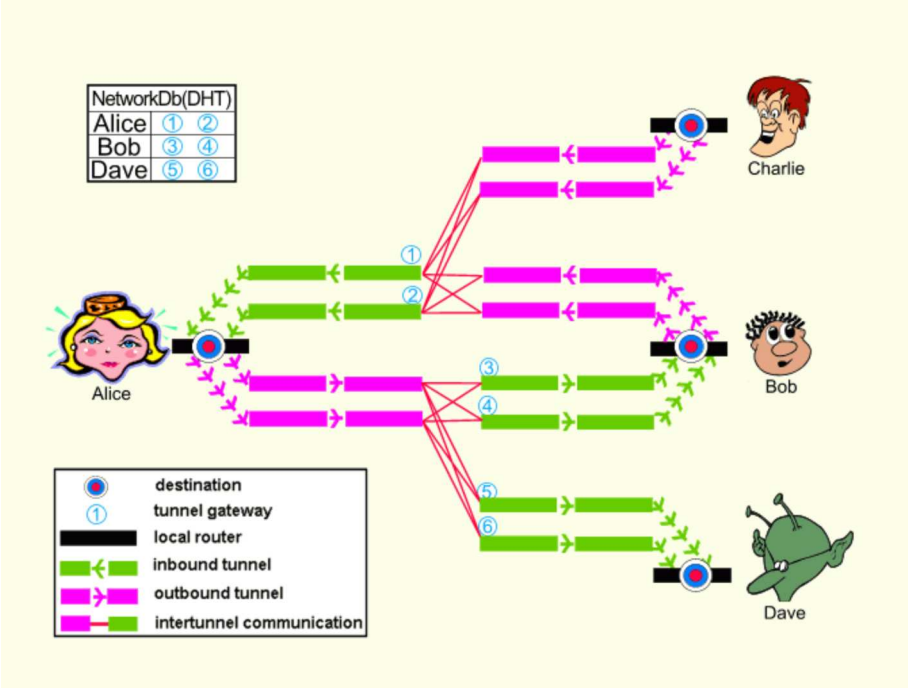

Abbildung 9: Darstellung von Kommunikationsbeziehungen in I2P (aus: [8] o.S.)

informationen und Zeitstempel für die Gültigkeit des Tunnels veröffentlicht. Ähnlich wie bei Tarzan werden in der dezentralen Netzwerkdatenbank, die als verteilte Hashtabelle realisiert ist, zusätzlich noch Listen aller verfügbaren Router und deren Kontaktinformationen gespeichert. Dies ist notwendig, damit bei einer Pfadinitialisierung die verfügbaren Router ausgewählt und kontaktiert werden können. Die Netzwerkdatenbank übernimmt somit die Funktionalität der zentralen Directory Server bei Tor.

Ähnlich wie bei Tor sind auch die im I2P-Netzwerk angebotenen Dienste in einem eignen Namensraum erreichbar. Diese sog. *eepsites* mit dem Suffix `i2p` sind vergleichbar mit den in Tor erwähnten Onion-Adressen. Sie werden beim Einrichten des Dienstes durch den Dienstanbieter festgelegt. Wie genau eine Adressauflösung im I2P-Netzwerk geschieht, ist der Dokumentation nicht entnehmbar. Um jedoch auf eine eepsite zugreifen zu können, muss ein Client zunächst die Adresse in einen Hashwert der öffentlichen Schlüssel des Empfängers auflösen. Unter diesem Hashwert ist das oben bereits erwähnte LeaseSet, das die Tunnelgateways angibt, in der verteilten Hashtabelle, der Netzwerkdatenbank, gespeichert. Über die eigenen ausgehenden Tunnel kann der Dienstnutzer nun eine Anfrage an einen Tunnelgateway des Empfängers senden. Dieser leitet die Anfrage über den eingehenden Tunnel weiter an den Dienstanbieter. Um auf Anfragen antworten zu können, ohne zunächst auf die verteilte Netzwerkdatenbank zugreifen zu müssen, kann der Client das eigene LeaseSet der Anfrage beifügen. Somit ist es möglich, dass der Dienstanbieter nach der beschriebenen Vorgehensweise auf die Anfrage des Clients reagiert. Ein Authentifizierungs- und Autorisierungsmechanismus, wie dies bei den Hidden Services von Tor vorgesehen ist, findet in der Dokumentation von I2P keine Erwähnung. Da jedoch der Tunnelgateway als als eine Art Proxy fungiert, wie dies in Kapitel 4.1 beschrieben wurde, treffen die Annahmen bezüglich des Einsatzes von Authentifizierungsverfahren auch auf I2P zu.

4.5 Fazit

In diesem Kapitel wurden die vier Systemarchitekturen Rewebber, Tor, I2P und Tarzan näher betrachtet und verglichen. Im Bezug auf die Umsetzung von Anonymisierungsverfahren ist anzumerken, dass in keinem der beschriebenen Systeme Authentifizierungsverfahren implementiert sind. Lediglich bei Tor sind Zugangskontrollen als Architekturziel definiert, eine konkrete Umsetzung oder Spezifikation dieses Ziels fehlt jedoch völlig. Es konnte gezeigt werden, dass alle vier vorgestellten Verfahren auf einem proxybasierten Ansatz beruhen und es somit theoretisch bei allen Verfahren möglich ist, diese um einen Authentifizierungsmechanismus auf Netzwerkebene zu erweitern. Auf Grund der Tatsache, dass die Entwicklung von Rewebber und Tarzan eingestellt wurde und für Rewebber auch keine Implementierung verfügbar ist, sollen diese beiden Systeme bei einer Auswahl für die Umsetzung außer Acht gelassen werden. Tor bietet gegenüber I2P den Vorteil, dass eine ausführliche Spezifikation des gesamten Projektes und Sourcecodes verfügbar ist und Zugangskontrollverfahren bereits bei den Designzielen berücksichtigt werden. Mit zur Zeit knapp 800 Routern und einer Kapazität von 200 Megabyte/s[4] stellt das Tornetzwerk außerdem eine stabile Infrastruktur bereit. In 2005 wurde das Tor Projekt von PC World in den Kreis der 100 besten Produkte gewählt. Aus diesen genannten Gründen soll eine Umsetzung des ausgewählten Authentifizierungsverfahrens im Tor-Projekt erfolgen.

[4] http://www.noreply.org/tor-running-routers [Stand: 2006.09.27]

5 Authentifizierungsverfahren zum Schutz von anonymen Diensten

Nachdem die theoretischen Grundlagen dieser Arbeit erörtert wurden und auf verschiedene Anonymisierungsprinzipien und -systeme eingegangen wurde, sollen in diesem Kapitel verschiedene Authentifizierungsverfahren und -konzepte untersucht werden, die für eine Authentifizierung bei Hidden Services im Tor-Netzwerk infrage kommen könnten. Wie bereits dargelegt, fehlen in den entsprechenden Spezifikationen zu den Hidden Services (vgl. [46] o.S.) jegliche nähere Beschreibung von möglichen Authentifizierungsverfahren, obwohl in den entsprechenden Designdokumenten zu Tor (vgl. [11] S. 8 f.) eine Zugangskontrolle als ausdrückliches Architekturziel definiert wurde. Diese fehlende Spezifikation soll in diesem Kapitel mit Inhalt gefüllt werden. Dazu ist es insbesondere notwendig, dass zunächst verschiedene Anforderungen an das Verfahren aufgestellt werden, anhand derer die Tauglichkeit der unterschiedlichen Authentifizierungsmechanismen beurteilt werden soll. In einem abschließenden Fazit soll ein Authentifizierungsverfahren ausgewählt werden, dass für die Implementierung im Tor-Netzwerk besonders geeignet erscheint.

5.1 Anforderungen an Authentifizierungsverfahren

In diesem Abschnitt sollen zunächst die Anforderungen, die an ein Authentifizierungsverfahren für Hidden Services in Tor gestellt werden, erläutert werden, bevor in den anschließenden Unterkapiteln die einzelnen Authentifizierungsverfahren vorgestellt und anhand der hier aufgestellten Anforderungen bewertet werden. Dabei ist die Bewertung der sechs Kriterien nicht gleich gewichtet: Während die ersten beiden Anforderungen bei Nichterfüllung zu einem Ausschluss des Verfahrens führen, sollten die vier folgenden innerhalb festgelegter Grenzen bestmöglich durch die Authentifizierungsverfahren erfüllt werden.

Authentifizierungsmöglichkeit durch Dritte In diesem Abschnitt wird auf die zentrale funktionale Anforderung des Verfahrens eingegangen. Danach soll das Verfahren die Möglichkeit bieten, dass sowohl auf Netzwerkebene durch Dritte (in diesem Fall durch den Introduction Point) eine Authentifizierung möglich ist als auch auf der Ebene des Dienstanbieters eine Authentifizierung eines Dienstnutzers erfolgen kann. Eine Authentifizierung auf Netzwerkebene beinhaltet den großen Vorteil, dass nicht-berechtigte Nutzer bereits durch den Introduction Point abgewiesen werden. Somit werden Verkehrsanalysen, eines der mächtigsten Werkzeuge, um die Anonymität eines Dienstanbieters zu gefährden, auf der Strecke zwischen Introduction Point und Hidden Service für nicht-berechtigte Nutzer unterbunden. Ebenso können Angriffe auf die Verfügbarkeit des Dienstes bereits im Netzwerk abgewehrt werden. Der Introduction Point kann somit die Rolle einer Firewall übernehmen, die bereits im Netzwerk nicht-berechtigte Anfrage unterbinden kann. Mit dieser funktionalen Hauptanforderung sind weitere Detailanforderungen verbunden. So ist Voraussetzung für das wirksame Unterbinden von Verkehrsanalysen und Angriffen auf die Verfügbarkeit eines Dienstes, dass ein Authentifizierungsverfahren durch den Introduction Point zum Zeitpunkt der Identitätsprüfung *keine* Interaktion mit dem Hidden Service erforderlich macht. Wären beispielsweise Rückfragen durch den Introduction Point zur Authentifizierung erforderlich, so könnten diese sehr einfach korreliert werden, und das Ziel des Unterbindens von Verkehrsanalysen wäre nicht erreicht. Da ein Introduction Point

als potentiell nicht-vertrauenswürdig eingestuft wird und dennoch eine Authentifizierungsaufgabe übernimmt, lässt sich als weitere Anforderung an das Verfahren formulieren, dass der authentifizierende Dritte keine Kenntnis von der privaten Information (beispielsweise ein Nutzername und ein zugehöriges Passwort) erhält, die der berechtigte Client als Nachweis der Identität erbringt. Ist dies nicht gewährleistet, kann die private Information durch den Introduction Point missbräuchlich verwendet werden. So wäre es möglich, dass ein nicht-berechtigter Dritter einen Dienst nutzen könnte.

Zweistufige Authentifizierung Auf Ebene der Dienstanbieter selbst ist eine zusätzliche Überprüfung der Berechtigung des Anfragenden notwendig, da ein Introduction Point nicht als vertrauenswürdig eingestuft werden sollte. Es ist also durchaus anzunehmen, dass ein Introduction Point korrupt ist und sämtliche Anfragen, auch von nicht-berechtigten Clients, weiterleitet. Wäre also eine Authentifizierung auf Ebene der Introduction Point das einzige mögliche Mittel der Zugangskontrolle, so wäre es durchaus möglich, dass nicht-berechtigte Clients auf einen Dienst zugreifen können. Dies soll unter keinen Umständen möglich sein. Weitere Anforderung an das Verfahren soll sein, dass dieser zweite Teil der Authentifizierung zwischen Client und Hidden Service keine zusätzliche Interaktion zwischen den beiden Parteien erforderlich macht. Lediglich die eigentliche Anfrage, die der Introduction Point an den Hidden Service weiterleitet, soll zur Authentifizierung ausreichen. Dies verhindert, dass durch zusätzlichen Nachrichtenverkehr Verkehrsanalysen ermöglicht werden. Abbildung 10 verdeutlicht den Ablauf der zweistufigen Authentifizierung. Während zwischen Introduction Point und Client (C) eine begrenzte Anzahl von Nachrichten (Nachricht 1-3) möglich sind, soll dem Hidden Service eine Authentifizierung mit einer einzigen Nachricht (Nachricht 4) ermöglicht werden. Eine Authentifizierung des

Abbildung 10: Darstellung einer zweistufigen Authentifizierung

Hidden Service gegenüber dem anfragenden Client soll nicht stattfinden, da eine Onion-Adresse von einem Angreifer, der den privaten Schlüssel des Services nicht kennt, nicht übernommen werden kann. Eine entsprechende Konsistenz-Überprüfung nimmt der Directory Server vor und kann auch vom Client selbst vorgenommen werden. Wenn ein Client eine Onion-Adresse von einem Dienstbetreiber erhält, kann er davon ausgehen, dass eine Anfrage an diese Adresse auch auf zugehörigen Hidden Service weitergeleitet wird bzw. dieses auch selbst durch durch Verifikation der Signatur des Rendezvous Service Descriptors (RSD) prüfen.

Einfluss auf die Anonymität Da es Sinn und Zweck eines Anonymisierungsnetzwerkes ist, Nutzern einen gewissen Grad an Anonymität zu gewährleisten, soll der Einfluss des Authentifizierungsverfahrens auf die Anonymität von Dienstnutzern und Dienstanbietern anhand dieser Anforderung untersucht werden. Um Grenzen zu definieren, die für eine Beeinträchtigung der Anonymität noch akzeptabel erscheinen, sollen die in Tabelle 1 dargestellten Anonymitätsklassen zu Hilfe genommen werden.

Auf Seiten des Dienstanbieters sollen keine Einschränkungen der Anonymität notwendig sein, was bedeutet, dass der Hidden Service auch bei Anwendung von Authentifizierungsverfahren den Dienst pseudonym anbieten kann. Der Hidden Service ist unter einem Pseudonym (der Onion-Adresse) erreichbar – es herrscht also unkontrollierte, personenbezogene

Pseudonymität. Für den Dienstnutzer soll die Anonymität gegenüber allen Parteien (Introduction Point und außenstehende Dritte) mit Ausnahme des Hidden Service möglichst wenig beeinträchtigt werden. Es soll hier also ein möglichst hoher Grad an Anonymität durch das Verfahren gewährleistet werden. Als Grenzwert ist akzeptabel, wenn ein Nutzer pseudonym gegenüber dem Introduction Point und Dritten auftreten kann. Pseudonym bedeutet in diesem Zusammenhang, dass kontrollierte oder unkontrollierte Pseudonymität herrschen kann, bei der der Hidden Service ggf. die Kontrollinstanz darstellt und bei der Transaktionen im Bezug auf *einen* Hidden Service korreliert werden können, bezogen auf unterschiedliche Hidden Services jedoch keine Verknüpfung von Transaktionen stattfinden kann. In Bezug auf das Verhältnis zwischen Hidden Service und Dienstnutzer soll es dem Hidden Service auf jedem Fall möglich sein, durch eine Zuordnung von Transaktionen zu Pseudonymen (hier herrscht dann unkontrollierte, relationsbasierte Pseudonymität) bzw. zu Identitäten (je nachdem, ob bei einem Schlüsselaustausch auch Identitätsinformationen ausgetauscht wurden) ein Fehlverhalten von Dienstnutzern zu erkennen und entsprechende Gegenmaßnahmen (beispielsweise Ausschluss des Nutzers) einleiten zu können.

Overhead des Verfahrens Ein Authentifizierungsverfahren soll auf Grund der möglichst geringen Latenzzeit, die ein echtzeitfähiges Anonymisierungsnetzwerk wie Tor bereitstellen soll, zunächst möglichst geringe Anforderungen an den Bedarf an Rechenzeit stellen. Dies ist insbesondere wichtig, da durch eine erhöhte Prozessorlast, beispielsweise in Folge von komplexen mathematischen Operationen, die Reaktionszeit von Introduction Points bzw. Dienstanbietern erhöht wird. Eine erhöhte Belastung beider Knoten kann einen Angriff auf die Verfügbarkeit des Dienstes begünstigen, insbesondere, wenn ein Introduction Point viele Anfragen an einen Dienst gleichzeitig verarbeiten muss. Des Weiteren soll die Anzahl der zusätzlichen Nachrichten möglichst gering gehalten werden. Es sollte möglich sein, die bisherigen Nachrichten in Tor, die bei einer Anfrage an einen Hidden Service zwischen Client und Introduction Point ausgetauscht werden, zu nutzen. Dabei sollte berücksichtigt werden, dass das Datenaufkommen durch zusätzliche Informationen, die bei einer Authentifizierung benötigt werden, möglichst gering gehalten wird. Zusätzliches Datenvolumen belastet das Tornetzwerk und führt unweigerlich zu einer Erhöhung der Latenzzeit und einer erhöhten Anfälligkeit für Angriffe auf die Verfügbarkeit eines Dienstes.

Infrastrukturbedingungen Hier sollen die Anforderungen an die Infrastruktur eines Anonymisierungsnetzwerkes zusammengefasst werden, die ein Authentifizierungsverfahren maximal stellen sollte. Hierbei ist es erforderlich, dass die bisher bestehende Infrastruktur des Tor-Netzwerkes so wenig wie möglich verändert werden sollte, um bei den Hidden Services das untersuchte Authentifizierungsverfahren umsetzen zu können. Insbesondere sollten keine zusätzlichen Anforderungen an das Vertrauen, das ein Dienstnutzer bzw. Dienstanbieter in andere Teilnehmer des Netzwerkes haben muss, erforderlich sein. Dazu zählen im speziellen Fall auch Introduction Point und Rendezvous Point. Ein gegenseitiges Vertrauen sollte lediglich zwischen Dienstanbieter und Dienstnutzer insofern bestehen, dass der Dienstnutzer seine Zugangsdaten nicht weitergibt und der Dienstanbieter nicht die Anonymität des Dienstnutzers aufdeckt. Hiermit verbunden sollte es nicht erforderlich sein, eine dritte Partei als sog. „Trust Center“ einsetzen zu müssen, das alle Teilnehmer des Netzwerkes als vertrauenswürdig einstuften. Das Einsetzen eines Trust Centers hat den Nachteil, dass diese zentrale vertrauenswürdige Instanz besonders Angriffen ausgesetzt ist und ein Ausfall dieser Instanz die Funktionsfähigkeit des gesamten Netzwerkes und damit

aller Dienste beeinträchtigen kann. Daher sollen Trust Center nicht eingesetzt werden. Auf Grund der Formulierung dieser Anforderung sollen eine Reihe von Authentifizierungsverfahren, die einen Trustcenter benötigen (bsp. Needham-Schroeder, Otway-Rees, Kerberos u.a) nicht Gegenstand der weiteren Betrachtung sein. Zusammenfassend kann gesagt werden, dass die Änderungen, die an der bestehenden Infrastruktur vorgenommen werden müssen, so gering wie möglich ausfallen sollen, um eine Abwärtskompatibilität zu älteren Versionen von Tor gewährleisten zu können.

Setup des Verfahrens Ein Initialisieren des Authentifizierungsverfahrens soll möglichst einfach möglich sein. Dies ist vor allem erforderlich, um eine Akzeptanz des Verfahrens zu gewährleisten. So soll es auch möglich sein, dass zwischen Dienstnutzern und Dienstanbietern ein Austausch der notwendigen Zugangsdaten anonym (beispielsweise über einen Chatroom) stattfinden kann. Dies macht es erforderlich, dass keine geheimen Zugangsdaten, sondern lediglich öffentliche Daten ausgetauscht werden müssen, bevor ein Dienst genutzt werden kann. Ein Setup des Verfahrens soll weiterhin ermöglichen, dass dynamische Benutzergruppen durch einen Dienst eingesetzt werden können. Konkret soll es möglich sein, einer Benutzergruppe neue Nutzer hinzuzufügen bzw. bestehende Nutzer aus der Gruppe zu entfernen (beispielsweise auf Grund von Fehlverhalten). Ein Re-Initialisieren des gesamten Systems (d.h. Schlüsselverteilung, Festlegen von Systemparametern etc.) soll dazu nicht notwendig sein.

5.2 Passwort-Verfahren

Traditionell werden zugriffsbeschränkte Systeme meist durch einfache Passwortprotokolle geschützt. Diese haben die Eigenschaft, dass zeitunabhängige Passwörter als Authentifizierungsmerkmale Verwendung finden. Menezes et al. bezeichnet diese Verfahren als „schwache Authentifizierungsmechanismen“ (vgl. [15] S. 388). Bei einem einfachen Passwort-Verfahren wird ein Nutzer eines Systems aufgefordert, einen Nutzernamen (ID) und ein zugehöriges Passwort (pw) anzugeben.

$$A \rightarrow B \quad : \quad ID, pw \tag{7}$$

Das System prüft daraufhin, ob die angegebene Identität berechtigt ist, auf die angeforderte Ressource zuzugreifen. Ein Beweis der angegebenen Identität erbringt der Nutzer über das mitgelieferte Passwort. Das System prüft, ob das bereitgestellte und das im System hinterlegte Passwort übereinstimmen und gewährt den Zugriff (vgl. [29] S. 143). Diese traditionelle Log-In Prozedur kann sehr einfach umgesetzt werden, da an der bestehenden Infrastruktur des Tor-Netzwerkes keinerlei Änderungen vorgenommen werden müssten.

Trotz der sehr einfachen Umsetzungsmöglichkeit in Tor erfüllt das Protokoll nicht die zentralen funktionalen Anforderungen an Authentifizierungsverfahren. So ist es nicht möglich, eine Authentifizierung durch Dritte durchzuführen, da ein u.U. nicht-vertrauenswürdiger Introduction Point, der die Aufgabe der Authentifizierung übernimmt, unweigerlich in den Besitz des Passwortes gelangt. So ist es dem Introduction Point selbst oder einem Dritten möglich, sich als Nutzer auszuweisen und somit unberechtigten Zugriff zu erlangen. Da die zentrale, funktionale Anforderung an das Verfahren nicht erfüllt werden konnte und dadurch eine Umsetzung nicht infrage kommt, soll auf eine Betrachtung der weiteren aufgestellten Anforderungen an dieser Stelle verzichtet werden.

5.3 Einmalpasswort-Verfahren

Einen ersten Schritt, um die Nachteile von schwachen Authentifizierungsmechanismen teilweise zu beseitigen und zu starken Verfahren zu gelangen, stellen die sog. Einmalpasswort-Verfahren dar. Wie im vorherigen Abschnitt dargestellt, spricht gegen eine Anwendung von traditionellen Passwort-Verfahren insbesondere die Möglichkeit, dass durch Abhören der Kommunikationsverbindung oder einfaches Wiederholen von Datenpaketen ein Angreifer unberechtigt Zugriff auf einen Dienst erlangen kann. Dies wird durch die Verwendung von Einmal-Passwörtern vermieden: Jedes Passwort wird ausschließlich für einen einzigen Zugriff auf den Dienst verwendet und verliert danach die Gültigkeit (vgl. [15] S. 395 f.). Menezes et al. unterscheiden drei Arten von Einmal-Passwörtern:

- *Eine Liste von Einmal-Passwörtern.* Der Dienstnutzer und der Dienstanbieter nutzen eine gemeinsame Liste von n Passwörtern, die lediglich einmal verwendet werden und vor der ersten Authentifizierung ausgetauscht werden müssen.
- *Eine kontinuierlich erneuerte Liste von Einmal-Passwörtern.* Hier wird bei jedem Authentifizierungsvorgang geheim ein neues Passwort für den den nächsten Zugriff festgelegt. Somit muss zu Beginn nur ein Passwort vereinbart werden.
- *Einmal-Passwörter, die mittels Einwegfunktionen erstellt werden.* Hier werden die Einmal-Passwörter durch Einwegfunktionen erstellt. Mit SKEY wird ein Authentifizierungsverfahren vorgestellt, dass auf diesem Grundprinzip beruht.

Im Folgenden soll ausschließlich auf die dritte Variante näher eingegangen werden, da eine praktische Umsetzung der beiden ersten Varianten auf Grund der Anforderung einer Authentifizierung durch Dritte nicht möglich erscheint. Eine nicht vertrauenswürdige dritte Instanz müsste die gesamte Passwortliste kennen (Variante 1) bzw. neue Passwörter aushandeln (Variante 2).

SKEY Authentifizierungsprotokoll

Das SKEY-Authentifizierungsprotokoll (vgl. [9] S. 53 und [29] S. 144 f.) nutzt zur Authentifizierung eine sichere Hashfunktion $h(x)$ und kann dadurch einige der Schwächen der traditionellen Log-In Protokolle beseitigen. Um ein Authentifizierungssystem zu -initialisieren, wird für jeden Nutzer vom Dienstanbieter eine Zufallszahl r als Geheimnis ausgewählt und diesem zugänglich gemacht. Gleichzeitig wird mittels der Hashfunktion die erste Benutzerkennung ermittelt. Dazu wird die Funktion eine fixe Anzahl von Durchläufen (beispielsweise 100) rekursiv auf die Zufallszahl angewandt($h(h...h(h(r))))$. Der resultierende Funktionswert (y_{100}) wird als Benutzerkennung beim Dienstanbieter und Dienstnutzer gespeichert. Der Dienstanbieter speichert das Geheimnis nicht weiter, sondern verwirft die Zufallszahl r. Soll nun ein Zugriff auf den Dienst erfolgen, dient y_{100} als Zugangskennung, das Passwort wird durch den Dienstnutzer rekursiv mittels der Zufallszahl r und der Hashfunktion ermittelt. Dazu wendet er die Funktion genau einmal weniger an als bei der aktuellen Zugangskennung und erhält so das Passwort y_{99}. Der Dienstanbieter kann sehr einfach überprüfen, ob das Passwort richtig errechnet wurde, indem er die Hashfunktion einmal auf das Passwort anwendet. Entspricht der Funktionswert der aktuellen Benutzerkennung ($h(y_{99}) = y_{100}$), kann ein Zugriff gewährt werden. Bei

einem nächsten Zugriff wird das vorherige Passwort als Benutzerkennung (y_{99}) verwendet und eine neues Passwort (y_{98}) muss errechnet werden. Der Vorteil dieses Verfahrens liegt darin, dass keine Informationen zu einem Passwort über das Netzwerk versendet werden oder durch eine authentifizierende Partei gespeichert werden müssen.

Die authentifizierende Partei muss keinerlei Kenntnis von der Zufallszahl r, die das vereinbarte Geheimnis darstellt, haben. Dadurch ist es mit Einschränkungen möglich, dass eine dritte Partei (beispielsweise der Introduction Point) eingesetzt wird, um die Authentifizierung durchzuführen. Einem Angreifer bzw. der authentifizierenden Partei ist es auf Grund der Einwegeigenschaft der Hashfunktion nicht möglich, aus einem erhaltenen Hashwert (y_k) das nächste gültige Passwort (y_{k-1}) oder gar das Geheimnis zu ermitteln. Einschränkungen bestehen dennoch insofern, dass zwischen den unterschiedlichen Introduction Point im Tor-Netzwerk eine Synchronisation stattfinden muss, damit jeder Introduction Point auf dem gleichen Stand bezogen auf das gerade aktuelle Passwort ist. Eine solche Synchronisation ist jedoch u.U. sehr umständlich und kann auf Grund von zeitlichen Differenzen zum Missbrauch des Protokolls führen. Ein Angreifer könnte eine Synchronisation blockieren und sich so Zugang zu einem System verschaffen, indem er ein altes Passwort verwendet. Die funktionalen Anforderungen bezüglich der Authentifizierung durch Dritte können also nur mit Einschränkungen erfüllt werden. Eine zweistufigen Authentifizierung kann sehr einfach über das Protokoll realisiert werden, da für einen Protokollablauf keine weitere Interaktion zwischen Client und authentifizierender Stelle stattfinden muss. Ebenso wie bei traditionellen Log-In-Protokollen sind beim Setup des Verfahrens Geheimnisse zwischen Dienstnutzer und Dienstanbieter auszutauschen, was ein gefordertes anonymes öffentliches Setup verhindert. Zu der Anonymität kann angemerkt werden, dass die geforderten Kriterien eingehalten werden. Ein Hidden Service kann insbesondere auf Grund des ausgetauschten Geheimnisses auch eine Transaktion einem Benutzer zuordnen. Zusätzliche Anforderungen an die Infrastruktur bestehen nicht. Ein Overhead für das Verfahren besteht insofern, dass ein Dienstnutzer eine u.U. rechenaufwändige Hashfunktion mehrere hundert Mal durchführen muss, um das aktuelle Passwort generieren zu können. Der zusätzliche Nachrichtenoverhead ist minimal.

5.4 Challenge-Response-Authentifizierung

„The idea of cryptographic challenge-response protocols is that one entity (the claimant) "proves" its identity to another entity (the verifier) by demonstrating knowledge of a secret known to be associated with that entity, without revealing the secret itself to the verifier during the protocol"([15] S. 397). Dies wird erreicht indem von der authentifizierenden Stelle dem Client eine sog. Challenge (Aufgabe) gestellt wird. Der Client antwortet auf diese Challenge und beweist dadurch seine Identität, da die Antwort sowohl von der vorherigen Challenge als auch von einem Geheimnis abhängig ist. Weder durch Wissen der Challenge noch der zugehörigen Antwort soll es einem Dritten möglich sein, Rückschlüsse auf die Identität des Client oder auf dessen Geheimnis zu ziehen (vgl. [15] S. 397). Challenge-Response-Protokolle können sowohl auf symmetrischen als auch auf asymmetrischen kryptografischen Primitiven beruhen. Beide Ansätze sollen im Folgenden näher betrachtet werden.

5.4.1 Verfahren basierend auf symmetrischer Kryptografie

Voraussetzung für den Einsatz von Challenge-Response-Verfahren, die auf symmetrischer Kryptografie basieren, ist, dass zunächst zwischen den beiden beteiligten Parteien ein Geheimnis (ein symmetrischer Schlüssel) ausgetauscht wurde. An dieser Stelle sollen lediglich Protokolle betrachtet werden, die unilaterale Authentifizierung bereitstellen und die keine vertrauenswürdige dritte Partei erfordern. Protokolle für multilaterale Authentifizierung werden nicht näher betrachtet, da ein Identitätsnachweis des Dienstanbieters bzw. des Introduction Point nicht als Anforderung an das Verfahren definiert wurde.

Im folgenden Abschnitt soll beispielhaft ein Protokoll vorgestellt werden, das auf der Grundlage symmetrischer Kryptografie beruht. Es stellt das SKID2 Protokoll (Secret-Key IDentification protocol) dar, das im Rahmen des RIPE (RACE Integrity Primitives Evaluation) Projektes entworfen wurde (vgl. [49] S. 171 - 178). Wie leicht ersichtlich ist, wird eine sichere Hashfunktion (h) eingesetzt, die einen Schlüssel (K) als Eingabeparameter benötigt. Solche speziellen Hashfunktionen errechnen den Hashwert aus dem Schlüssel und dem Eingabewert. Somit ist es nur möglich über den korrekten Schlüssel den passenden Hashwert zu bestimmen (vgl. [9] S. 31). Der Client A initiiert den Protokollablauf durch eine Leernachricht. Daraufhin erzeugt B eine Zufallszahl und übermittelt diese an A. Durch die Zufallszahl wird ein Wiederholungsangriff durch Wiedereinspielen der Nachricht über einen Angreifer verhindert. B akzeptiert lediglich Nachrichten, die eine „nichtverbrauchte“ Zufallszahl enthalten. A erzeugt eine Signatur aus einer eigenen Zufallszahl (r_A), der erhaltenen Zufallszahl (r_B) und der eigenen Kennung B. Um zu überprüfen, ob ein Client berechtigt ist, auf ein System zuzugreifen, muss die authentifizierende Stelle überprüfen, ob der Hashwert, den die Nachricht enthält, durch den Client korrekt gebildet wurde. Ist dies der Fall, hat der Client seine Identität nachgewiesen, da lediglich über den richtigen Schlüssel der passende Hashwert gebildet werden konnte.

$$A \rightarrow B \quad : \quad init \tag{8}$$

$$A \leftarrow B \quad : \quad r_B \tag{9}$$

$$A \rightarrow B \quad : \quad r_A, h_K(r_A, r_B, B) \tag{10}$$

Weitere Protokolle, die symmetrische Verschlüsselung einsetzen, sind denkbar. So kann die sichere Hashfunktion durch einen Verschlüsselungsalgorithmus ersetzt oder ein Wiedereinspielen von Nachrichten statt durch Zufallszahlen mit Zeitstempeln verhindert werden. Beide Verfahren werden im ISO/IEC Standard 9798-2 beschrieben (vgl. [50] zitiert nach [15] S. 401 f.), sollen hier aber nicht detaillierter betrachtet werden, da das Grundprinzip, das Gleiche ist, wie beim vorgestellten Verfahren.

Authentifizierungsverfahren, die auf symmetrischer Kryptografie beruhen, sollen nun dahingehend untersucht werden, inwieweit sie geeignet erscheinen, die in Kapitel 5.1 definierten Anforderungen zu erfüllen. Da ihnen allen gemein ist, dass eine Authentifizierung mittels symmetrischer Schlüssel erreicht wird, ist die funktionale Anforderung der Authentifizierung durch Dritte nicht realisierbar. Bei einer Authentifizierung wird bei allen drei Protokollen das Geheimnis, das zwischen Dienstanbieter und Dienstnutzer bei Aufnahme eines neuen Nutzers vereinbart wurde, benötigt. Eine Authentifizierung ist sonst durch den Introduction Point allein nicht möglich. So gelangt der Introduction Point unweigerlich in Kenntnis der Schlüssel und kann sich unberechtigterweise Zugriff auf den Dienst verschaffen bzw. einem Angreifer die Zugangsdaten bereitstellen. Dies macht den Einsatz von symmetrischen Authentifizierungsmechanismen unmöglich, da ein Introduction

Point als nicht-vertrauenswürdig eingestuft wird. Da die zentrale, funktionale Anforderung an das Verfahren nicht erfüllt werden konnte und dadurch eine Umsetzung nicht infrage kommt, soll auf eine Betrachtung der weiteren aufgestellten Anforderungen an dieser Stelle verzichtet werden.

5.4.2 Verfahren basierend auf asymmetrischer Kryptografie

Da gezeigt werden konnte, dass eine Authentifizierung mit Hilfe von symmetrischer Verschlüsselungstechnik nicht möglich ist, soll in diesem Kapitel untersucht werden, inwieweit dies mit asymmetrischen Verfahren möglich ist. Einem Client, der seine Identität beweisen will, bieten sich Menezes et al. zufolge (vgl. [15] S. 403 ff.) zwei unterschiedliche Weisen, dies zu tun:

1. Durch Entschlüsseln einer Challenge oder
2. durch Signieren einer Challenge.

Beide Verfahren sollen anhand eines Protokolls dargestellt und anschließend gemeinsam auf eine Entsprechung der gestellten Anforderung hin untersucht werden. Durch ein Leernachricht soll die Initiierung des Protokollablaufes der authentifizierende Stelle (A) übertragen werden.

Beim Challenge-Response-Verfahren, das auf dem Entschlüsseln einer Challenge beruht, generiert die authentifizierende Stelle zunächst eine Challenge für den anfragenden Client, indem sie eine Zufallszahl r mit dem öffentlichen Schlüssel vom Client A verschlüsselt. Zusammen mit dem durch eine Hashfunktion ermittelten Hashwert der Challenge ($h(r)$) wird diese Nachricht dem Client übermittelt. Der Client entschlüsselt mit dem eigenen privaten Schlüssel die Challenge und überprüft, ob der von A erhaltene Hashwert mit dem Hashwert der entschlüsselten Challenge übereinstimmt. Ist dies der Fall, antwortet der Client mit r auf die Challenge. Die authentifizierende Stelle überprüft, ob der erhaltene Wert mit der generierten Zufallszahl übereinstimmt. Ein Übermitteln des Hashwertes der Zufallszahl ist notwendig, um sog. chosen-text Angriffe auf die kryptografischen Primitive zu unterbinden (vgl. [15] S. 404).

$$A \rightarrow B \quad : \quad init \tag{11}$$
$$A \leftarrow B \quad : \quad h(r), B, P_A(r) \tag{12}$$
$$A \rightarrow B \quad : \quad r \tag{13}$$

Das Challenge-Response-Verfahren durch Signieren einer Challenge beruht darauf, dass ein sich authentifizierender Client eine Challenge der authentifizierenden Stelle mit dem eigenen privaten Schlüssel signiert. Dazu sendet die authentifizierende Stelle eine Zufallszahl (r_B) als Challenge, die der Client zusammen mit einer eigenen Zufallszahl (r_A) und einer ID von B mit dem eigenen privaten Schlüssel signiert. Zusammen mit einem sog. Zertifikat des öffentlichen Schlüssels ($cert_A$) und der eigenen Zufallszahl wird die Signatur der authentifizierenden Stelle übermittelt. Das Zertifikat beweist, dass ein öffentlicher Schlüssel an eine bestimmte Identität gebunden ist. Das Zertifikat kann bei einer Initialisierung des Verfahrens beispielsweise durch die authentifizierende Stelle selbst ausgestellt werden. Die authentifizierende Stelle überprüft mit Hilfe des öffentlichen Schlüssels aus dem Zertifikat

die erzeugte Signatur und kann so die Identität des Clients bestätigen und einen Zugriff gewähren. Dieses Verfahren, das durch den Austausch von Zufallszahlen einen Wiederholungsangriff verhindert, lässt sich, ebenso wie die symmetrischen Verfahren, sehr einfach auch durch Zeitstempel realisieren.

$$A \rightarrow B \quad : \quad init \tag{14}$$
$$A \leftarrow B \quad : \quad r_B \tag{15}$$
$$A \rightarrow B \quad : \quad cert_A, r_A, B, S_A(r_A, r_B, B) \tag{16}$$

Nachfolgend soll überprüft werden, inwieweit mit den beiden vorgestellten Verfahren, die auf asymmetrischer Kryptografie beruhen, die Anforderungen, die zu Anfang des Kapitels aufgestellt wurden, erfüllt werden können. Zunächst soll auf die funktionale Anforderung, der Möglichkeit einer Authentifizierung durch Dritte, eingegangen werden. Da ein Introduction Point zur Überprüfung der Identität mit Hilfe asymmetrischer Verfahren keine privaten Schlüssel oder Geheimnisse besitzen muss, sondern öffentliche Schlüssel verwendet, die er bei einer Initialisierung vom Dienstanbieter erhält, steht einer Authentifizierung durch Dritte nichts im Wege. Allein die Tatsache, dass eine Authentifizierung auch getrennt zwischen Dienstanbieter und Dienstnutzer direkt erfolgen muss, schränkt die Bandbreite der verfügbaren Verfahren ein. Alle Verfahren, die die Challenge-Response-Eigenschaft über Zufallszahlen erhalten, können hier nicht verwendet werden, da zwischen Dienstnutzer und Dienstanbieter auf Grund der geforderten zweistufigen Authentifizierung keine zusätzliche Interaktion stattfinden soll. Zeitstempel können statt der Zufallszahlen zum Einsatz kommen. Hierbei ist zu beachten, dass eine Synchronisation der Uhren stattfinden muss. Die Anonymität der Dienstnutzer wird durch das Verfahren gewahrt. Weder einem außenstehenden Dritten noch irgendeinem Knoten im Tor-Netzwerk ist es möglich, Identitätsinformationen durch den Mechanismus auszuspähen. Allein dem Dienstanbieter ist es möglich, ein verwendetes Pseudonym (hier verwendete Schlüsselpaare) aufzudecken. Es herrscht also kontrollierte Pseudonymität. Bezüglich der Anonymität des Dienstanbieters müssen, wie gefordert, keinerlei Einschränkungen vorgenommen werden. Erweiterte Infrastrukturanforderungen stellt das Verfahren nicht, auch scheint der Overhead des Verfahrens tragbar zu sein. Zusätzliche Nachrichten sind bei Verwendung von Zeitstempeln zunächst nicht erforderlich, jedoch wird eine Zeitsynchronisation benötigt. Diese Zeitsynchronisation könnte über die Directory Server laufen, was zusätzliche Nachrichten und eine Belastung der zentralen Instanzen mit sich bringt. Insgesamt stellen asymmetrische Verfahren höhere Anforderungen an Rechenzeit als symmetrische Verfahren. Da jedoch nur einzelne Operationen durchgeführt werden müssen, scheint dies vertretbar zu sein. Ein Setup des Verfahrens ist sehr einfach durch Austausch des öffentlichen Schlüssels des Dienstnutzer mit dem Dienstanbieter (auch anonym und öffentlich) möglich.

5.5 Zero-Knowledge-Protokolle

Es konnte in den vorherigen Kapiteln gezeigt werden, dass Challenge-Response-Verfahren gegenüber einfachen Passwort-Verfahren den Vorteil besitzen, dass sie ein Geheimnis nicht direkt an eine authentifizierende Stelle weiterleiten und somit ermöglichen, eine missbräuchliche Verwendung zu unterbinden. Beim Challenge-Response-Verfahren reagiert ein Client in geeigneter Weise auf eine Challenge der authentifizierenden Stelle und zeigt dadurch, dass er das Geheimnis kennt, ohne dass die authentifizierende Stelle oder ein Dritter die gelieferte Response direkt verwerten kann. Dennoch können durch

strategisch ausgewählte Challenges zum Teil Informationen über das Geheimnis erlangt werden. Zero-Knowledge-Protokolle setzen hier an, indem sie einem Client erlauben, eine authentifizierenden Stelle von der Kenntnis eines Geheimnisses zu überzeugen, ohne dass jegliche Information über das Geheimnis preisgegeben wird. Im Folgenden soll nun zunächst definiert werden, was unter einem Zero-Knowledge-Beweis verstanden werden soll und das Grundprinzip, das hinter allen Zero-Knowledge-Protokollen steckt erläutert werden, ehe auf die drei Authentifizierungsprotokolle von Feige-Fiat-Shamir (vgl. [51]), Guillou-Quisquater (vgl. [52]) und Schnorr (vgl. [53]) eingegangen wird. Diese drei Protokolle wurden deshalb ausgewählt, da sie die Literatur verzeichneten Standardprotokolle für eine Authentifizierung mit Hilfe von Zero-Knowledge-Beweisen darstellen. Zahlreiche Erweiterungen und Abwandlungen zu den drei hier vorgestellten Protokollen sind in der Literatur zu finden (vgl. [54], [55], [56], etc.), sollen in dieser Arbeit jedoch nicht im Detail betrachtet werden. Auf Grund der Gemeinsamkeiten der hier betrachteten Protokolle, soll in einem abschließenden Unterkapitel zunächst für Zero-Knowledge-Protokolle allgemein dargelegt werden, inwieweit die in Kapitel 5.1 aufgestellten Anforderungen an einen Authentifizierungsmechanismus erfüllt werden können, ehe anschließend auf einen Vergleich der drei Protokolle eingegangen wird.

Goldwasser et al. definieren in ihren Arbeiten zu interaktiven Beweisen Zero-Knowledge-Beweise wie folgt: *„Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question“* ([16] S. 186). Diese sehr einfache Definition von Zero-Knowledge-Beweisen soll im weiteren Arbeitskontext ausreichen.

Zero-Knowledge-Protokolle werden den sog. interaktiven Beweisen zugerechnet, wobei zwischen dem Beweisenden und dem Prüfer mehrere Nachrichten ausgetauscht werden, die normalerweise von Zufallszahlen abhängig sind. Allgemein haben ein Großteil der Zero-Knowledge-Protokolle dabei die folgende dreiteilige Nachrichtenstruktur (vgl. [15] S. 409):

$$A \rightarrow B \quad : \quad witness \tag{17}$$
$$A \leftarrow B \quad : \quad challenge \tag{18}$$
$$A \rightarrow B \quad : \quad response \tag{19}$$

Dabei stellt die „witness“ ein von der sich identifizierenden Partei (A) gewähltes Zufallselement dar, das einen Protokolllauf von jedem anderen unterscheiden soll. Die darauf folgende, von der authentifizierenden Stelle (B) zufällig gewählte „challenge“ wird an A übermittelt, die in Abhängigkeit beider Zufallszahlen mit Hilfe des privaten Schlüssels eine „response“ ermittelt. Diese wird an B übermittelt, die die Richtigkeit der Berechnung überprüfen kann und so A authentifiziert. Das Ziel dieser Nachrichten ist letztendlich, dass der Prüfende von der Richtigkeit einer Annahme überzeugt werden kann, beispielsweise der Annahme, dass der Beweisende ein Geheimnis kennt. Interaktive Beweise, die die Kenntnis eines bestimmten Geheimnisses für einen korrekten Ablauf voraussetzen, werden allgemein als „Proof of Knowledge“ bezeichnet. Diese Beweise müssen die Eigenschaften der Vollständigkeit (ein ehrlicher Beweisender hat mit sehr hoher Wahrscheinlichkeit auch Erfolg, den Prüfer zu überzeugen) und der Korrektheit (ein betrügerischer Beweisender hat nur mit vernachlässigbarer Wahrscheinlichkeit Erfolg) erfüllen. Ein „Proof of Knowledge“ ist dann zusätzlich auch ein Zero-Knowledge-Beweis, wenn durch den Protokollablauf sogar einem böswilligen Protokollteilnehmer keinerlei Informationen, außer den Informationen, die auch öffentlich zugänglich sind, preisgegeben werden (vgl. [15] S. 405 ff.). Eine

Unterscheidung zwischen „perfect zero-knowledge“, „computational zero-knowledge“ und „statistical zero-knowledge“, so wie dies von Brassard et al. (vgl. [57] S. 184 f.) vorgeschlagen wird, soll an dieser Stelle nicht getroffen werden, da dies den über Rahmen der Arbeit hinaus führen würde. Aus dem gleichen Grund soll eine sequentielle Verknüpfung von Zero-Knowledge-Protokollen außer Acht bleiben.

Vielmehr soll allgemein das Grundprinzip erläutert werden, das hinter Zero-Knowledge-Protokollen steckt. Unter dem Titel „How to Explain Zero-Knowledge Protocols to Your Children“ haben Quisquater et al. (vgl. [58]) die Funktionsweise von Zero-Knowledge-Beweisen mit Hilfe einer Geschichte über eine Höhle sehr anschaulich und leicht verständlich erklärt. Die in Abbildung 11 dargestellte Höhle besitzt einen verschlossenen Durchgang von C nach D, der mittels eines Geheimwortes geöffnet werden kann. Möchte

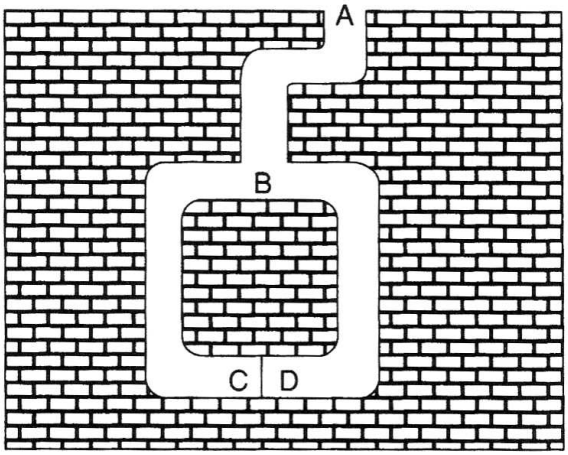

Abbildung 11: Die Zero-Knowledge-Höhle (aus: [9] S. 102)

eine Person (Peggy) nun einer zweiten Person (Victor) beweisen, dass sie im Besitz des Geheimnisses ist, ohne Victor das Geheimnis mitzuteilen, sollte ein Protokoll zum Einsatz kommen, das Zero-Knowledge-Eigenschaft besitzt. Dazu wäre das folgende Protokoll mehrmals zu durchlaufen:

1. Victor steht am Punkt A, während Peggy sich in die Höhle zu Punkt C oder D begibt.
2. Nachdem Peggy gegangen ist, geht Victor vor zu Punkt B und gibt Peggy an, auf welchem Weg sie aus der Höhle kommen soll, rechts oder links.
3. Peggy nutzt das Geheimnis, falls dies nötig ist, um den Durchgang zu öffnen, und erscheint am geforderten Ausgang.

Ein mehrmaliges Wiederholen des Protokolls verhindert, dass Peggy zufällig die richtige Seite gewählt hatte. Die Wahrscheinlichkeit, dass Peggy zufällig den richtigen Eingang gewählt hat, ohne das Geheimnis zu kennen, kann mit 1 zu 2^n angegeben werden und sinkt somit mit der Anzahl der Protokollrunden (n).

Anhand dieses einfachen Beispiels konnte gezeigt werden, was die in den folgenden Abschnitten beschriebenen Zero-Knowledge-Protokolle gemein haben: Es ist weder Victor noch einer anderen dritten Partei, selbst durch Aufzeichnungen des Protokollablaufes,

möglich etwas über das Geheimnis von Peggy zu erfahren. Das Protokoll ist also ein Zero-Knowledge-Protokoll. Weiteres Charakteristikum dieses interaktiven Protokolls ist, dass Victor nicht die Möglichkeit besitzt, eine dritte Partei davon zu überzeugen, dass Peggy das Geheimnis kennt, selbst dann nicht, wenn Victor den gesamten Protokollablauf zum Beispiel per Video aufzeichnen würde. Diese beiden Eigenschaften sollten bei einer Untersuchung auf die Verwendbarkeit von Zero-Knowledge-Protokollen bei der Authentifizierung im Tor-Netzwerk Berücksichtigung finden.

5.5.1 Feige-Fiat-Shamir-Authentifizierungsprotokoll

Das Zero-Knowledge-Authentifizierungsprotokoll von Feige, Fiat und Shamir (vgl. [51]) ist eine Erweiterung des Authentifizierungsprotokolls von Fiat und Shamir (vgl. [59]), das keine Zero-Knowledge-Eigenschaft besitzt. Beim Protokollablauf wird beim letzteren Mechanismus ein Bit an Information über das Geheimnis der sich authentifizierenden Partei preisgegeben. Die Sicherheit beider Verfahren beruht auf der sogenannten „Square-Root-Annahme“, der Schwierigkeit des Quadratwurzelziehens im Restklassenring $\mathbb{Z}_n$ bei unbekannter Faktorisierung von n. Im Folgenden soll lediglich das Zero-Knowledge-Protokoll von Feige, Fiat und Shamir näher betrachtet werden, indem auf die Systeminitialisierung (welche Schritte sind notwendig, wenn ein Authentifizierungssystem parametrisiert wird), die Nutzerinitialisierung (welche Schritte sind notwendig, wenn ein neuer Nutzer hinzukommt) und den eigentlichen Protokollablauf (welche Schritte sind bei einer Authentifizierung nötig) eingegangen wird.

Bei Initialisierung des Authentifizierungssystems muss durch eine vertrauenswürdige Instanz (diese Funktion kann der Hidden Service übernehmen, da diese Instanz nur bei der erstmaligen Initialisierung benötigt wird) einen gemeinsamen Modul veröffentlichen, der systemweit für alle Nutzer gilt. Dieser Modul n sollte das Produkt zweier großer Primzahlen der Form $4r + 3$ (sog. Blum Integers) sein (vgl. [51] S. 81). Systemweit werden ebenfalls die Sicherheitsparameter t und k festgelegt.

Bei der Nutzerinitialisierung müssen von jedem neuen Nutzer zunächst die geheimen und öffentlichen Schlüssel festgelegt werden. Dazu wählt jeder Nutzer k private Schlüssel $(S_1, ..., S_k)$ aus, indem er Zufallszahlen in $\mathbb{Z}_n^*$ generiert, die er geheim hält. Zu jedem privaten Schlüssel generiert der Nutzer den passenden öffentlichen Schlüssel J_i $(i = 1, ..., k)$ aus dem privaten Schlüssel wie folgt: $J_i = \pm 1/S_i^2 (\text{mod } n)$. Die Auswahl des Vorzeichens erfolgt zufällig und unabhängig.

Beim Protokollablauf (vgl. [15] S. 410 f.) werden drei Nachrichten zwischen der sich authentifizierenden Partei A und der authentifizierenden Stelle B ausgetauscht. Dazu wählt A zunächst eine Zufallszahl r aus $\mathbb{Z}_n^*$ sowie ein zufälliges Bit (b) aus. Daraufhin berechnet sie $x = (-1)^b \cdot r^2 \ (\text{mod } n)$ und sendet dieses an B. B antwortet mit einem k-Bit Vektor zufälliger Ausprägung. A berechnet daraufhin die Antwort $y = r \cdot \prod_{j=1}^{k} e_j S_j \ (\text{mod } n)$ (also das Produkt aus r und denjenigen S_i, die durch die Challenge spezifiziert wurden) und versendet diese an B. Die authentifizierende Stelle B überprüft, ob die Bedingung $(x = \pm y^2 \cdot \prod_{j=1}^{k} e_j J_j \ (\text{mod } n))$ erfüllt ist. Ist dies der Fall, ist eine Authentifizierungsrunde erfolgreich verlaufen. Der Nachrichtenaustausch kann wie folgt dargestellt werden:

$$A \rightarrow B \quad : \quad x = \pm r^2 \pmod{n} \tag{20}$$
$$A \leftarrow B \quad : \quad (e_1, ..., e_k), e_i \in \{0, 1\} \tag{21}$$
$$A \rightarrow B \quad : \quad y = r \cdot \prod_{j=1}^{k} e_j S_j \pmod{n} \tag{22}$$

Beim oben dargestellten Protokollablauf wird, wie beim Grundprinzip aus Kapitel 5.5 auch, lediglich die Wahrscheinlichkeit, dass eine Challenge gelöst wurde, obwohl ein Geheimnis nicht bekannt ist, verringert. Beim hier dargestellten Protokoll verringert sich die Irrtumswahrscheinlichkeit bei einem Durchlauf auf 1 zu 2^k. Das Protokoll wird insgesamt t-mal durchlaufen, sodass die Irrtumswahrscheinlichkeit nach t Runden auf 1 zu 2^{tk} sinkt. Die Autoren des Originalartikels (vgl. [51] S. 93) sehen einen Wert von $kt = 20$ als ausreichend an, wobei hier die Wahrscheinlichkeit einer missbräuchlichen Authentifizierung auf 1 zu 1000000 sinkt. Entscheidend für die Praktikabilität des Protokolls ist, wie sich kt zusammensetzt. Durch eine einzige parallele Durchführung des Protokolls ($k = 20, t = 1$) bei gleichzeitigem konstantem Produkt aus kt könnte zwar eine gleiche Irrtumswahrscheinlichkeit erreicht werden bei einer gleichzeitig verringerten notwendigen Interaktivität der Teilnehmer, eine Zero-Knowledge-Eigenschaft des Protokolls wäre dann jedoch nicht mehr gegeben (vgl. [51] S. 92).

5.5.2 Guillou-Quisquater-Authentifizierungsprotokoll

Während beim Zero-Knowledge-Authentifizierungsprotokoll nach Feige et al. mehrere Iterationen des Protokollablaufes benötigt werden, haben Guillou und Quisquater (vgl. [60] und [52]) das ursprüngliche Protokoll nach Fiat und Shamir dahin gehend erweitert, dass die Anzahl der notwendigen Protokolldurchläufe auf ein Minimum (einen einzigen) reduziert werden konnte. Entsprechend müssen nur drei Nachrichten zwischen anfragendem Client und authentifizierender Stelle ausgetauscht werden, ohne die Zero-Knowledge-Eigenschaft einbüßen zu müssen. Jedoch muss für ein gleiches Sicherheitslevel dafür der dreifache Aufwand an Rechenzeit gegenüber dem Protokoll von Feige, Fiat und Shamir aufgebracht werden (vgl. [9], S. 508). Die Sicherheit des Verfahrens beruht auf der Schwierigkeit, die v-te Wurzeln im Restklassenring $\mathbb{Z}_n$ bei unbekannter Faktorisierung von n zu berechnen.

Wie beim vorherigen Protokoll auch, soll nachfolgend auf die Systeminitialisierung, die Nutzerinitialisierung sowie den Protokollablauf eingegangen werden.

Die Systeminitialisierung wird durch eine vertrauenswürdige Instanz durchgeführt. Da in Tor keine vertrauenswürdige dritte Instanz vorhanden ist, muss dies durch den Dienstanbieter erfolgen. Dazu wird zunächst ein gemeinsamer Modul n veröffentlicht, der durch die Multiplikation zweier großer geheim gehaltener Primzahlen p und q ermittelt wird. Weiterhin wird ein systemweit bekannter Exponent e durch den Dienstanbieter ermittelt. Dabei sollte für den Exponent v die Eigenschaft $\gcd(v, \phi) = 1$ (wobei $\phi = (p-1)(q-1)$) gelten (vgl. [15] S. 413).

Bei der Nutzerinitialisierung hat der Client sowohl einen öffentlichen als auch einen privaten Schlüssel zu generieren. Der öffentliche Schlüssel (J) wird aus den Identitätsinformationen generiert, indem aus diesen beispielsweise ein Hashwert errechnet wird. Der

private Schlüssel (S) wird aus dem öffentlichen Schlüssel errechnet, sodass die Bedingung $JS^v \equiv 1(\bmod n)$ gilt. Dies kann über den erweiterten euklidischen Algorithmus sehr effizient geschehen.

Beim Protokollablauf (vgl. [9] S. 509) generiert der sich authentifizierende Client zunächst eine Zufallszahl r, die in $\mathbb{Z}_n^*$ liegen muss, und ermittelt daraus $T = r^v \pmod n$. Diese Zahl wird als erste Nachricht (witness) an B versendet. Die authentifizierende Stelle B generiert und versendet daraufhin eine Zufallszahl d, die sich im Bereich von 0 und $v-1$ befinden muss. A errechnet hieraus zusammen mit der versendeten witness T eine Antwort $D = rS^d \pmod n$ und übermittelt diese an B. Daraufhin überprüft A, ob die Bedingung $T' \equiv T(\bmod n)$ erfüllt ist, wobei $T' = D^v J^d \pmod n$. Ist dies der Fall, war eine Authentifizierung erfolgreich. Der Nachrichtenverlauf stellt sich wie folgt dar:

$$A \rightarrow B \quad : \quad J, T = r^v \pmod n \tag{23}$$

$$A \leftarrow B \quad : \quad d \text{ (wobei } 0 \leq e \leq v-1) \tag{24}$$

$$A \rightarrow B \quad : \quad D = r \cdot S^d \pmod n \tag{25}$$

Die Wahrscheinlichkeit, dass ein Angreifer eine Authentifizierung aushebelt, indem er die Challenge mit einer korrekten Antwort löst, wird beim Guillou-Quisquater-Protokoll durch den Exponenten v determiniert. Ein Angreifer hat eine Chance von 1 zu v, die Challenge d bereits im Vorhinein korrekt zu erahnen und damit ohne den privaten Schlüssel eine korrekte Antwort (D) zu erzeugen. Ein Wiederholen des Verfahrens ist nicht notwendig, wenn die Größe von v das benötigte Sicherheitslevel garantiert.

5.5.3 Schnorr-Authentifizierungsprotokoll

Im Gegensatz zu den beiden vorherigen Zero-Knowledge-Protokollen, basiert die Sicherheit des Zero-Knowledge-Authentifizierungsprotokolls nach Schnorr (vgl. [53]) auf dem diskreten Logarithmus-Problem. Hierbei besteht die Annahme, dass es sich als komplexitätstheoretisch schwer erweist, bei einer gegebenen Primzahl p, einer Basis α und einem Element β ($\alpha, \beta \in \mathbb{Z}_p^*$) eine Ganzzahl x ($0 \leq x \leq p-2$) zu erzeugen, für die gilt: $\alpha^x = \beta \pmod p$.

Wie bei den beiden anderen Protokollen auch wird die Festlegung der Systemparameter durch eine vertrauenswürdige dritte Partei vorgenommen. Dazu legt diese zunächst eine geeignete Primzahl p fest, sodass $p-1$ durch eine andere Primzahl q teilbar ist. Schnorr schlägt eine Mindestgröße von 512 bzw. 140 Bit für p und q vor. Neben diesen beiden Parametern ist ein weiterer Parameter $\alpha \in \mathbb{Z}_p$ zur Ordnung q ($\alpha^q = 1 \pmod p$) systemweit bekannt. Ein festgelegter Sicherheitsparameter t (wobei $2^t < q$) determiniert die Größe der durch B wählbaren Challenges.

Im Originalprotokoll nach Schnorr hat die vertrauenswürdige dritte Partei neben der Festlegung der Systemparameter auch die Aufgabe, bei der Nutzerinitialisierung sog. Zertifikate ($cert_x$) der öffentlichen Schlüssel der einzelnen Nutzer zu erstellen. Damit wird eine Verknüpfung zwischen einem öffentlichen Schlüssel und Identitätsmerkmalen hergestellt. Für die folgende Betrachtung sind diese Zertifikate zu vernachlässigen, da auch die öffentlichen Schlüssel als Pseudonyme verwendet werden können. Zur Schlüsselgenerierung wird der private Schlüssel (S) von jedem Nutzer durch eine Zufallszahl ($S \in \{1, .., p\}$) generiert und geheimgehalten. Der öffentliche Schlüssel J ist aus dem privaten sehr effi-

zient errechenbar: $J = \alpha^{-S} \pmod{p}$. Ein Berechnen des privaten Schlüssels mit Hilfe des öffentlichen erfordert hingegen das Lösen des diskreten Logarithmus-Problems.

Nachfolgend wird der Nachrichtenaustausch zwischen den beiden beteiligten Parteien verdeutlicht:

$$A \rightarrow B \quad : \quad x = \alpha^r \pmod{p} \tag{26}$$
$$A \leftarrow B \quad : \quad e \text{ (wobei } e \in \left\{0, ..., 2^t - 1\right\}) \tag{27}$$
$$A \rightarrow B \quad : \quad y = r + Se \pmod{q} \tag{28}$$

Das Protokoll läuft nach der Initialisierung des Systems und der Nutzer folgendermaßen ab: Die sich authentifizierende Partei A wählt eine Zufallszahl $r \in \{1, ..., q-1\}$, errechnet hieraus die witness $x = \alpha^r \pmod{p}$ und übersendet x der authentifizierenden Stelle B. Daraufhin antwortet B mit einer Challenge e ($e \in \{0, ..., 2^t - 1\}$). Aus dieser erhaltenen Challenge errechnet der sich authentifizierende Client zusammen mit der Zufallszahl r und dem privaten Schlüssel S die Antwort auf die Challenge: $y = r + Se \pmod{q}$. Dies erfordert statt einer aufwendigeren Exponentiation lediglich eine wenig rechenintensive Multiplikation im Restklassenring. In einem letzten Schritt prüft die authentifizierende Stelle B, ob die erhaltene witness x und x' ($x' = \alpha^y J^e \pmod{p}$ übereinstimmen. Ist dies der Fall, ist eine Authentifizierung erfolgreich.

Die Irrtumswahrscheinlichkeit, dass ein Client A einen korrekten Beweis geliefert hat, ohne den privaten Schlüssel S gekannt zu haben, wird durch den Sicherheitsparameter t bestimmt und kann mit 1 zu t festgelegt werden. Somit kann mit k Protokolldurchläufen die Wahrscheinlichkeit eines unberechtigten Zugriffes auf 1 zu $2^k t$ festgelegt werden. Wie beim Guillou-Quisquater-Protokoll auch kann die Anzahl der Protokolldurchläufe auf einen einzigen reduziert werden, wenn der Sicherheitsparameter entsprechend groß gewählt wird. Zu beachten ist jedoch, dass für große Challenges die Zero-Knowledge-Eigenschaft des Protokolls verloren geht. Da die Größe der Challenges durch dem Sicherheitsparameter t bestimmt wird, sind große Bereiche für die Auswahl einer Challenge jedoch unabdingbar, wenn die Anzahl der Protokolldurchläufe minimiert werden soll (vgl. [61] S. 634). Somit kann festgehalten werden, dass bei Einhaltung der Zero-Knowledge-Eigenschaft nicht gleichzeitig die Anzahl der Protokolldurchläufe minimiert werden kann. Das Protokoll erlaubt weiterhin ein Vorausberechnen der rechenintensiven Operationen, sodass ein sich authentifizierender Client lediglich eine einzige Multiplikation im Restklassenring zum Zeitpunkt der Authentifizierung vornehmen muss. Für die authentifizierende Stelle ergibt sich aber keine Möglichkeit der Vorausberechnung, sodass hier kein Geschwindigkeitsvorteil ggü. den anderen Protokollen ensteht.

5.5.4 Bewertung und Vergleich der Zero-Knowledge-Protokolle

Nachdem drei unterschiedliche Zero-Knowledge-Protokolle in den vorherigen Abschnitten erläutert wurden, soll in diesem Abschnitt zunächst ein Vergleich der Protokolle stattfinden. Danach wird darauf eingegangen, inwieweit die in Kapitel 5.1 aufgestellten Anforderungen an ein Authentifizierungsverfahren durch Zero-Knowledge-Verfahren erfüllt werden können und welches Protokoll am geeignetsten für eine Umsetzung erscheint. Gemeinsam ist allen drei Protokollen, dass eine Überführung des Authentifizierungsprotokolls in ein Signaturverfahren sehr einfach möglich ist, indem die authentifizierende Stelle durch eine sichere Hashfunktion ersetzt wird. Weiterhin ist allen Protokollen gemein, dass

pro Protokollrunde genau drei Nachrichten zwischen authentifizierender Stelle und sich authentifizierendem Client ausgetauscht werden.

Zunächst sollen die Verfahren bezüglich des Rechenaufwands, der durch eine Authentifizierung verursacht wird, verglichen werden. Zu betrachten ist bei einem Performance-Vergleich nicht so sehr die Erzeugung von Signaturen bzw. gleichbedeutend der Nachrichten durch den anfragenden Client, sondern im Kontext dieser Arbeit viel mehr die Rechenzeit, die für eine Verifikation der Signatur bzw. des Authentifizierungsvorgangs von einem Hidden Service bzw. einem Introduction Point benötigt wird. Dies ist umso wichtiger, da rechenaufwendige Operationen auf Seiten der authentifizierenden Stelle können einen Angriff auf die Verfügbarkeit eines Dienstes begünstigen. Schnorr (vgl. [53] S. 249) hat die Performance aller drei aufgeführten Zero-Knowledge-Protokolle bei einem Sicherheitsparameter von 2^{72} untersucht, d.h. der anfragende Client hat eine Chance von 1 zu 2^{72}, eine gültige Signatur zu erzeugen bzw. sich zu authentifizieren, ohne den privaten Schlüssel zu kennen. Dabei konnte festgestellt werden, dass bezüglich der Performance das Feige-Fiat-Shamir-Protokoll dem Protokoll von Guillou-Quisquater im Faktor drei und dem Protokoll nach Schnorr im Faktor fünf überlegen ist. Lediglich bei der Erzeugung von Signaturen besitzt das Schnorr-Protokoll klare Vorteile.

Ein zweiter Punkt bei der Beurteilung der Verfahrens sollte der benötigte zusätzliche Nachrichtenoverhead sein. Pro Protokollablauf werden, wie oben bereits erläutert, genau drei Nachrichten (*witness, challenge, response*) ausgetauscht. Unterschiedlich ist jedoch bei den Verfahren, wie viele Protokollrunden pro Authentifizierungsvorgang benötigt werden. Während alle drei Verfahren theoretisch soweit modifiziert werden können, dass eine Authentifizierung auch mit einem einzigen Protokolldurchlauf die Fehlerwahrscheinlichkeit auf ein definiertes Maß reduziert, ist es bei den beiden Protokollen nach Feige, Fiat und Shamir sowie nach Schnorr nicht möglich, die Zero-Knowledge-Eigenschaft der Verfahren dabei zu bewahren. Lediglich beim Protokoll von Guillou und Quisquater bleibt auch die Zero-Knowledge-Eigenschaft erhalten.

Auf Grund der Tatsache, dass das Guillou-Quisquater-Protokoll das Nachrichtenaufkommen als einziges Protokoll auf insgesamt drei Nachrichten reduzieren kann, was sich bei der Benutzung von interaktiven Diensten bezüglich der Latenzzeit als entscheidend erweisen kann, und ein Aufwand an Rechenoperationen sich im Rahmen hält (108 Operationen ggü. beispielsweise 750 bei asymmetrischen Verfahren bei gleichen Sicherheitsparametern (vgl. [53] S. 249)), ist diesem Protokoll gegenüber den beiden anderen Zero-Knowledge-Protokollen der Vorzug zu geben. Im Folgenden soll daher nur noch das Guillou-Quisquater-Protokoll betrachtet werden und anhand dieses Protokolls überprüft werden, inwieweit die Anforderungen, die an ein Authentifizierungsverfahren für Hidden Services in Tor gestellt werden, erfüllt werden.

Das Guillou-Quisquater-Protokoll erfüllt zunächst die zentrale Anforderung an Authentifizierungsverfahren für Hidden Services in Tor, dass eine Authentifizierung durch Dritte ermöglicht wird. Durch die Zero-Knowledge-Eigenschaft werden keinerlei Informationen über das Geheimnis des Clients bei der Authentifizierung dem nicht-vertrauenswürdigen Introduction Point preisgegeben. Es ist nicht notwendig, dass eine Rückfrage an den Hidden Service bei der Authentifizierung erfolgen muss, was Verkehrsanalysen ermöglichen würde.

Bezüglich der zweistufigen Authentifizierung ergeben sich auf Grund der geforderten Interaktivität von Zero-Knowledge-Protokollen Probleme mit der Umsetzung des Verfahrens. Da Schneier zufolge (vgl. [9] S. 512) jedoch alle Identifizierungsverfahren in Signaturverfahren umgewandelt werden können, indem die authentifizierende Stelle durch eine sichere Hashfunktion (H) ersetzt wird, kann die Interaktivität eingeschränkt werden. Die zufällige Generierung der Challenge wird dann durch H simuliert. Anhand des Guillou-Quisquater-Protokolls soll der Protokollablauf bei einem Signaturverfahren verdeutlicht werden.

Sowohl Nutzer- als auch Systeminitialisierung sind analog der Initialisierung beim Authentifizierungsverfahren vorzunehmen. Dies bedeutet, dass mit einer Initialisierung und Schlüsselgenerierung sowohl Authentifizierung als auch Signieren von Nachrichten vorgenommen werden können, was eine zusätzliche Schlüsselgenerierung überflüssig macht. Das Signaturprotokoll sieht folgendermaßen aus: Die signierende Stelle generiert eine Zufallszahl r aus $\mathbb{Z}_n^*$ und errechnet hieraus $T = r^v \pmod n$. Zusammen mit der Nachricht M bildet dies die Eingabe für die sichere Hashfunktion H. Mittels der Hashfunktion wird die Challenge $d = H(M, T)$ ermittelt. d muss sich in einem Bereich zwischen 0 und $v - 1$ befinden, was durch eine Modulo-Operation mit v erreicht werden kann. Die Signatur besteht neben dem errechneten d, dem öffentliche Schlüssel J und der Nachricht M auch aus D, das sich wie folgt errechnen lässt: $D = rS^d \pmod n$. Diese Signatur wird an die prüfende Stelle B übermittelt:

$$A \rightarrow B \quad : \quad M, d, D, J \text{ (wobei } d = H(M, T);\ D = rS^d \pmod n) \tag{29}$$

B überprüft die Korrektheit der erzeugten Signatur und errechnet hierzu zunächst $T' = D^v J^d \pmod n$. In einem folgenden Schritt ermittelt B mit Hilfe der verwendeten Hashfunktion $d' = H(M, T')$. Falls d' und d aus der Signatur identisch sind, muss A den privaten Schlüssel bei einer Fehlerwahrscheinlichkeit von 1 zu v kennen, und die Signatur ist korrekt. Ein solches Signaturverfahren erfordert, wie in den Anforderungen spezifiziert, lediglich eine Nachricht zwischen Client und Hidden Service. Zu beachten ist, dass die verwendeten Signaturen mit einer Komponente versehen werden sollten, die die Aktualität der Signatur beweist, um einen Wiederholungsangriff zu verhindern. Hier können auf Grund der mangelnden Interaktivität lediglich Sequenznummern oder Zeitstempel zum Einsatz kommen. Die Anforderung der zweistufigen Authentifizierung kann mit dem Guillou-Quisquater-Protokoll also vollständig erfüllt werden.

Die funktionalen Anforderungen sind folglich vollständig abgedeckt. In Bezug auf die Anforderungen der Wahrung der Anonymität kann ebenfalls davon ausgegangen werden, dass ein Mindestmaß an Anonymität („kontrollierte Pseudonymität“) für den Nutzer ggü. Dritten und keine Beeinträchtigungen für den Anbieter eines Hidden Service gewährleistet sind. Durch die Verwendung eines u.U. nicht-vertrauenswürdigen Introduction Point ist es möglich, dass auch Angreifer auf Grund des verwendeten öffentlichen Schlüssels verschiedene Transaktionen von einem Nutzer in Verbindung setzen können. Es ist jedoch nicht möglich, eine Verbindung zwischen dem öffentlichen Schlüssel und der eigentlichen Identität des Nutzers herzustellen: Es herrscht Pseudonymität. Für den Hidden Service besteht die Möglichkeit, wie dies in den Anforderungen definiert wurde, Transaktionen einem Pseudonym bzw. einer Identität zuordnen zu können. Besondere erweiterte Infrastrukturbedingungen stellt ein Zero-Knowledge-Verfahren nicht. Lediglich die Systeminitialisierung ist durch eine vertrauenswürdige Partei (beispielsweise den Dienstanbieter selbst) vorzunehmen. Dies stellt jedoch keine Problem dar, da im weiteren Protokollablauf keine vertrauenswürdige Stelle mehr benötigt wird. Der Nachrichtenoverhead des

Verfahrens ist mit drei zusätzlichen Nachrichten, die in den bestehenden Protokollablauf von Tor größtenteils eingebunden werden können, noch akzeptabel und auf Grund der Zero-Knowledge-Eigenschaft unvermeidbar. Für die Authentifizierung ggü. dem Hidden Service selbst könnte eine Signatur zum Einsatz kommen, was eine zusätzliche Schlüsselgenerierung überflüssig macht und keine zusätzlichen Interaktion zwischen Hidden Service und Dienstnutzer erfordert. Zusätzliches Datenvolumen für die Authentifizierung entsteht durch die Schlüsselverbreitung, durch die drei Nachrichten (witness, challenge, response) und die Signatur für die Authentifizierung zwischen Dienstnutzer und Hidden Service. Das zusätzliche Datenvolumen ist direkt von einer Schlüssellänge sl sowie dem gewählten Sicherheitsparameter v abhängig. Es ergibt sich ein zusätzliches Datenvolumen pro Authentifizierung von $sl \cdot 4 + v$ sowie einige zusätzliche Bits für Verwaltungsinformationen der Nachrichten. Dieses zusätzliche Datenvolumen scheint beispielsweise für eine heute geläufige Schlüssellänge von 1024 Bit und einen Sicherheitsparameter von 16 bit (Risiko von 1 zu 65537) akzeptabel. Zusätzlich muss bei einer Initialisierung des Introduction Point noch eine sog. Zugangskontrollliste zwischen Hidden Service und Introduction Point ausgetauscht werden, um festzulegen, welche Pseudonyme Zugang zu einem Dienst erhalten sollen. Dies kann über gehashte Schlüsselinformationen (beispielsweise mit 20 Bit pro Nutzer) geschehen und ist nur bei der Initialisierung des Introduction Point erforderlich. Da zwischen dem Dienstanbieter und dem Dienstnutzer ausschließlich öffentliche Schlüssel ausgetauscht werden, ist ein Setup des Authentifizierungsverfahrens sehr einfach auch anonym möglich. Das Verwalten von dynamischen Benutzergruppen sollte auch keine Probleme verursachen, wenn Zugangskontrolllisten verwendet werden, bei denen bei Bedarf Nutzer hinzugefügt oder gelöscht werden können.

5.6 Gruppenauthentifizierungsprotokolle

Während alle bisher verglichenen Authentifizierungsverfahren darauf ausgerichtet waren, dass *ein* Nutzer gegenüber einer authentifizierenden Stelle die *eigene* Identität beweist, sich also authentifiziert, haben die in diesem Kapitel näher untersuchten Gruppenauthentifizierungsprotokolle eine andere Intention: Hier ist es einem Nutzer möglich, die Zugehörigkeit zu einer bestimmten Gruppe in der Art zu beweisen, dass es der authentifizierenden Stelle (hier: dem Introduction Point) nicht möglich ist, die Identität des Nutzers auszumachen und auch keine Möglichkeit für die authentifizierende Stelle besteht, Verknüpfungen zwischen verschiedenen Anfragen des Nutzers herzustellen. Dem Nutzer ist es also möglich, die Gruppenzugehörigkeit anonym zu beweisen und daher anonym innerhalb der Gruppe zu agieren. Einige Verfahren besitzen einen Gruppenmanager (hier: der Hidden Service), der eine Signatur öffnen und die Identität des Signierenden feststellen kann.

Verschiedene Ansätze sind in der Literatur zu finden, durch welche Gruppenauthentifizierung stattfinden kann. Insbesondere sollen Gruppen- und Ring-Signatur-Verfahren ausführlich beleuchtet und daraufhin untersucht werden, inwieweit sie die aufgestellten Anforderungen an Authentifizierungsverfahren für Hidden Services in Tor erfüllen können. Abschließend sollen noch verschiedene andere Gruppenauthentifizierungsprotokolle vorgestellt werden. Auf diese Verfahren soll jedoch nicht im Detail eingegangen werden.

5.6.1 Gruppen-Signatur-Verfahren

In diesem Abschnitt soll auf Gruppen-Signatur-Verfahren, die von Chaum 1991 (vgl.[17]) als neue Anwendungsdomäne von herkömmlichen Signaturverfahren (vgl. 5.4.2) vorgestellt wurden, eingegangen werden. Gruppen-Signaturen haben, im Gegensatz zu „normalen“ Signaturen, die folgenden drei Eigenschaften (vgl.[35] S. 57):

1. Nur Gruppenmitglieder können gültige Signaturen erzeugen.
2. Der Empfänger einer solchen Signatur kann überprüfen, ob die Signatur korrekt ist, kann jedoch nicht feststellen, welches Gruppenmitglied die Signatur erzeugt hat.
3. Falls es notwendig erscheint, kann durch eine dritte Instanz eine Signatur „geöffnet“ werden, d.h. die Identität des Signierenden kann dadurch festgestellt werden.

Gruppen-Signatur-Verfahren sind in einer ganzen Reihe von Anwendungsgebieten vorzufinden. Camenisch und Groth geben für Gruppen-Signaturen ein Anwendungsszenario vor, das den Zielsetzungen und Rahmenbedingungen dieser wissenschaftlichen Arbeit sehr nahe kommt: *„The most prominent one* [application, der Autor] *is probably in trusted computing, where a computing device is required to authenticate as proper (i.e., secure) device, i.e., that it has obtained attestation by some third party. To protect privacy of the device's user, this authentication should not allow identification of the device.“* ([62] S. 120). Wie dem beschriebenen Anwendungsszenario zu entnehmen ist, sind auch hier drei Parteien beteiligt, die vergleichbare Rollen übernehmen, wie die Parteien in dem dieser Arbeit zugrunde liegenden Anwendungsszenario. Ein Dienstnutzer (computing device) soll einer authentifizierenden Stelle die Berechtigung zur Nutzung einer Ressource nachweisen, indem eine Attestierung des Dienstanbieters (third party) anonym vorgelegt wird. Es ist also naheliegend, dass die Ähnlichkeit der Anwendungsszenarien auch eine Verwendung von ähnlichen Authentifizierungsverfahren erlaubt.

Gruppen-Signatur-Protokoll nach Chaum

Um die drei genannten Eigenschaften von Gruppen-Signaturen zu verdeutlichen, soll nun zunächst ein sehr einfaches Protokoll vorgestellt werden, das dem Artikel von Chaum et al. entnommen ist (vgl. [35] S. 259), ehe auf komplexere Verfahren eingegangen wird. Die folgenden sechs Schritte beschreiben den Ablauf:

1. Eine vertrauenswürdige dritte Partei (der Gruppenmanager) übernimmt zunächst eine Verteilung von öffentlichen und privaten Schlüsseln an die einzelnen Teilnehmer. Jedes der n Gruppenmitglieder erhält eine Liste mit m privaten Schlüsseln, die zum Signieren einer Nachricht verwendet werden können. Die einzelnen Listen der Teilnehmer sind disjunkt.
2. Alle öffentlichen Schlüssel der Gruppenmitglieder werden in einer Gesamtliste veröffentlicht.
3. Wenn ein Mitglied der Gruppe eine Signatur erzeugen möchte, so verwendet es hierzu einen privaten Schlüssel aus der erhaltenen Liste. Jeder Schlüssel darf lediglich einmal verwendet werden, um ein Verknüpfen von Nachrichten zu verhindern.

4. Die Gültigkeit der erstellten Signatur kann mit Hilfe des zugehörigen öffentlichen Schlüssels der Gesamtliste von jedem überprüft werden.

5. Durch den Gruppenmanager kann anhand der einzelnen Teilnehmerlisten festgestellt werden, welches Mitglied die Signatur erzeugt hat.

Dieses sehr einfache Verfahren erfüllt zunächst die drei oben genannten generellen Eigenschaften von Gruppen-Signatur-Verfahren. Es sind jedoch einige Probleme mit diesem Protokoll verbunden, die durch komplexere kryptografische Verfahren behoben werden können. Insbesondere stellt es sich als problematisch dar, dass eine vertrauenswürdige Stelle benötigt wird, die eine Schlüsselverteilung vornimmt, folglich auch alle privaten Schlüssel kennt und aus diesem Grund gefälschte Signaturen erzeugen kann. Außerdem erscheint es unpraktikabel, dass die Anzahl der Signaturen, die eine Person erzeugen kann, beschränkt ist, da Schlüssel nur ein einziges Mal verwendet werden können, um ein Verknüpfen von Nachrichten zu verhindern.

Gruppen-Signatur-Protokoll nach Camenisch

Auf Grund dieser Einschränkungen soll nun ein Verfahren vorgestellt werden, das eine Schlüsselgenerierung durch die einzelnen Mitglieder nutzt und so ein missbräuchliches Signieren durch den Gruppenmanager verhindert. Dieses Protokoll geht auf Arbeiten von Camenisch zurück (vgl. [63]). Die Sicherheit basiert auf dem diskreten Logarithmus-Problem. Basis für die Verschlüsselung und Signierung ist eine Variation des ElGamal-Verschlüsselungsverfahrens (vgl. [64]) mit denselben Sicherheitseigenschaften. Für dieses Verschlüsselungsverfahren besitzt jeder Gruppenteilnehmer sowohl einen privaten Schlüssel x_i, der als Zufallszahl aus $\mathbb{Z}_q$ generiert wird (q ist eine Primzahl), als auch einen öffentlichen Schlüssel y_i, der in Abhängigkeit von dem privaten Schlüssel erzeugt wird: $y_i = g^{x_i} \pmod q$ (wobei g als Basis eine Zufallszahl darstellt, die jedoch kleiner als q sein muss). Des Weiteren wird eine sichere Hashfunktion (h) benötigt. Der Gruppenmanager besitzt ebenfalls sowohl einen privaten Schlüssel w als auch einen öffentlichen Schlüssel $v = g^w \pmod q$.

Neben diesen kryptografischen Primitiven werden für das Gruppen-Signatur-Verfahren zwei weitere Algorithmen benötigt (vgl. [63] S. 470 ff.). Der erste Baustein, der Verwendung findet, ist eine Signatur, mit der der Signierende beweisen kann, dass er Kenntnis von einem diskreten Logarithmus besitzt. Faktisch kann der Signierende damit einen Dritten überzeugen, dass er durch das Erstellen einer Signatur (c, s) einen privaten Schlüssel (x) bei gegebenem g, q und öffentlichem Schlüssel y besitzt. Diese Signatur (c, s) löst die Gleichung der folgenden Struktur:

$$c = h(g|y|g^s y^c|m) \tag{30}$$

Die passende Signatur kann nur erzeugt werden, wenn der private Schlüssel (x) bekannt ist. Um die Signatur zu erstellen, wählt der Signierende zunächst eine Zufallszahl r und generiert den ersten Teil der Signatur durch die Hashfunktion: $c = h(g|y|g^r|m)$. Über den privaten Schlüssel (x) lässt sich das zugehörige s effizient berechnen: $s = r - cx \pmod q$. Jeder Empfänger einer solchen Signatur kann damit überprüfen, ob der Ersteller den zu y passenden privaten Schlüssel kannte, indem er überprüft, ob die Signatur korrekt gebildet wurde. Das obige Signaturverfahren kann dahingehend erweitert werden, dass

ein Signierender zeigen kann, dass er für mindestens einen öffentlichen Schlüssel aus einer Menge von n Schlüsseln $(y_1, ..., y_n)$ den passenden privaten Schlüssel kennt, ohne jedoch anzugeben, für welchen Schlüssel er den diskreten Logarithmus x_i lösen kann. Die Signatur $(c_1, ..., c_n, s_1, ..., s_n)$ muss die Gleichung der folgenden Struktur lösen, um den Beweis zu erbringen:

$$\sum_{i=1}^{n} c_i = h(g|y_1|...|y_n|g^{s_1}y_1^{c_1}|...|g^{s_n}y_n^{c_n}|m) \pmod q \tag{31}$$

Die beiden dargestellten Beweise mittels Signaturen sollen im weiteren Verlauf dieser Betrachtung mit $SKDL(g, y, m)$ bzw. mit $SKDL[^n_1](g, y_1, ..., y_n, m)$ (Signature of Knowledge of a Discrete Logarithm) bezeichnet werden. Ein $SKDL[^n_1](g, y_1, ..., y_n, m)$ kann nur erzeugt werden, wenn der signierenden Partei mindestens einer der privaten Schlüssel bekannt ist. Um diese Signatur zu erstellen und damit einen Beweis zu erbringen, wählt der Signierende zunächst $r, c_2, ..., c_n, s_2, ..., s_n$ zufällig aus $\mathbb{Z}_q$. In einem folgenden Schritt wird der erste Wert der Signatur (c_1) durch die Hashfunktion ermittelt: $c_1 = h(g|y_1|...|y_n|g^r|g^{s_2}y_2^{c_2}|...|g^{s_n}y_n^{c_n}|m) - \sum_{i=2}^{n} c_i \pmod q$. Mit Hilfe des eigenen privaten Schlüssels x_1 kann nun der noch fehlende Wert der Signatur (s_1) errechnet werden: $s_1 = r - x_1c_1 \pmod q$.

Ein zweiter Baustein, der für das Gruppen-Signatur-Verfahren benötigt wird, ist ein Beweis, der die Gleichheit zweier diskreter Logarithmen zeigt. Durch eine Signatur (c, s), die die folgende Gleichung:

$$c = h(h|g|z|y|h^s z^c|g^s y^c|m) \tag{32}$$

löst, lässt sich zeigen, dass der diskrete Logarithmus von z zur Basis h und von y zur Basis g identisch sein müssen. Nur so ist es dem Signierenden möglich, über $s = r - cx$ ein eindeutiges s zu bestimmen, sodass die obige Gleichung gelöst wird. Für dieses Schema lässt sich ebenfalls ein Beweis für eine Gruppe dergestalt konstruieren, dass ein Signierender zeigen kann, dass für ein Paar z_i, y_i $(i \in \{1, ..., n\})$ gilt, dass der diskrete Logarithmus zur Basis h bzw. zu Basis g identisch ist. Dieses Signatur-Schema soll nachfolgend mit $SEQDL[^n_1](h, g, z_1, y_1, ..., z_n, y_n, m)$ (Signature of Equality of Discrete Logarithms) bezeichnet werden.

Nachdem die Basiskonzepte, die für das Gruppen-Signatur-Protokoll benötigt werden, erläutert wurden, soll auf den eigentlichen Protokollablauf (vgl. [63] S. 473 f.) eingegangen werden. Folgende Beweis-Idee lässt sich bei dem Protokoll ausmachen: Ein Gruppenmitglied j $(j \in \{1, ..., n\})$ verschlüsselt den eigenen öffentlichen Schlüssel y_j aus der Menge aller öffentlichen Schlüssel der Gruppe $(y_1, ..., y_n)$ mit dem öffentlichen Schlüssel des Gruppenmanagers. Anschließend beweist es durch Signaturen, dass es einen der öffentlichen Schlüssel (y_j) chiffriert hat und dass es den diskreten Logarithmus (privaten Schlüssel x_j) von y_j kennt. Dazu sind die folgenden Schritte notwendig:

1. Auswahl einer Zufallszahl α.
2. Verschlüsseln von y_j: $A = v^\alpha$ und $B = y_j g^\alpha$.
3. Erzeugen der Signatur $(c_1, ..., c_n, s_1, ..., s_n) = SEQDL[^n_1](z, g, A, \frac{B}{y_1}, ..., A, \frac{B}{y_n}, m)$.
4. Erzeugen einer zweiten Signatur $(\bar{c}, \bar{s}) = SKDL(g, B, m)$.
5. Kombinieren der beiden erzeugten Signaturen zu einer Gruppen-Signatur, die die folgenden Bestandteile besitzt: $(A, B, c_1, ..., c_n, s_1, ...s_n, \bar{c}, \bar{s})$.

Durch den dritten Schritt zeigt der Signierende, dass ein Schlüssel aus der Liste $(y_1, ..., y_n)$ mit dem öffentlichen Schlüssel des Gruppenmanagers chiffriert wurde. Durch die Signatur im vierten Schritt wird bewiesen, dass der Signierende darüber hinaus auch den zugehörigen privaten Schlüssel (x_j), also den diskreten Logarithmus zu diesem öffentlichen Schlüssel, kennt. Beide Signaturen können von einem Empfänger auf Korrektheit überprüft werden. Somit beweist der Signierende indirekt, dass er Mitglied der Gruppe ist.

Um eine Signatur zu öffnen, kann der Gruppenmanager mit seinem privaten Schlüssel direkt aus B den öffentlichen Schlüssel y_j extrahieren und die Identität des Signierenden j erfahren:

$$\frac{B}{A^{w^{-1}}} = \frac{y_j g^{\alpha}}{v^{\alpha w^{-1}}} = \frac{y_j g^{\alpha}}{g^{w \alpha w^{-1}}} = y_j \pmod{q} \tag{33}$$

Das hier vorgestellte Verfahren stellt einen Ausgangspunkt für zahlreiche weitere Gruppen-Signatur-Protokolle dar, deren detaillierte Betrachtung über den Kontext dieser Arbeit hinausgeht. Diese Erweiterungen betreffen insbesondere die folgenden Einschränkungen, die mit einer Umsetzung des vorgestellten Protokolls verbunden sind:

- Die Länge des Gruppenschlüssels und der Gruppen-Signaturen ist abhängig von der Größe der Gruppe. Dies ist insbesondere bei großen Gruppen sehr problematisch. Camenisch und Stadler schlugen daher 1997 ein abgewandeltes Gruppen-Signatur-Protokoll vor (vgl. [65]), das diese Einschränkung nicht mehr besitzt.
- Ein großes Problem von Gruppen-Signatur-Verfahren stellt das Löschen von Mitgliedern aus einer bestehenden Gruppe dar, wenn die Anonymität der Gruppenmitglieder erhalten bleiben soll. Unterschiedliche Ansätze werden hierzu vorgeschlagen. Während Ateniese et al. (vgl. [66]) das Ändern des öffentlichen Gruppenschlüssels sowie ein Re-Initialisieren von Mitgliederschlüsseln vorschlagen, verknüpfen Bresson et al. (vgl. [67]) die Informationen zu den Schlüsseln der gelöschten Mitglieder mit dem Gruppenschlüssel. Dies scheint insbesondere bei vielen gelöschten Mitgliedern ineffizient, da die Größe der Signaturen und des öffentlichen Schlüssels linear mit der Anzahl der gelöschten Teilnehmer ansteigt. Camenisch und Lysyanskaya schlagen daher ein Verfahren vor, das ein Löschen von Mitgliedern erlaubt, ohne signifikant den Aufwand für die einzelnen Operationen zu erhöhen (vgl. [68]).
- Eine letzte Einschränkung, die mit dem vorgestellten Verfahren verbunden ist, ist die Tatsache, dass die Aufnahme von Mitgliedern ein Ändern der Gruppenschlüssel erfordert. Mit dem Verfahren von Camenisch und Stadler (vgl. [65]) kann auch diese Restriktion beseitigt werden.

Bewertung von Gruppen-Signatur-Protokollen

In diesem Abschnitt soll eine Bewertung von Gruppen-Signatur-Protokollen unter der Annahme erfolgen, dass die zuvor dargestellten Erweiterungen des Grundkonzeptes mit in die Bewertung einfließen. Zunächst soll der funktionale Aspekt betrachtet werden: Ein Gruppen-Signatur-Protokoll bietet die Möglichkeit, dass eine dritte Stelle (der Introduction Point) diese Signatur überprüfen kann, ohne dass hierfür vertrauenswürdige Daten dem Introduction Point zugänglich gemacht werden müssen. Allein über den öffentlichen

Gruppenschlüssel kann eine Gruppen-Signatur überprüft werden. Die einzige Einschränkung, die mit dem Verfahren verbunden ist, ist die Tatsache, dass die Gültigkeit der Signaturen zeitlich begrenzt werden muss, um ein missbräuchliches Verwenden von Signaturen zu verhindern. Dies kann über Zeitstempelverfahren, Zufallszahlen oder Sequenznummern geschehen. Da Sequenznummern auf Grund der mangelnden Synchronisierung von Introduction Points nicht infrage kommen und Zeitstempel ein Synchronisieren von Systemzeiten erfordern, sollten hier Zufallszahlen trotz der zwei zusätzlichen Nachrichten gewählt werden. Eine zweistufige Authentifizierung ist auf Grund der Einschränkung auf nur eine ausgetauschte Nachricht sehr einfach realisierbar. Die Aktualität der Signaturen sollte aber bei einer Authentifizierung zwischen Client und Hidden Service nicht über Zufallszahlen geschehen, da hier zusätzliche Interaktionen benötigt werden. Hier sollten vielmehr Sequenznummern oder Zeitstempel zum Einsatz kommen. Bezüglich der Anonymität bietet das Verfahren kaum Einschränkungen im Hinblick auf Nutzer bzw. Anbieter. Ein Introduction Point bzw. jede dritte Partei kann nur bestimmen, ob die vorliegende Anfrage eine gültige Signatur enthält, also der Zugriff auf einen Dienst gestattet ist oder nicht. Ein Verknüpfen von unterschiedlichen Anfragen ist nicht möglich. Ein Hidden Service hingegen kann die Signaturen „öffnen“ und einzelne Transaktionen den Pseudonymen bzw. Identitäten auf Grund der Gruppenmanager-Rolle zuordnen. Ein Setup des Verfahrens scheint einfach realisierbar, da ein asymmetrisches Verschlüsselungsverfahren die Grundlage der Gruppen-Signatur bildet und somit ein anonymer Austausch von öffentlichen Schlüsseln möglich ist. Weiterhin sind dynamische Benutzergruppen auch effizient bei Verwendung des Protokolls von Camenisch und Lysyanskaya (vgl. [68]) realisierbar. Bezogen auf die Infrastrukturanforderungen ergeben sich keine erweiterten Anforderungen an die Hidden Service-Architektur des Tor-Netzwerkes. Zusätzliche Nachrichten sind durch eine Implementierung eines Gruppen-Signatur-Verfahrens nicht zu erwarten. Eine Authentifizierung kann durch das Übermitteln einer einzelnen Signatur vorgenommen werden. Positiv wirken sich in dieser Hinsicht auch Erweiterungen des Basisprotokolls aus, die dazu führen, dass die Länge der Gruppen-Signaturen und des Gruppenschlüssels konstant bleibt, auch wenn die Größe der Nutzergruppe anwächst. Insgesamt spricht lediglich die sehr komplexe Konstruktion eines solchen Protokolls gegen eine erste prototypische Umsetzung eines Authentifizierungsmechanismus im Rahmen dieser Arbeit.

5.6.2 Ring-Signatur-Verfahren

In diesem Abschnitt soll das von Rivest et al. (vgl. [10]) vorgeschlagene Ring-Signatur-Verfahren näher beleuchtet werden. Im Gegensatz zu den im vorherigen Kapitel betrachteten Gruppen-Signaturen ist bei den Ring-Signatur-Verfahren kein Gruppenmanager vorhanden, der eine Gruppe initialisiert und ggf. die Signaturen bestimmten Identitäten zuordnen kann, womit die Anonymität der signierenden Partei ggü. dem Gruppenmanager aufgehoben wäre. Bei den Ring-Signaturen sind vielmehr gleichberechtigte Nutzer vorhanden, die Signaturen erzeugen können, welche nicht auf den ursprünglichen Signierenden zurückgeführt werden können, sondern bei denen nur bestimmt werden kann, dass eine Signatur von einem Mitglied einer ausgewählten Gruppe erzeugt wurde. Im Unterschied zu den Gruppen-Signaturen sind bei den Ring-Signaturen zunächst keine vordefinierten Nutzergruppen vorhanden, womit auch ein Setup des Verfahrens entfällt. Es ist lediglich erforderlich, dass eine signierende Partei die öffentlichen Schlüssel der anderen beteiligten Ringmitglieder kennt. Da ein Austausch der öffentlichen Schlüssel beispielsweise über eine Schlüsselinfrastruktur stattfinden kann, ist ein Zutun der anderen Ringmitglieder nicht

erforderlich. Einzige Voraussetzung für das Verfahren ist, dass Schlüssel für ein vereinbartes asymmetrisches Verschlüsselungsverfahren (beispielsweise Rivest-Shamir-Adleman (RSA)) erzeugt wurden.

Zwei wesentliche Protokollaktionen sind bei dem Ring-Signatur-Verfahren zu unterscheiden (vgl. [10] S. 554):

- *ring-sign*$(m, P_1, ..., P_r, s, S_s)$ – Diese Protokollaktion erzeugt eine Signatur σ von einer Nachricht m, bei Verwendung der öffentlichen Schlüssel P_i der r Ringmitglieder und dem privaten Schlüssel S_s des Signierenden s.
- *ring-verify*(m, σ) – Diese zweite Protokollaktion mit der Nachricht m und der zugehörige Signatur σ als Input zeigt an, ob eine Signatur korrekt erzeugt wurde.

Im Folgenden soll nun auf die Funktionsweise des Signaturprotokolls im Detail eingegangen werden. Dabei sollen zunächst kurz die Anwendung findenden Basisalgorithmen (vgl. [10] S. 556 f.) aufgezeigt werden. Vorausgesetzt wird, dass jedes Ringmitglied einen öffentlichen (J_i) und privaten Schlüssel (S_i) besitzt, der auf einer Trap-Door-Einwegfunktion, beispielsweise RSA, beruht. Da alle Mitglieder bei diesen Einwegfunktionen unterschiedliche Klar- und Chiffretextmengen $\mathbb{Z}_n$ besitzen, was eine Kombination von individuellen Signaturen erschwert, sind erweiterte Trap-Door-Einwegfunktionen g_i zu definieren, die auf den RSA-Einwegfunktionen beruhen. Diese erweiterten Trap-Door-Einwegfunktionen basieren auf der Sicherheit der RSA-Funktion, besitzen jedoch gleiche Eingabe- und Chiffretextmengen und sollen statt dieser im Folgenden für die asymmetrische Verschlüsselung verwendet werden. Das Funktionsprinzip wird ausführlich im Originaltext erläutert (vgl. [10] S. 557). Weiterhin wird ein symmetrischer Verschlüsselungsalgorithmus E_k mit Schlüssel k der Länge l sowie eine sichere Hashfunktion h bei dem Ring-Signatur-Verfahren verwendet. Außerdem soll eine sog. „Combining Function“ $C_{k,v}(y_1, ..., y_r)$ mit einer Zufallszahl v sowie einem Schlüssel k als Parametern, beliebigen Eingabewerten $y_1, ..., y_r$ und einer Ausgabe z derart definiert sein, dass sie den folgenden Anforderungen genügt:

- Die Combining Function stellt eine Permutation für jeden Eingabewert dar. Bei sonst konstanten Eingabewerten besteht zwischen jedem y_i ($i \in \{1, ..., r\}$) und dem Ausgabewert z eine eineindeutige Abbildung.
- Die Combining Function ist effizient berechenbar. Bei sonst konstanten Eingabewerten ist sowohl z aus y_i effizient berechenbar als auch umgekehrt.
- Es ist unmöglich, die Gleichung $C_{k,v}(g_1(x_1), ..., g_r(x_r)) = z$ bei gegebenen k, v und z für $x_1, ..., x_r$ zu lösen, wenn nicht für mindestens eine der Trap-Door Funktionen die Umkehrung berechnet werden kann.

Eine Combining Function, die die oben genannten Eigenschaften erfüllt, wurde von Rivest et al. mit Hilfe der symmetrischen Verschlüsselungsfunktion E_k und XOR (Exclusive OR) Operationen realisiert (vgl. [10] S. 559):

$$C_{k,v}(y_1, ..., y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus ... \oplus E_k(y_1 \oplus v))) \tag{34}$$

Durch die Bedingung, dass der Eingabeparameter v gleich dem Ausgabewert z sein soll, kann diese Combining Function zur Generierung einer Signatur verwendet werden. Voraussetzung für die Nutzung ist, dass der Hashwert $h(m)$ der Originalnachricht m als Schlüsselwert k verwendet wird. Lediglich ein Mitglied der Gruppe, das einen passenden privaten Schlüssel S_s besitzt, kann bei den gegebenen Anforderungen die Bedingung $z = v$ bei sonst fixen Parametern durch Umkehrung der eigenen Trap-Door-Einwegfunktion g_s erzeugen. Abbildung 12 verdeutlicht die entstehende Ringstruktur, die dem Signaturverfahren den Namen gibt.

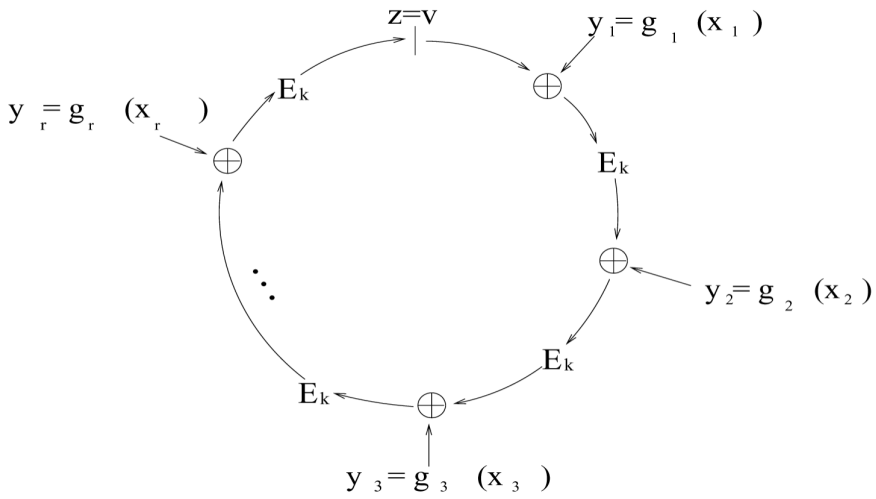

Abbildung 12: Erzeugen von Ring-Signaturen (aus: [10] S. 560)

Im Folgenden soll nun dargestellt werden, wie mit Hilfe der aufgeführten Funktionen Ring-Signaturen erzeugt werden und in einem zweiten Schritt durch eine beliebige dritte Partei verifiziert werden können.

Die Signaturerzeugung vollzieht sich in einer sechsstufigen Abfolge (vgl. [10] S. 559 f.):

1. Der Signierende (s) erzeugt zunächst unter Anwendung einer Hashfunktion ($h(x)$) auf die Nachricht m den symmetrischen Schlüssel $k = h(m)$.

2. Als zweiten Eingabeparameter v der Combining Function wird eine Zufallszahl generiert.

3. Weiterhin generiert die signierende Stelle für jedes Mitglied im Ring eine Zufallszahl x_i $(i = 1, ..., r)$ und berechnet über die Trap-Door-Einwegfunktion g_i und die jeweiligen öffentlichen Schlüssel (J_i) der anderen Mitglieder P_i die zugehörigen y_i.

4. s berechnet den eigenen Wert y_s, indem die Gleichung $C_{v,k}(y_1, ..., y_r) = v$ gelöst wird. Auf Grund der Permutationseigenschaft der Combining Function erhält s einen eindeutigen Wert y_s.

5. Durch das Invertieren der Trap-Door-Funktion $g_s(y_s)$ unter Zuhilfenahme des eigenen privaten Schlüssels (S_s) erhält er $x_s = g_s^{-1}(y_s)$.

6. Die Signatur passend zur Nachricht m sieht nun folgendermaßen aus:

$$(P_1, ..., P_r, v, x_1, ..., x_r)$$

Eine prüfende Stelle kann in einem dreistufigen Verfahren die Gültigkeit einer Ring-Signatur $(P_1, ..., P_r, v, x_1, ..., x_r)$ einer Nachricht m wie folgt überprüfen (vgl. [10] S. 560 f.):

1. Zunächst ermittelt die prüfende Stelle den symmetrischen Schlüssel k, indem sie die Hashfunktion, ebenso wie der Signierende, auf die Nachricht m anwendet.
2. In einem zweiten Schritt werden für alle übermittelten x_i $(i = 1, ..., r)$ die zugehörigen y_i mit Hilfe der modifizierten Trap-Door Einwegfunktion g_i und den zugehörigen übermittelten öffentlichen Schlüsseln P_i errechnet.
3. Nun wird überprüft, ob die so ermittelten y_i die Gleichung $C_{v,k}(y_1, ..., y_r) = v$ erfüllen. Ist dies der Fall, ist eine Signatur gültig.

Nachdem auf die Funktionsweise des Ring-Signatur-Protokolls nun detailliert eingegangen wurde, soll zum Abschluss der Betrachtung zunächst auf eine Verwendung als alternative Authentifizierungsmethode eingegangen werden, ehe eine Bewertung des Verfahrens anhand des aufgestellten Anforderungskatalogs vorgenommen wird. Eine Grundvoraussetzung für die korrekte Funktionsweise des Verfahrens ist, dass die vom Signierenden angegebenen Ringmitglieder alle ein Zugriffsrecht besitzen, da es einer authentifizierenden Stelle auf Grund der Anonymität des Signaturverfahrens nicht möglich ist zu unterscheiden, ob eine Signatur von einem zugriffsberechtigten Nutzer erzeugt wurde oder nicht.

Bei einem Authentifizierungsverfahren für Hidden Services in Tor ergäbe sich das Problem, dass ein Nutzer Zugriff auf die öffentlichen Schlüssel aller anderen authentifizierten Nutzer haben müsste. Nur so ist es diesem Nutzer möglich, aus dem Pool der berechtigen Nutzer Signaturen auszuwählen und dadurch eine eindeutige Authentifizierung zu ermöglichen. Eine Verteilung der Schlüssel müsste des Weiteren dynamisch bei jedem Zugriff neu erfolgen, damit bei jeder erzeugten Signatur zum einen alle neu aufgenommenen Nutzer mit eingebunden werden, zum anderen ein Entfernen von öffentlichen Schlüsseln möglich ist, was einem Ausschluss eines Nutzers vom Dienst gleichkommt. Nur so ist die Anforderung nach dynamischen Gruppen realisierbar.

Bezüglich der beiden funktionalen Anforderungen an Authentifizierungsverfahren ergeben sich im Hinblick auf das Ring-Signatur-Protokoll keine Einschränkungen. Sowohl eine zweistufige Authentifizierung als auch eine Authentifizierung durch Dritte sind mit Hilfe der Signaturen möglich. Lediglich die zeitliche Gültigkeit der Signaturen müsste eingeschränkt werden, was über Zufallszahlen bzw. Zeitstempel möglich wäre. Mit einem Ring-Signatur-Verfahren, wie es zuvor beschrieben wurde, ist es einem Nutzer möglich, eine Nachricht zu verschlüsseln und seine eigene Identität innerhalb eines Pools von Nutzern zu verbergen. Somit scheint es möglich, wenn dieser Pool aus Nutzern ausschließlich aus zugriffsberechtigten Mitgliedern besteht, nachzuweisen, dass der Signierende ein Zugriffsrecht besitzt, ohne dass die eigene Identität preisgegeben wird. Da diese Anonymität auch nicht durch einen Manager, beispielsweise den Hidden Service, aufgehoben werden kann, kann die Anonymitätsbedingung, dass ein Hidden Service Transaktionen den einzelnen Pseudonymen bzw. Identitäten zuordnen kann, nicht erfüllt werden.

Mit diesem Verfahren sind weitere Einschränkungen bezüglich der Praktikabilität verbunden: Durch das Verfahren entsteht ein großer Nachrichtenoverhead, da zunächst bei

jeder Authentifizierung eine Aktualisierung der berechtigten öffentlichen Schlüssel stattfinden muss. Dies ist erforderlich, damit ein Nutzer lediglich Schlüssel von berechtigten Nutzern für eine Signatur verwendet. Weiterhin besteht jede Ring-Signatur auch aus mitgesendeten öffentlichen Schlüsseln deren Anzahl entscheidend die Anonymität des Nutzers beeinflusst, was die Erzeugung von unangemessenem Nachrichtenoverhead zur Folge hat. Wenngleich durch die Verwendung von Hashfunktionen der Overhead reduziert werden könnte, so bleibt doch die Praktikabilität des Verfahrens bei großen Nutzergruppen eingeschränkt. Weiterhin ist anzumerken, dass ein Ausschluss von Nutzern durch den Hidden Service zwar prinzipiell möglich ist, dynamische Gruppen also realisiert werden können, Fehlverhalten jedoch auf Grund der Anonymität des Nutzers auch gegenüber dem Hidden Service nicht erkannt werden kann.

Auf Grund der Anonymität des Nutzers gegenüber dem Hidden Service und einem hohen Nachrichtenoverhead des Verfahrens soll deshalb von einer Umsetzung im Rahmen dieser Arbeit Abstand genommen werden.

5.6.3 Sonstige Gruppenauthentifizierungsverfahren

De Santis et al. (vgl. [69]) haben das in Kapitel 5.5.1 beschriebene Zero-Knowledge-Protokoll von Feige, Fiat und Shamir dahingehend erweitert, dass mit der gleichen Systeminitialisierung eine anonyme Gruppenauthentifizierung möglich ist. Ziel des Protokolles ist es, den nötigen Kommunikationsaufwand des Protokollablaufes zu reduzieren. Wie bei den Original Zero-Knowledge-Protokollen werden beim Protokoll von de Santis et al. drei Nachrichten pro Protokolldurchlauf ausgetauscht. Da jedoch mit jedem Durchlauf die Irrtumswahrscheinlichkeit (1 zu 2^n, wobei n die Anzahl der Durchläufe darstellt) für eine authentifizierende Stelle lediglich halbiert wird, sind mehrere Durchläufe des Protokolls notwendig, um eine gewisse Sicherheit vor einem unberechtigtem Zugriff zu erhalten. Auf Grund der unverhältnismäßig hohen Interaktion zwischen anfragendem Client und einem prüfendem Introduction Point soll das Protokoll nicht näher untersucht werden. Zero-Knowledge-Beweise werden in den meisten Gruppen-Signatur-Verfahren eingesetzt, eine weitere Anwendung außerhalb dieser Signatur Verfahren und des Ansatzes von de Santis konnte in der Literatur nicht ausgemacht werden.

Ein anderer Ansatz für den anonymen Nachweis der Mitgliedschaft in einer Gruppe wird von Schechter et al. (vgl. [70]) vorgeschlagen. Die Autoren benutzen die Methode des „Verifiably Common Secret Encoding“, um eine Gruppenauthentifizierung zu ermöglichen. Dabei wird eine Challenge von der authentifizierenden Stelle mit den öffentlichen Schlüsseln aller Gruppenmitglieder chiffriert und dem sich authentifizierenden Client übersendet. Dieser kann mit dem eigenen privaten Schlüssel die für ihn verschlüsselte Challenge entschlüsseln und der authentifizierenden Stelle als Response zustellen. Die authentifizierende Stelle erkennt, dass der Client ein Mitglied der Gruppe sein muss, da er eine der Challenges lösen konnte. Um zu verhindern, dass die authentifizierende Stelle unterschiedliche Challenges für alle Nutzer verwendet und so die Identität offenlegt, kann der Nutzer überprüfen, ob der Vektor der verschlüsselten Challenges richtig erzeugt wurde. Dazu verschlüsselt er die gelöste Challenge, bevor er diese an die authentifizierende Stelle verschickt, mit allen öffentlichen Schlüsseln der anderen Nutzer und vergleicht den entstehenden Vektor mit dem erhaltenen. Sind beide identisch, wurden für alle Nutzer die gleichen Challenges verwendet. Nachteilig beim vorgeschlagenen Protokoll nach Schechter et al. ist insbesondere, dass bei jeder Authentifizierung ein Vektor, bestehend aus den für

alle Nutzer oder zumindest für eine große Anzahl an Nutzern verschlüsselten Challenges, mitgeschickt werden muss. Dies stellt einen großen Nachrichtenoverhead dar. Weiterhin müssen sowohl die authentifizierende Stelle als auch der anfragende Client genau diese Anzahl an asymmetrischen Verschlüsselungsoperationen durchführen, was insbesondere im Hinblick auf große Nutzerzahlen nicht effizient und akzeptabel erscheint. Eine Umsetzung dieses Verfahrens kommt daher aus Effizienzgesichtspunkten ebenfalls nicht infrage.

Das Gruppenauthentifizierungsprotokoll Homage geht auf Arbeiten von Handley (vgl. [71]) zurück und basiert auf dem diskreten Logarithmus-Problem. Dieses Verfahren ist insbesondere für große Gruppen geeignet, da sowohl Nachrichtenvolumen als auch Rechenzeit nicht durch die Anzahl der Mitglieder beeinflusst werden. Das Protokoll besitzt neben der Zusicherung, dass eine authentifizierende Stelle nicht die Anonymität innerhalb einer Gruppe auflösen kann, auch die Eigenschaft, dass selbst die Stelle, die eine Mitgliedschaft in einer Gruppe festlegt, die Identität der Nutzer nicht feststellen kann. Dies führt unweigerlich dazu, dass zwar eine Aufnahme von Nutzern möglich ist, das Protokoll jedoch keine Möglichkeit vorsieht, Nutzer aus der Gruppe zu entfernen. Dynamische Gruppen lassen sich daher nicht realisieren (vgl. [71] S. 297). Weiterhin ist es ausschließlich der Stelle, die eine Mitgliedschaft in einer Gruppe beschließt, möglich, eine Authentifizierung der Mitglieder durchzuführen. Dies verhindert, dass eine dritte Partei, beispielsweise der Introduction Point, eine Authentifizierungsaufgabe übernimmt. Die zentrale Anforderung an das Verfahren kann also nicht erfüllt werden. Fleming und Gohil (vgl. [72] und [73]) konnten außerdem zeigen, dass das Protokoll Sicherheitslücken aufweist. Wenngleich durch Modifikationen an dem Protokoll diese Sicherheitslücken geschlossen werden konnten, steht ein formaler Beweis der Sicherheit des Verfahrens noch aus (vgl. [73] o.S.). Auf Grund dieser aufgeführten Restriktionen soll im Weiteren nicht detaillierter auf den Protokollablauf eingegangen werden, da auch dieses Protokoll für eine Umsetzung nicht infrage kommt.

5.7 Fazit

Nachdem die unterschiedlichsten Authentifizierungsmechanismen detailliert erläutert und anschließend auch bezüglich ihrer Eignung für eine Authentifizierung anhand des aufgestellten Kriterienkatalogs bewertet wurden, soll nun ein Verfahren für die Implementierung ausgewählt werden. Tabelle 2 fasst zunächst die Ergebnisse dieses Kapitels zusammen.

Wie zu sehen ist, wurden weitere Betrachtungen nicht vorgenommen, wenn die zentralen funktionalen Anforderungen (Authentifizierung durch Dritte und zweistufige Authentifizierung) nicht erfüllt wurden. Bei den Einmalpasswort-Verfahren ist eine Authentifizierung durch Dritte mit Einschränkungen möglich, daher wurden hier auch die anderen Anforderungen untersucht.

Für eine anschließende Implementierung nicht infrage kommen zunächst alle Mechanismen, die eine Authentifizierungsmöglichkeit durch Dritte und eine zweistufige Authentifizierung nicht vollständig unterstützen, namentlich alle Passwortverfahren und Verfahren, die symmetrische Verschlüsselung verwenden. Zero-Knowledge-Protokolle haben gegenüber Challenge-Response-Verfahren, die auf asymmetrischer Kryptografie basieren, den Vorteil, dass durch den Protokollablauf keine Degeneration der Schlüssel stattfindet und dass eine effizientere Implementierung möglich ist. Während das vorgestellte Ring-Signatur-Verfahren auf Grund der nicht geeigneten Anonymitätseigenschaften und dem

	Authentifizierungsverfahren						
Anforderungen	Passwort-Verfahren	Einmalpasswort-Verfahren	Symmetrische C-R-Authentifizierung	Asymmtrische C-R-Authentifizierung	Zero-Knowledge-Protokoll (GQ)	Gruppen-Signatur-Verfahren	Ring-Signatur-Verfahren
Authentifizierung durch Dritte	–	(–)	–	+	+	+	+
Zweistufige Authentifizierung	•	+	•	+	+	+	+
Einfluss auf die Anonymität	•	+	•	+	+	+	–
Overhead des Verfahrens	•	+	•	+	+	+	–
Infrastrukturbedingungen	•	+	•	+	+	+	+
Setup des Verfahrens	•	–	•	+	+	+	+

+: genügt Anforderung –: genügt nicht Anforderung •: nicht detailliert betrachtet

Tabelle 2: Vergleich der Authentifizierungsmechanismen

Overhead des Verfahrens nicht infrage kommt, stellt sich ein Gruppen-Signatur-Protokoll, wegen der positiven Eigenschaften im Hinblick auf die gewährte Anonymität als Authentifizierungsverfahren erster Wahl dar. Im Folgenden soll trotzdem ein Zero-Knowledge-Protokoll als Authentifizierungsverfahren in Tor implementiert werden, da auch Zero-Knowledge-Protokolle alle aufgestellten Anforderungen vollständig erfüllen und eine prototypischen Implementierung des Verfahrens im Rahmen dieser Arbeit realisiert werden kann. Konkret soll in Anbetracht der geringeren Anforderungen, bezogen auf die notwendigen Interaktionen, das Guillou-Quisquater-Protokoll umgesetzt werden.

6 Design und Implementierung eines Authentifizierungsverfahrens in Tor

Nachdem eine Evaluierung der unterschiedlichen Authentifizierungsmechanismen im vorherigen Kapitel erfolgt ist, soll in diesem Kapitel die Umsetzung des gewählten Guillou-Quisquater-Protokolls für die Hidden Services in Tor beschrieben werden.

Bei dieser Beschreibung sollen zunächst die Rahmenbedingungen und Anforderungen, die im Zusammenhang mit einer Implementierung des Authentifizierungsverfahrens zu berücksichtigen sind, betrachtet werden. Darauf folgend sollen die beiden im Rahmen dieser Arbeit identifizierten Anwendungsszenarios getrennt von einander analysiert werden. Zunächst wird auf den Anwendungsfall der Einrichtung eines Hidden Service in Tor eingegangen, indem die Modellierung mittels eines Aktivitätsdiagramms und die Implementierung des Anwendungsfalls beschrieben werden. Ein gleiches Vorgehen soll auch beim zweiten Anwendungsfall, der Nutzung eines Hidden Service, bei dem eine Authentifizierung des Nutzers erforderlich ist, eingehalten werden. Weiterhin soll kurz erläutert werden, wie ein lokales Tor-Netzwerk initialisiert wird. Zum Abschluss des Kapitels wird aufgezeigt, welche verschiedenen Tests zum Gewährleisten einer korrekten Arbeitsweise des Authentifizierungsmechanismus durchgeführt wurden.

6.1 Rahmenbedingungen der Implementierung

In diesem Unterkapitel sollen kurz die Rahmenbedingungen erfasst werden, die im Zusammenhang mit der Implementierung zu beachten sind. Die Implementierung des Authentifizierungsverfahrens stellt eine Erweiterung des seit 2002 entwickelten Tor-Netzwerkes dar. Da sich Tor noch in der Entwicklungsphase befindet, sollte laut Hinweisen der Entwickler noch nicht auf die bereitgestellte Anonymität vertraut werden. Da Tor als freie Software entwickelt und verteilt wird, sind sowohl der Quellcode als auch plattformspezifische Installationsversionen auf der Projekt-Homepage[5] zum freien Download verfügbar. Für die Implementierung im Rahmen dieser wissenschaftlichen Arbeit wurde die zum Zeitpunkt der Arbeit aktuelle Entwicklerversion 0.1.1.17-rc eingesetzt. Auch der Quellcode zu dieser Version ist auf der Projekt-Seite[6] verfügbar. Das Tor-Projekt hat mit allen Erweiterungen zum Stand der eingesetzten Version einen Umfang von ca. 53.000 Codezeilen der Programmiersprache C vorzuweisen, die in insgesamt 71 Quelldateien organisiert sind.

Für eine Compilierung von Quellcode werden zusätzlich zu den von Tor bereitgestellten Quelldateien noch externe Bibliotheken benötigt. Konkret werden Funktionen der ebenfalls frei verfügbaren Softwareprojekte Libevent[7] (Callback-Funktionalität bei Eintritt von Ereignissen), Zlib[8] (für eine Komprimierung von Datenpaketen) und die kryptografischen Bibliotheken von OpenSSL[9] durch Tor verwendet.

Abschließend soll zusätzlich die Implementierung hinsichtlich der bisher umgesetzten Authentifizierungsverfahren beleuchtet werden. Authentifizierungsmechanismen wurden in

[5] http://tor.eff.org/download.html.de [Stand: 2006.09.27]
[6] https://tor-svn.freehaven.net/svn/tor/tags/tor-0_1_1_17_rc/ [Stand: 2006.09.27]
[7] http://www.monkey.org/~provos/libevent/ [Stand: 2006.09.27]
[8] http://www.zlib.net/ [Stand: 2006.09.27]
[9] http://www.openssl.org/ [Stand: 2006.09.27]

den Designdokumenten zu Tor (vgl. [11] S. 8) bereits im Jahr 2004 klar als Ziel einer Implementierung herausgestellt. Weder eine nähere Spezifikation noch eine Implementierung eines Mechanismus hat jedoch bisher stattgefunden. Vorgeschlagen werden zwar sog. „authorization token“, mit denen sowohl auf Ebene der Introduction Points als auch auf Ebene des Dienstanbieters eine Zugangskontrolle stattfinden soll. In den entsprechenden Spezifikationen findet sich hierzu allerdings keine detaillierte Beschreibung (vgl. [46] o.S.).

6.2 Anforderungen an die Implementierung

Nachdem in Kapitel 5.1 auf die Anforderungen an einen Authentifizierungsmechanismus im Allgemeinen eingegangen wurde, sollen in diesem Abschnitt Anforderungen an eine konkrete Implementierung des Guillou-Quisquater-Protokolls in Tor aufgestellt werden.

Da die aktuelle Tor-Implementierung auf sehr unterschiedlichen Betriebssystemen lauffähig ist und somit einer breiten Basis an Nutzern zur Verfügung steht, soll eine Erweiterung der Implementierung diese Betriebssystem-Unabhängigkeit nicht einschränken, um keine Akzeptanzprobleme zu verursachen. Hierbei ist insbesondere zu beachten, dass keine betriebssystemspezifischen Bibliotheken verwendet werden und dass auf Grund von unterschiedlichen Byte-Reihenfolgen bei der Speicherbelegung Daten in Form einer plattformübergreifenden Darstellung (Network Byte Order) ausgetauscht werden müssen.

Zunächst sollen, um eine Abwärtskompatibilität mit alten Versionen des Tor-Netzwerkes zu ermöglichen, die Änderungen an den Nachrichtenstrukturen und an der Architektur möglichst gering ausfallen. Vor allem sollen die Änderungen so integriert werden, dass eine Nutzung von Hidden Services ohne jegliche Authentifizierung auch weiterhin möglich bleibt und diese auch mit älteren Tor-Versionen erfolgen kann. Bei den Änderungen an den Nachrichtenstrukturen und der Architektur ist weiterhin zu beachten, dass die zentralen Tor-Directories nicht durch zusätzliche Operationen bzw. zusätzlichen Nachrichtenoverhead belastet werden. Dies bedeutet auch, dass die veröffentlichten Rendezvous Service Descriptors (RSD), die Details zu den Hidden Services beschreiben und durch die Directories verwaltet werden, möglichst wenig zusätzliche Informationen bezüglich des Authentifizierungsverfahrens speichern.

Weitere Anforderung an die Implementierung soll sein, dass eine dynamische Nutzerverwaltung auf Seiten des Hidden Services einfach möglich ist und auch die Verwaltung der öffentlichen Schlüssel mit einbezieht. Generell soll es nicht erforderlich sein, separat Schlüssel zu erzeugen. Optimalerweise sollte eine Nutzerverwaltung über die Konfigurationsdatei `torrc` erfolgen, sodass an einer zentralen Stelle Änderungen vorgenommen werden können. Außerdem soll es für Dienstbetreiber kein Problem darstellen, mehrere Hidden Services mit jeweils unterschiedlichen Nutzergruppen auf einem Server zu betreiben.

Als letzte Anforderung an eine Implementierung soll formuliert werden, dass eine Implementierung möglichst generisch erfolgt. Es sollte demnach zu einem späteren Zeitpunkt möglich sein, eine Implementierung anderer Authentifizierungsverfahren vorzunehmen, ohne dass die Nachrichtenstrukturen erweitert oder verändert werden müssen. Daher sollen die Authentifizierungsinformationen, die über Nachrichten ausgetauscht werden, durch eine einheitliche Struktur festgelegt werden. Diese generische Nachrichtenstruktur verdeutlicht Abbildung 13: In einem ersten Datenfeld von 16 Bit (entspricht zwei Oktetten) soll zunächst das verwendete Authentifizierungsverfahren angegeben werden, ehe die eigentli-

chen Authentifizierungsdaten (variable Länge) inklusive deren Längenangabe (16 Bit) folgen. Einem Empfänger bleibt es dann überlassen, die Authentifizierungsdaten abhängig vom vorgegebenen Authentifizierungsverfahren zu interpretieren. Diese Vorgehensweise lässt genügend Freiraum für die Implementierung eines beliebigen Authentifizierungsverfahrens.

Bezeichnung	Auth. Typ	Datenlänge	Auth. Daten
Länge	2 Oktette	2 Oktette	variabel

Abbildung 13: Generische Authentifizierungsstruktur aller Nachrichten

6.3 Einrichtung eines Hidden Service mit Authentifizierung

In diesem Abschnitt soll auf den ersten Anwendungsfall, der Einrichtung eines Hidden Service mit Authentifizierung durch einen Dienstanbieter näher eingegangen werden. Die Einrichtung des Hidden Service umfasst alle Aktionen, die von den beteiligten drei Parteien (Introduction Point, Hidden Service, Directory) veranlasst werden, damit ein Hidden Service im Tor-Netzwerk verfügbar ist. Alle Schritte, die notwendig sind, damit ein Client danach einen Hidden Service nutzen kann (Schlüsselaustausch, Authentifizierung), werden im zweiten Anwendungsfall beschrieben. In beiden Anwendungsfällen sollen insbesondere die Schritte, die sich von einer Initialisierung ohne Verwendung von Authentifizierungsverfahren unterscheiden, näher beleuchtet werden. Dazu soll zunächst eine Modellierung des Anwendungsfalls mittels eines Aktivitätsdiagramms erfolgen, ehe auf die konkreten Schritte bei der Implementierung eingegangen wird.

6.3.1 Modellierung des Anwendungsfalls

Alle hier beschriebenen Abfolgen sind in Anlage A als entsprechendes Aktivitätsdiagramm in UML (Unified Modeling Language) dargestellt. In diesem Abschnitt soll eine detaillierte Beschreibung des Ablaufes erfolgen. Zur besseren Anschaulichkeit sind Datenobjekte und Aktionen, die für eine Authentifizierung mittels Guillou-Quisquater-Protokoll modifiziert oder hinzugefügt werden mussten, rot eingefärbt. Auf diese soll näher eingegangen werden. Die Struktur der modifizierten Nachrichtenobjekte ist separat in Anlage B dargestellt. Auch hier sind die entsprechenden modifizierten Felder rot gekennzeichnet. Anzumerken ist, dass die in den Tabellen im Anhang angegebenen Nachrichtenfelder auf Grund einer hexadezimalen Kodierung der Authentifizierungsdaten doppelt so groß sind, wie die im Text erwähnten Schlüssel- und Datenlängen. Die hexadezimale Darstellung der Daten hat den Vorteil, dass diese sowohl auf Big-Endian-Architekturen als auch auf Little-Endian-Architekturen gleich ist und sich somit keine Konvertierungsprobleme ergeben.

Die Einrichtung eines Hidden Service wird zunächst vom Dienstbetreiber angestoßen. Wird ein Hidden Service zum allerersten Mal vom Dienstbetreiber gestartet, sind drei Parameter vom System zu generieren: Zunächst wird ein Schlüsselpaar, das im Folgenden für alle Operationen im Zusammenhang mit dem Hidden Service benutzt wird, erzeugt. Weiterhin wird aus dem so generierten öffentlichen Schlüssel mittels einer Hashfunktion die sog. Onion-Adresse als Hostname berechnet, die den Service eindeutig kennzeichnet. Zum Schluss ist es initial erforderlich, dass ein gemeinsamer Modul (n) durch den Hidden

Service erzeugt wird. Dieser gemeinsame Modul wird durch Multiplikation zweier großer Primzahlen ermittelt und hat eine Schlüssellänge von 1024 Bit.

Danach lädt der Hidden Service alle bisher mit Clients ausgetauschten Zugangsschlüssel. Dieser Schlüsselaustausch ist, wie oben bereits erläutert, Bestandteil des Anwendungsszenarios der Nutzung eines Hidden Service. In Abhängigkeit von der Aktualität der Network Status Documents (NSD) und der Router Descriptions (RD) fordert der Hidden Service von den ihm bekannten Directory Servern aktuelle Router- und Netzwerkinformationen an. Da sich dieser Schritt nicht von einer Einrichtung ohne Authentifizierung unterscheidet, soll auf nähere Details nicht eingegangen werden. Eine nähere Spezifikation der Routerverwaltung kann im Tor Directory Protocol gefunden werden (vgl. [74]). Anhand der Einträge des Network Status Documents wählt der Hidden Service drei Router, die im Folgenden als Introduction Points dienen sollen, aus der Liste aus. Im Anschluss daran wird eine sog. Access Table (AT) für den Hidden Service vom Dienstanbieter erstellt. Diese Liste besteht aus den Hashwerten der öffentlichen Schlüssel aller zugriffsberechtigten Clients und dient somit als Autorisierungsinformation für die Introduction Points. Ein Hashen der öffentlichen Schlüssel mit einer allgemein bekannten Hashfunktion, beispielsweise SHA-1 (Secure Hash Algorithm), erfolgt, um den Bandbreitenbedarf für den späteren Austausch der Liste mit den Introduction Points zu reduzieren. Weiterhin erzeugt der Client nun einen Rendezvous Service Descriptor (RSD), der später für den anfragenden Client die notwendigen Informationen zum Auffinden des Hidden Service bereitstellt. Da dieser RSD über die zentralen Directory Server veröffentlicht wird, werden lediglich marginale Erweiterungen an dem RSD vorgenommen, um die zentralen Server nicht unnötig zu belasten. In Tabelle 4 des Anhangs B ist der Aufbau eines RSD in der Protokollversion V1 beschrieben. Eine vorhergehende Protokollversion V0 soll, trotz einer geforderten Abwärtskompatibilität, nicht unterstützt werden, da diese keine Veröffentlichung von detaillierten Informationen zu Introduction Points zulässt. Außerdem wird die aktuelle Protokollversion V1 bereits seit der Torversion 0.1.1.5 von August 2005 unterstützt. Der RSD wird für eine Authentifizierung über das Guillou-Quisquater (GQ) Protokoll lediglich dahingehend verändert, dass ein Authentifizierungstyp (Wert: 1) an der Stelle AUTHT vorbelegt wird. Durch diese Information ist es einem anfragenden Client möglich zu erkennen, welcher Authentifizierungsmechanismus durch den Hidden Service gefordert wird. Die Struktur der Datenfelder wurde so generisch gewählt, dass beliebige Authentifizierungsinformationen durch den RSD veröffentlicht werden können, wie diese als Anforderung an die Implementierung in Kapitel 6.2 aufgestellt wurde.

Der Hidden Service baut zu jedem angegebenen Introduction Point eine Verbindung über das Tor-Netzwerk auf und versendet an die einzelnen Introduction Point eine sog. Relay_Establish_Intro_Cell (REIC). Der Introduction Point überprüft die Korrektheit der beigefügten Signatur des Hidden Service und ob kein Wiederholungsangriff vorliegt. Fallen beide Prüfungen positiv aus, akzeptiert der Introduction Point die REIC. Tabelle 5 verdeutlicht den Aufbau dieses Datenobjektes. Über die REIC erhält der Client zunächst die Information, welcher Authentifizierungstyp für den Hidden Service vom Dienstanbieter gefordert wird. Ist der Typ = 1, wird GQ-Authentifizierung gefordert und der Introduction Point wertet die Authentifizierungsdaten entsprechend aus. Diese Daten bestehen zunächst aus dem gemeinsamen Modul (n), der für einen GQ-Beweis benötigt wird, sowie der zuvor erzeugten AT. Anhand der AT kann der Introduction Point später entscheiden, welche Clients zugriffsberechtigt sind. Die beiden Informationen werden durch den Introduction Point für eine spätere Zugriffskontrolle abgespeichert.

Über eine Relay_Intro_Established_Cell (RIEC) antwortet der Introduction Point auf die REIC des Hidden Service. Bei Vorliegen aller Bestätigungen der Introduction Points kann der Hidden Service den zuvor erzeugten Rendezvous Service Descriptor veröffentlichen. Dazu schickt er den Descriptor über einen HTTP-Request an die folgende Adresse auf einem der zentralen Directory Server: `http://<diraddress>/tor/rendezvous/publish`. Dieser nimmt weitere Kontrollen bezüglich der Signatur und des Zeitstempels vor und veröffentlicht den RSD für Clients. Ebenfalls über das HTTP-Protokoll unter der Adresse `http://<diraddress>/tor/rendezvous1/<onion-address>` ist der RSD auf den zentralen Directory Servern verfügbar. Der Hidden Service ist nun initialisiert und steht Clients über die Onion-Adresse zur Verfügung.

6.3.2 Implementierung des Anwendungsfalls

Nachdem für den ersten Anwendungsfall eine Modellierung des Ablaufes mit Hilfe eines Aktivitätsdiagrammes stattgefunden hat, sollen in diesem Abschnitt zentrale Stellen der konkreten Implementierung vorgestellt werden. Auf Grund des Umfangs kann an dieser Stelle lediglich auf Ausschnitte eingegangen werden. Entsprechende Ausschnitte des Quellcodes sind dem Anhang C zu entnehmen. Für weite Teile der Implementierung werden Pakete der frei verfügbaren kryptografischen Bibliothek OpenSSL[10] benötigt. Ausführlich beschrieben werden die unterschiedlichen Funktionen von OpenSSL bei Viega et al. (vgl. [75] und [76]). Insbesondere die Unterstützung von mathematischen Operationen auf großen Ganzzahlen, die durch herkömmliche Standard Bibliotheken von C nicht unterstützt werden, ist bei dieser Implementierung wichtig. Die BN-Bibliothek (Big Number) unterstützt durch den Datentyp BIGNUM Ganzzahlen ohne festen oberen Grenzwert und definiert mathematische Operationen auf diesem Datentyp.

In Listing 1 wird nun aufgezeigt, wie ein Rendezvous Service Descriptor erweitert werden muss, um eine Authentifizierung zu ermöglichen. Wie bereits im vorherigen Abschnitt erläutert, soll eine Erweiterung der RSD möglichst generisch erfolgen. Dazu wird in den ersten 16 Bit der Erweiterung zunächst der Authentifizierungstyp festgelegt (Zeile 5). Falls das Guillou-Quisquater-Protokoll zur Authentifizierung gewählt wurde, wird die Datenlänge mit 0 belegt (Zeile 8) und die eigentlichen Authentifizierungsdaten werden nicht verwendet. Dies entspricht dem Aufbau des RSD, der in Tabelle 4 in Anhang A dargestellt wird. Eine Decodierung auf Seiten des Dienstnutzers wird in den Zeilen 19 bis 29 aufgezeigt und erfolgt analog der Codierung des RSD. Zur Zeit ist lediglich das Guillou-Quisquater Protokoll zur Authentifizierung umgesetzt (Zeile 23), weitere Mechanismen sind an dieser Stelle denkbar.

6.4 Nutzung eines Hidden Service, der Authentifizierung erfordert

Nachdem darauf eingegangen wurde, welche Schritte insbesondere auf Seiten des Dienstanbieters erforderlich sind, um einen Hidden Service für die Nutzergruppe verfügbar zu machen, soll in diesem Abschnitt beschrieben werden, welche Maßnahmen durch den Nutzer ausgeführt werden müssen, um die Dienste des Hidden Service nutzen zu können. Da-

[10] http://www.openssl.org/ [Stand: 2006.09.27]

zu soll analog zur vorherigen Vorgehensweise zunächst der Anwendungsfall mittels eines UML-Aktivitätsdiagramms dargestellt und erläutert werden, ehe auf eine konkrete Implementierung eingegangen wird.

6.4.1 Modellierung des Anwendungsfalls

Bei der Nutzung eines Hidden Service müssen zunächst zwischen dem Hidden Service und den einzelnen Clients öffentliche Schlüssel ausgetauscht werden. Der Schlüsselaustausch muss jedoch nur dann ausgeführt werden, wenn ein Client erstmalig den Service nutzen möchte. Ein Schlüsselpaar (öffentlicher Schlüssel J und privater Schlüssel S) wird dazu am Anfang durch den Client erzeugt, wozu dem Client ein vom Hidden Service erzeugter gemeinsamer Modul bekannt sein muss. Dieser Modul (n) ist für alle Clients einheitlich und wird außerhalb des eigentliches Protokolls vereinbart. Ist dem Client der Modul bekannt, kann er das eigentliche Schlüsselpaar erzeugen, für das die folgende Relation gelten muss: $JS^v \equiv 1 \pmod{n}$. v stellt einen systemweit fixen Exponenten dar. Bei der Implementierung wurde der Wert $v = 65537$ gewählt, der Geschwindigkeitsvorteile bei Multiplikationsoperationen besitzt (vgl. [76] S. 329). Beide Schlüssel haben eine Schlüssellänge von 1024 Bit. Der private Schlüssel kann in einem ersten Schritt vom Client frei gewählt werden. In einem nächsten Schritt wird die Exponentiation mit v vorgenommen. Den zugehörigen öffentlichen Schlüssel kann man nun in einem letzten Schritt durch die Anwendung des erweiterten Euklidischen Algorithmus erhalten. Diesen öffentlichen Schlüssel tauscht der Client mit dem Hidden Service ebenfalls außerhalb des Protokollablaufes von Tor aus. Der Dienstanbieter veröffentlicht diese neuen Schlüsselinformationen, indem er den Hidden Service über einen Neustart einrichtet, so wie dies im Kapitel 6.3 gezeigt wurde. Diese beschriebene Prozedur ist nur erforderlich, wenn ein neuer Nutzer der zugriffsberechtigten Gruppe des Hidden Service hinzugefügt wird.

Um auf einen anonymen Dienst zugreifen zu können, ist es erforderlich, dass der Client die Onion-Adresse des Dienstes kennt. Da diese Adresse nichts über die Identität des Dienstanbieters aussagt und keine sicherheitsrelevanten Informationen enthält, kann diese Information auch außerhalb des Tor-Netzwerkes, beispielsweise über eine Veröffentlichung auf der Webseite eines Dritten bzw. über anonyme Chatrooms, publiziert werden. Tabelle 3 gibt ein Übersicht über die außerhalb des Tor-Protokolls ausgetauschten Daten.

C ← HS	Gemeinsamer Modul
C → HS	Öffentlicher Schlüssel des Clients
C ← HS	Onion-Adresse

Tabelle 3: Außerhalb des Tor-Protokolls ausgetauschte Daten

Die Onion-Adresse ist erforderlich, um einen Rendezvous Service Descriptor (RSD), der weitere Informationen zu einem Hidden Service enthält, vom Directory Server beziehen zu können. Um den RSD zu erhalten, startet der Client einen HTTP-Request an die folgende Adresse: `http://<diraddress>/tor/rendezvous1/<onion-address>` auf einem ausgewählten Directory-Server, der wiederum als Response den entsprechenden RSD liefert bzw. eine Fehlernachricht sendet, wenn der RSD auf dem Server nicht verfügbar ist. Die Struktur des erhaltenen RSD ist Tabelle 4 der Anlage B zu entnehmen. Da der RSD auch Informationen zu den Introduction Points, über die ein Hidden Service erreichbar

ist, enthält, kann der Client anhand dieser Informationen zunächst einen Introduction Point auswählen, über den der Verbindungsaufbau zum Hidden Service laufen soll. Weiterhin erkennt der Client, welcher Authentifizierungsmechanismus für einen Zugriff auf den Hidden Service erforderlich ist. Bei Verwendung des GQ-Protokolls ist der Authentifizierungstyp mit 1 belegt; ist keine Authentifizierung erforderlich, mit 0. Ist die GQ-Authentifizierung angegeben, nimmt der Client anonym über das Tor-Netzwerk Kontakt mit dem ausgewählten Introduction Point auf. Zwei zusätzliche Nachrichten gegenüber einem Protokollablauf ohne Authentifizierung werden nun zwischen den beiden Parteien ausgetauscht: Eine Relay_Access_Cell (RAC) und eine Relay_Challenge_Cell (RCC). Der Client übermittelt zunächst eine RAC, deren Aufbau in Tabelle 6 verdeutlicht wird. Die HS_ID identifiziert einen Hidden Service eindeutig und ermöglicht dem Introduction Point beispielsweise eine direkte Zuordnung der empfangenen Access Table des Hidden Service. Der Public Userkey stellt den öffentlichen Schlüssel (J) des Clients dar und wird zur Identifizierung des Clients benutzt. Die Testnummer T wird mittels einer zuvor generierten Zufallszahl r vom Client wie folgt ermittelt: $T = r^v \pmod{n}$ und hat eine Länge von 1024 Bit. Da die Zufallszahl bei einem späteren Beweis benötigt wird, speichert der Client sowohl Zufallszahl als auch Testnummer ab. Die RAC beinhaltet als Testnummer also die *witness* des interaktiven Beweises.

Bei Empfang einer RAC überprüft der Introduction Point zunächst, ob für die HS_ID eine entsprechende Access Table des Hidden Services hinterlegt wurde. Daraufhin erzeugt der Client mit Hilfe einer allgemein bekannten Hashfunktion (beispielsweise SHA-1) aus dem mitgelieferten Public Userkey eine User_ID, die er mit der Access Table abgleicht. Ist die User_ID vom Hidden Service in der Access Table verzeichnet worden, speichert der Introduction Point die beigefügte Testnummer und den öffentlichen Schlüssel unter der User_ID ab. Anschließend erzeugt er eine Challenge als Zufallszahl für den Client aus dem Bereich von 0 bis $v - 1$. Somit hat die Challenge eine Länge von 16 Bit. Diese Challenge wird mittels der RCC an den Client zurückgesandt. Ist ein User_ID nicht in der Access Table vorhanden oder keine Access Table zu dem Hidden Service registriert, wird eine RCC zurückgesandt, die keinen Nachrichteninhalt besitzt. Der Client erkennt somit, ob eine Authentifizierung fehlgeschlagen ist.

Der Client prüft den Nachrichteninhalt der Relay_Challenge_Cell und entfernt bei einem Authentifizierungsfehler den Introduction Point aus der Liste der Introduction Points, die über den RSD veröffentlicht wurde. Daraufhin initialisiert er einen Authentifizierungsvorgang mit dem nächsten verfügbaren Introduction Point. Ist der Nachrichteninhalt nicht leer, errechnet der Client in einem nächsten Schritt den entsprechenden Beweis, um den Authentifizierungsvorgang abzuschließen. Dieser Beweis D wird mit Hilfe der Zufallszahl r, dem privaten Schlüssel S und der erhaltenen Challenge d wie folgt berechnet: $D = r \times S^d \pmod{n}$. Die Sicherheit des Verfahrens wurde bereits in Kapitel 5.5.2 dargestellt. Der Client baut eine Verbindung zu einem gewählten Router auf und initialisiert diesen als Rendezvous Point. Da keinerlei Modifikationen bei diesem Protokollschritt vorgenommen wurden, soll der genaue Ablauf nicht Gegenstand dieser Betrachtung sein.

Um eine Authentifizierung auch gegenüber dem Hidden Service direkt zu ermöglichen, ist es erforderlich, dass die Interaktivität des Guillou-Quisquater-Protokolls minimiert wird. Nur so ist es möglich, dass die Gefahr, durch zusätzlichen Nachrichtenverkehr zwischen Client und Hidden Service Verkehrsanalysen zu ermöglichen, minimiert wird. Da jedes Zero-Knowledge-Protokoll auch in ein Signaturverfahren überführt werden kann, indem die Funktion des Prüfenden durch eine sichere Hashfunktion ersetzt wird, kann mittels

einer Signatur eine Authentifizierung gegenüber dem Hidden Service erfolgen. Dies hat zudem den Vorteil, dass die erzeugten Schlüssel auch für diese Authentifizierung verwendet werden können. Das Signaturverfahren des Guillou-Quisquater-Protokolls wurde bereits in Kapitel 5.5.2 beschrieben. Die Nachricht M, die signiert werden soll, besteht aus dem Tupel User_ID des Clients und einem generierten Zeitstempel. Die User_ID ermöglicht dem Hidden Service, eine Zuordnung zu dem zu verwendenden öffentlichen Schlüssel des Clients herzustellen. Der Zeitstempel soll verhindern, dass durch ein Wiedereinspielen von alten Signaturen ein Angriff auf den Hidden Service ermöglicht wird. Die Signatur besteht aus den drei Elementen M, d, D, wobei $d = H(M, T);\ D = rS^d \pmod n$ und hat eine Länge von 1136 Bit. Als sichere Hashfunktion (H) wird SHA-1 verwendet.

Der Client schickt nach den Berechnungen und der Initialisierung des Rendezvous Points eine Relay_Introduce1_Cell an den Introduction Point. Diese Nachricht besteht aus einem unverschlüsselten Teil, der für den Introduction Point bestimmt ist und einem für den Hidden Service mit dessen öffentlichen Schlüssel chiffrierten Teil (vgl. Tabelle 8). Der Introduction Point kontrolliert zunächst, ob eine Authentifizierung für den Hidden Service erforderlich ist. Diese Authentifizierung ist dann erforderlich, wenn eine Access Table vom Hidden Service übermittelt wurde. Eine erneute Überprüfung, ob der User sich in der AT befindet, kann entfallen, da in einem zweiten Schritt nun die Korrektheit des interaktiven Zero-Knowledge-Beweises geprüft wird. Dazu wird der in dem unverschlüsselten Teil der Nachricht übermittelte Beweis auf Korrektheit untersucht, indem der Introduction Point die Bedingung $T' \equiv T \pmod n$ kontrolliert, wobei $T' = D^v J^d \pmod n$. Ist dies der Fall, war eine Authentifizierung des Clients gegenüber dem Introduction Point erfolgreich. Wenn eine Registrierung des Hidden Service beim Introduction Point vorliegt, leitet der Introduction Point den verbleibenden verschlüsselten Teil der Nachricht als Relay_Introduce2_Cell auf dem initialisierten Pfad weiter an den Hidden Service. Außerdem erhält der anfragende Client eine Bestätigung auf die Relay_Introduce1_Cell. Der Introduction Point sendet dazu eine Relay_Command_Introduce_Ack_Cell an den Client. Der Nachrichteninhalt gibt dem Client an, ob eine Anfrage erfolgreich an den Hidden Service weitergeleitet wurde (kein Inhalt), ob eine Authentifizierung verweigert wurde (Nachrichteninhalt = 2) oder ob ein sonstiger Fehler im Protokollablauf aufgetreten ist (Nachrichteninhalt = 1). War eine Authentifizierung erfolgreich, wartet der Client auf eine Antwort des Hidden Service. Andernfalls versucht der Client eine Authentifizierung über einen der anderen verfügbaren Introduction Points.

Der Hidden Service erhält vom Introduction Point über die bereits initialisierten Pfade eine Relay_Introduce2_Cell, den verschlüsselten Teil der Relay_Introduce1_Cell. Mit Hilfe des eigenen privaten Schlüssels kann der Hidden Service die Daten dechiffrieren. Zur Authentifizierung ist dieser Nachricht die Guillou-Quisquater-Signatur beigefügt. Da die Nachricht aus einem Tupel (Zeitstempel|User_ID) besteht, kann der Hidden Service zunächst die Aktualität der Signatur mit Hilfe des Zeitstempels prüfen. Ist der Zeitstempel inkorrekt, verwirft der Hidden Service die Nachricht und reagiert nicht auf sie. Da die verschlüsselte Nachricht neben der Signatur auch Rendezvous-Informationen enthält, d.h. Daten des initialisierten Rendezvous Points, wird eine Signatur über die Verschlüsselung mit diesen Rendezvous Informationen verknüpft. Ein Angreifer besitzt nicht die Möglichkeit, die Signatur getrennt mit einem frei gewählten Rendezvous Point zu nutzen. Da der gewählte Rendezvous Point dem Angreifer auf Grund der Verschlüsselung nicht bekannt ist, wird dadurch das Risiko eines Wiederholungsangriffes weiter eingeschränkt. Nachdem die Korrektheit des Zeitstempels überprüft wurde, wird in einem zweiten Schritt die Signatur kontrolliert. Dazu sucht der Hidden Service zunächst über die User_ID der

Nachricht den passenden öffentlichen Schlüssel des Nutzers. Analog der Vorgehensweise aus Kapitel 5.5.2 überprüft der Hidden Service die Signatur und berechnet hierzu zunächst $T' = D^v J^d \pmod n$. Mit Hilfe der sicheren Hashfunktion SHA-1 erzeugt der Hidden Service $d' = H(M, T')$. Stimmen d aus der Signatur und das errechnete d' überein, ist die Authentifizierung erfolgreich. In diesem Fall baut der Hidden Service über den der Relay_Introduce2_Cell angegebenen Rendezvous Point eine Verbindung zu dem Client auf. Scheitert eine Authentifizierung, verwirft der Hidden Service die Nachricht und nimmt keinen Kontakt mit dem Client auf.

6.4.2 Implementierung des Anwendungsfalls

Nachdem die Schritte, die für die Nutzung eines eingerichteten Hidden Services notwendig sind, anhand eines Aktivitätsdiagrammes modelliert und beschrieben wurden, sollen in diesem Abschnitt ausgewählte Teile der Implementierung in Tor vorgestellt werden. Zunächst wird in Listing 2 dargestellt, wie die für das Guillou-Quisquater-Protokoll benötigten privaten und öffentlichen Schlüssel des Nutzers erzeugt werden, die mit dem gemeinsamen Modul in einem Struct crypto_accessk_env_t zusammengefasst werden. Dazu wird durch die im BN-Packet bereitgestellte Funktion zum Erzeugen von Primzahlen (BN_generate_prime) für den privaten Schlüssel zunächst eine Primzahl der Länge LK_BYTES*8 (1024 Bit) erzeugt. In Zeile 14 wird nun die Exponentiation des privaten Schlüssels (S) mit dem systemweit festgelegten Exponenten (GQ_EXP = 65537) vorgenommen und in tmp gespeichert. Die Exponentiation findet im Restklassenring statt. Um einen öffentlichen Schlüssel (J) zu generieren, wird in Zeile 17 der errechnete Wert tmp invertiert (BN_mod_inverse), sodass für die erzeugten Schlüssel die Gleichung $JS^{exp} \equiv 1 \pmod{mod}$ gilt. Ein Erzeugen des gemeinsamen Modulus durch den Dienstanbieter erfolgt entsprechend der Vorgehensweise zur Generierung des öffentlichen Schlüssels durch den Nutzer.

Weiterhin wird nun darauf eingegangen, wie eine Signatur für die zweite Stufe der Authentifizierung beim Hidden Service direkt erzeugt wird. Diese Funktion wurde gewählt, da große Teile des Algorithmus auch bei der Authentifizierung gegenüber dem Introduction Point eingesetzt werden, jedoch auf verschiedene Nachrichten verteilt sind. Die Erzeugung der Signatur erfolgt in der Funktion crypto_gq_generate_signature, die als Eingabeparameter die entsprechende Schlüsselumgebung besitzt, deren Erzeugung bereits dargestellt wurde. In Listing 3 wird die Generierung der Signatur verdeutlicht. Dazu werden zunächst verschiedene BIGNUM Variablen ebenso wie Zeichenketten, die Pufferfunktion besitzen (buf) und der Ausgabeparameter (sig_out) initialisiert. Im ersten Abschnitt (Zeile 22-25) erzeugt der Nutzer zunächst die Testnummer, indem er eine Zufallszahl generiert, die kleiner als der vom Hidden Service vorgegebene Modus ist und diese im Restklassenring Modus mod mit dem systemweit bekannten Exponenten (65537) potenziert. In einem zweiten Schritt wird die Nachricht (M), die signiert werden soll, erzeugt. Die Nachricht besteht aus zwei Bestandteilen, einem Zeitstempel, der die Aktualität der Signatur gewährleisten soll und der eigenen User-ID, die einen Hashwert des öffentlichen Schlüssel darstellt. Zusammen mit der zuvor erzeugten Testnummer bildet die Nachricht den Eingabewert für die Hashfunktion (Zeile 39-42). Als Hashfunktion kommt SHA-1 zum Einsatz, die bereits in den OpenSSL Bibliotheken implementiert ist. Da der erzeugte Hashwert im Bereich von 0 bis $exp - 1$ (65536) liegen muss, wird dies in einem anschließenden Schritt durch eine entsprechende Modulo-Operation gewährleistet. Der so errechnete Wert d stellt

die Challenge und neben der Nachricht M den zweiten Teil der Signatur dar. Im nun folgenden Abschnitt (Zeile 48-50) erfolgt in zwei Schritten die Berechnung des Beweises ($d2$), indem der private Schlüssel zunächst mit d potenziert wird und anschließend mit der in Zeile 23 erzeugten Zufallszahl multipliziert wird (alle Operationen im Restklassenring des festgelegten Modul). In einem letzten Schritt werden die BIGNUM-Variablen d und $d2$ hexadezimal codiert und somit für einen Nachrichtenversand vorbereitet, ehe sie mit der Nachricht M zu der gewünschten Signatur zusammengesetzt werden.

In einem nächsten Listing soll dargestellt werden, wie eine Zugangskontrolle am Hidden Service bei der zweiten Authentifizierung stattfindet. Bei Fehlern, die bei der Zugangskontrolle auftreten, erfolgt jeweils keinerlei Nachricht an den Client oder Introduction Point, um das Verkehrsaufkommen und damit die Angriffe mittels Verkehrsanalysen möglichst gering zu halten. Die Nachricht (Introduce2_Cell) wird dann vom Hidden Service verworfen. Zunächst wird vom Hidden Service überprüft, ob der geforderte Authentifizierungsmechanismus (service->auth_mech) und der vom Client angegebene Mechanismus (get_uint16(buf+pos)) übereinstimmen. Darauffolgend können unterschiedliche Authentifizierungsmechanismen an dieser Stelle unterschieden werden. Dies ist auf Grund der geforderten generischen Implementierung notwendig. Bisher wird lediglich zwischen GQ-Authentifizierung (Zeile 2) und keiner Authentifizierung (Zeile 33) unterschieden. In einem zweiten Schritt wird überprüft, ob die erhaltene Introduce2_Cell die geforderte Mindestlänge besitzt. Ist dies nicht der Fall, wird die Zelle verworfen. Anschließend (Zeile 12-17) wird die 20 Byte lange User_ID aus der Nachricht extrahiert und anhand der eigenen Access-Table überprüft, ob der Nutzer mit der angegebenen ID eine Zugangsberechtigung für den Hidden Service besitzt. Befindet sich der Nutzer in der Access-Table, wird anschließend die Aktualität der erzeugten Signatur anhand des Zeitstempels aus der Nachricht (M) kontrolliert. Dies ist notwendig, um Angriffe durch Wiedereinspielen einer alten Nachricht zu verhindern. Damit eine Signatur gültig ist, darf sich der Zeitstempel weder in der Zukunft befinden, noch darf die Zeitspanne zwischen Erstellung und Validierung zu groß sein (GQ_AUTH_MAX_AGE - z.Zt. 60 Sekunden). Dies macht eine Synchronisation der Uhren erforderlich. In einem letzten Schritt wird die Gültigkeit der eigentlichen Signatur überprüft. Dies geschieht mit Hilfe der Funktion crypto_gq_check_signature, die im nächsten Abschnitt erläutert wird. Sind alle Prüfungen positiv, ist eine Authentifizierung erfolgreich und der Hidden Service antwortet auf die Anfrage des Nutzers.

In einem letzten Listing 5 soll darauf eingegangen werden, wie eine vom Client erzeugte Signatur durch den Hidden Service auf Gültigkeit überprüft werden kann. Zunächst ist es erforderlich, dass unterschiedliche BIGNUM-Variablen und Zeichenketten initialisiert werden. In einer ersten Operation (Zeile 20-28) wird eine Testnummer durch den Hidden Service errechnet. Dazu werden die einzelnen Teile der Signatur-Zeichenkette zunächst in BIGNUM-Variablen umgewandelt und der öffentliche Schlüssel mit der mitgelieferten Challenge (d) sowie der Beweis ($d2$) mit dem Exponenten (exp) expotenziert. In einem zweiten Schritt (Zeile 30-37) wird zunächst der Input für die Hashfunktion, bestehend aus mitgelieferter Nachricht (M) und generierter Testnummer ($testnumber$), zusammengesetzt und an die Hashfunktion (SHA-1), die durch die OpenSSL-Bibliothek bereit gestellt wird, übergeben. Der so generierte Hashwert wird in einem folgenden Schritt durch eine Modulo-Operation auf einen geforderten Wertebereich von 0 bis 65536 ($exp - 1$) gebracht (Zeile 39-42). In Zeile 47 wird letztendlich entschieden, ob erzeugte und erhaltene Signatur identisch sind, und es erfolgt eine entsprechende Nachricht an die aufrufende Stelle (Zeile 50 bzw. Zeile 54).

6.5 Initialisierung des Systems

In diesem Abschnitt soll dargestellt werden, welche Konfigurationseinstellungen und Maßnahmen auf Seiten des Clients sowie des Hidden Service notwendig sind, um ein System mit Authentifizierung zu initialisieren. Weiterhin soll aufgezeigt werden, wie ein Tor-Netzwerk als lokale Testumgebung eingerichtet sein muss, um Tests der Systeme unabhängig vom bestehenden globalen Tor-Netzwerk durchführen zu können.

Zentrale Einstellungen an der Konfiguration des Systems können mit Hilfe der sog. `torrc`-Datei vorgenommen werden, die sich standardmäßig bei Windows im Benutzerprofil unter C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten\tor befinden bzw. bei Linux unter /etc/tor/torrc oder /etc/torrc zu finden ist. Über Einträge in dieser Konfigurationsdatei können die Einstellungen eines Tor-Knotens verändert werden. Eine Dokumentation aller Einträge, die in der Konfigurationsdatei vorgenommen werden können, ist auf der Homepage des Projektes[11] verfügbar. Zu beachten ist, dass vorgenommene Änderungen in der Konfigurationsdatei erst bei einem Neustart des Programms wirksam werden. Alternativ kann ein erneutes Einlesen der Konfigurationsdatei über einen Tor-Controller erreicht werden. Eine Überprüfung auf syntaktische Korrektheit erfolgt dann ebenfalls. Um einen Hidden Service betreiben zu können, sind mindestens die folgenden Einträge in der Konfigurationsdatei vorzunehmen:

- `HiddenServiceDir <Verzeichnis>` – Spezifiziert, in welchem Verzeichnis die Daten des Hidden Service gespeichert werden.
- `HiddenServicePort <Port1> <Adresse>:<Port2>` – Spezifiziert, auf welchem virtuellen Port1 der Service erreichbar ist und an welche Adresse und welchen Port2 die Anfrage weitergeleitet wird.

Da auf einem Tor-Knoten mehrere Hidden Services betrieben werden können, sind für jeden einzelnen Dienst diese Einträge vorzunehmen. Um eine Authentifizierung einzurichten, ist zusätzlich der Eintrag `HiddenServiceAuthenticationMech <Mech>` in der Konfigurationsdatei für den jeweiligen Hidden Service vorzugeben. `<Mech>` spezifiziert dabei die Art der Authentifizierung. Zur Zeit ist lediglich der Eintrag `1` erlaubt, der eine Authentifizierung mittels des Guillou-Quisquater-Protokolls vorschreibt. Durch diese Einträge werden beim ersten Start des Hidden Service im angegebenen Verzeichnis drei Dateien erzeugt, die entsprechend den privaten Schlüssel des Services (private_key), den Hostnamen (hostname) und den gemeinsamen Modul (md) enthalten.

Auf Seiten des Clients ist ebenfalls eine Einstellung an der Konfigurationsdatei `torrc` vorzunehmen. Mit dem Eintrag `GQAuthentication <Liste>` wird durch eine kommaseparierte Liste angegeben, auf welche Onion-Adressen mit Hilfe einer GQ-Authentifizierung zugegriffen werden soll. Entsprechend werden für diese Dienste öffentliche und private Schlüssel erzeugt. Dazu muss im Unterverzeichnis \keys des oben spezifizierten allgemeinen Tor-Verzeichnisses zunächst ein mit der Onion-Adresse benanntes Unterverzeichnis angelegt werden. In dieses Unterverzeichnis muss der gemeinsame Modul des Hidden Service, der außerhalb des Protokolls ausgetauscht wird, kopiert werden. Beim Start des Clients erzeugt dieser, wenn noch kein Schlüsselpaar vorhanden ist, automatisch den privaten und den öffentlichen Schlüssel und lädt die Schlüsselinformationen. Der öffentliche

[11] http://tor.eff.org/tor-manual-dev.html [Stand: 2006.09.27]

Schlüssel muss nun mit dem Hidden Service ebenfalls außerhalb des Tor-Protokolls ausgetauscht werden. Der Dienstanbieter platziert die Schlüsseldatei im Datenverzeichnis des entsprechenden Hidden Service. Um die Schlüsselinformationen zu übernehmen, ist ein Neustart des Tor-Knotens auf Seiten des Dienstanbieters notwendig. Der Client kann nun über eine GQ-Authentifizierung auf den Dienst zugreifen.

Abschließend soll darauf eingegangen werden, welche Einstellungen und Maßnahmen notwendig sind, um ein lokales Tor-Netzwerk aufzusetzen und Tests der Implementierung durchführen zu können. Um ein eigenes lokales Tor-Netzwerk betreiben zu können, ist es erforderlich, dass mindestens zwei Tor-Directory-Server und zusätzlich drei Tor-Server ohne Directory-Funktionalität betrieben werden. Von allen Server-Knoten müssen den Directory-Servern die sog. „Fingerprints“ (Hashwert der öffentlichen Schlüssel) bekannt sein. Fingerprints können erzeugt werden, indem der Befehl `tor -list-fingerprint` ausgeführt wird. Diese Fingerprints werden in einer im Tor-Verzeichnis der beiden Directory-Server neu zu erstellenden Datei „approved-routers“ zeilenweise eingetragen.

Für jeden der fünf Tor-Knoten müssen außerdem in der Konfigurationsdatei zumindest die folgenden Einträge vorgenommen werden:

- `ORPort <Port>` – Port, auf dem der Server Verbindungen von anderen Tor-Knoten entgegennimmt.
- `Address <IP-Address>` – IP-Adresse des Knotens.
- `Nickname <Name>` – Name, der diesen Server kennzeichnet.
- `ContactInfo <Info>` – Information, wie der Serverbetreiber erreichbar ist, beispielsweise eine E-Mail-Adresse.
- `AssumeReachable <0|1>` – Diese Option sollte auf `1` gesetzt werden, um einen Erreichbarkeitstest zu umgehen.
- `DirServer <Nickname Adress:DirPort Fingerprint>` – Gibt den Nickname, die IP-Adresse, den Directory-Port und den Fingerprint eines Directory-Servers an. Die Informationen beider Directory-Server müssen hier zeilenweise eingetragen werden.

Zusätzlich zu den obigen Einstellungen müssen bei den beiden Directory-Servern folgende Einträge in der Konfigurationsdatei vorgenommen werden:

- `DirPort <Port>` – Port, auf dem die Directory-Informationen verteilt werden.
- `AuthoritativeDirectory <0|1>` – Dieser Eintrag sollte auf `1` gesetzt werden, damit die Directory-Server eigene Informationen über das Tor-Netzwerk erstellen.
- `RecommendedVersions <Liste>` – Die kommaseparierte Liste, die angibt, welche Software-Versionen vom Directory-Server akzeptiert werden. Zumindest die verwendete Version sollte hier eingetragen werden.
- `VersioningAuthoritativeDirectory <0|1>` – Ist dieser Eintrag auf `1` gesetzt, übernimmt das Directory die Versionsverwaltung.

- `DirAllowPrivateAdresses <0|1>` – Um private IP-Adressen nutzen zu können, muss dieser Eintrag auf `1` gesetzt werden.

Sind diese Einträge vorgenommen worden, kann ein privates Tor-Netzwerk betrieben werden, bei dem eigene Directory-Server die Routerverwaltung übernehmen. Auf Grund der Unabhängigkeit von einem externen globalen Tor-Netzwerk ist es damit möglich, Erweiterungen am Quellcode vornehmen zu können und durch entsprechende Testläufe zu überprüfen.

6.6 Test der Implementierung

In diesem Abschnitt soll nun darauf eingegangen werden, welche Maßnahmen und Tests durchgeführt wurden, um eine anforderungsgemäße Funktionsweise der vorgestellten Implementierung zu gewährleisten. Zunächst wurden einzelne neu implementierte Funktionen mit Hilfe von CUnit-Tests daraufhin überprüft, ob die einzelnen Module das geforderte Verhalten unter bestimmten Rahmenbedingungen und Parametern aufweisen. CUnit[12] ist ein Framework, das die automatisierte Durchführung dieser Modultests unterstützt. Dies bietet sich insbesondere dann an, wenn getestete Komponenten weitgehend unabhängig vom Systemumfeld deterministisch arbeiten sollen. Unit-Tests sind dann schwierig durchzuführen, wenn mehrere Komponenten interagieren und diese Interaktionen über mehrere Knoten verteilt sind. Eine Simulation und anschließende Validierung des Verhaltens mit Unit-Tests ist dann äußerst komplex und teilweise nicht möglich. Da insbesondere die kryptografischen Funktionen sich zum einen als sicherheitskritisch erweisen und hier auf Grund der teilweise komplexen Implementierung Fehler zu vermuten sind, zum anderen aber auch eine gute Validierung der Funktionen mit Unit-Tests dargestellt werden kann, wurde das Verhalten dieser Funktionen mit Hilfe von Unit-Tests überprüft. Die durchgeführten Unit-Tests sind dem Listing 6 im Anhang C zu entnehmen.

Da durch Unit-Tests, wie gezeigt, nur ein Teil der Implementierung auf korrektes Verhalten hin überprüft werden konnte und des Weiteren dadurch lediglich lokale Tests dargestellt werden konnten, wurden neben diesen Tests mit Hilfe des Frameworks noch Tests des gesamten Systems durchgeführt. Verteilte Testumgebungen mit einem eigenen lokalen Tor-Netzwerk wurden sowohl für das Betriebssystem Microsoft Windows XP als auch für SUSE Linux 9.0 (glibc Version 2.3.3) eingerichtet. Welche Details bei der Einrichtung eines lokalen Tor-Netzwerkes und bei Hidden Services, die Authentifizierung erfordern, zu beachten sind, wurde bereits im vorherigen Kapitel erläutert. Für beide Betriebssysteme wurden insgesamt vier Tor-Server und zusätzlich zwei Tor-Directories installiert. Auf einem Tor-Server wurde ein Hidden Service, der GQ-Authentifizierung fordert, initialisiert. Mit den weiteren Tor-Servern wurden für diesen Hidden Service passende Schlüssel ausgetauscht.

Zunächst wurde der Gutfall überprüft, indem mit allen eingerichteten Nutzern teilweise parallel auf den Hidden Service zugegriffen wurde. Hier sollte jeder Nutzer auf Grund der erzeugten Authentifizierungsdaten Zugang erhalten. Im Folgenden wurden dann mehrere Schlechtfälle in die Validierung mit einbezogen. Zu unterscheiden war, ob ein Introduction Point mit einem potentiellen Angreifer zusammenarbeitet oder nicht. Zunächst wurde

[12] http://cunit.sourceforge.net/ [Stand: 2006.09.27]

angenommen, dass der ausgewählte Introduction Point nicht mit dem Angreifer zusammenarbeitet. Hierbei wurde getestet, ob der Introduction Point Anfragen von Nutzern, die nicht im Besitz von gültigen Authentifizierungsdaten waren, blockiert und nicht an den Hidden Service weiterleitet. Dazu wurden nacheinander der öffentliche wie auch der private Schlüssel um jeweils ein Bit verändert. Beide Änderungen mussten zu einem Zurückweisen der Anfrage führen, da entweder der öffentliche Schlüssel sich nicht in der Zugriffsliste befunden hat oder ein falscher Beweis errechnet wurde. In einem zweiten Schritt wurde angenommen, dass der Introduction Point nicht-vertrauenswürdig sei und entsprechend alle Anfragen, egal ob berechtigt oder nicht, an den Hidden Service weiterleitet. Drei Testfälle waren hierbei zu unterscheiden: Der Nutzer besitzt einen öffentlichen Schlüssel, der nicht in der Zugriffsliste hinterlegt wurde, der Nutzer verwendet einen fehlerhaften privaten Schlüssel (dies führt zu einer ungültigen Signatur) oder der Nutzer verwendet eine Signatur mit einem nicht-korrekten Zeitstempel (Zeitstempel zu alt oder in der Zukunft). Bei allen drei Fällen soll der Hidden Service eine Anfrage des Clients verwerfen und keine Verbindung zum Client herstellen.

Beide Testverfahren haben gezeigt, dass die Implementierung der GQ-Authentifizierung das geforderte Verhalten aufweist und lediglich authentifizierte Nutzer einen Hidden Service nutzen konnten, nicht authentifiziert Clients hingegen abgewiesen wurden. Wenn auch auf beiden Systemumgebungen (Microsoft Windows XP und SUSE Linux) zahlreiche Tests durchgeführt wurden, sollte, im Gegensatz zu Windows XP, der Betrieb unter Linux als im Status der Entwicklung bezeichnet werden, da bei diesem Betriebssystem weniger umfangreiche Tests durchgeführt wurden. Auch wurde der Betrieb unteren andere Versionen von Linux bzw. Unix bisher nicht getestet. Der Einsatz der Implementierung unter Microsoft Windows XP kann hingegen als stabil gewertet werden.

7 Fazit und Ausblick

Die Ergebnisse der vorliegenden wissenschaftlichen Arbeit sollen in diesem letzten Kapitel zunächst resümiert werden, bevor ein Ausblick über weiterführende Fragestellungen, die mit der bearbeiteten Problemstellung zusammen hängen, aufgezeigt werden.

Ein erstes Ziel dieser wissenschaftlichen Arbeit war es, unterschiedliche Authentifizierungsverfahren anhand eines geeigneten Kriterienkatalogs daraufhin zu untersuchen, inwieweit eine Umsetzung in Verbindung mit Hidden Services in Tor möglich ist. Als reales Anwendungsszenario sollte bei dieser Arbeit der Betrieb eines anonymen Blog-Service oder einer anonymen Wiki-Site dienen, bei dem es angemeldeten Nutzern möglich ist, anonym Beiträge zu veröffentlichen und zu lesen, ohne dass Identitätsinformationen des Dienstbetreibers hierzu nötig waren. Die Authentifizierungsverfahren sollten insbesondere dazu geeignet sein, den Hidden Service vor Verkehrsanalysen, welche die Anonymität des Dienstes gefährden können, und vor Angriffen auf die Erreichbarkeit des Dienstes zu schützen. Ein ausgewähltes Verfahren sollte anschließend als Prototyp im Rahmen des Tor-Projektes umgesetzt werden.

Nachdem die theoretischen Grundlagen und definitorischen Abgrenzung in diesem Themengebiet zu Beginn der Arbeit geschaffen wurden, fand in Kapitel 5 eine Evaluierung von verschiedenen Authentifizierungsverfahren anhand eines aufgestellten Kriterienkatalogs statt. Die untersuchten Verfahren sollten vor allem eine Authentifizierung durch eine u.U. nicht-vertrauenswürdige dritte Partei (den Introduction Point) sowie eine zweistufige Authentifizierung (beim Introduction Point und beim Hidden Service) ermöglichen. Als Ergebnis dieser Evaluierung kann festgestellt werden, dass im Wesentlichen drei Authentifizierungsverfahren den aufgestellten Kriterien genügen können: Gruppen-Signatur-Verfahren, asymmetrische Challenge-Response-Authentifizierung sowie das Zero-Knowledge-Protokoll von Guillou und Quisquater. Während das erste Verfahren gegenüber den beiden anderen vor allem Vorteile bezüglich der gewährten Anonymität für einen Dienstnutzer aufweist, besitzt das Guillou-Quisquater-Protokoll gegenüber der asymmetrischen Challenge-Response-Authentifizierung den Vorteil, dass durch den Protokollablauf keine Degeneration der Schlüssel stattfindet. Da das Guillou-Quisquater-Authentifizierungsprotokoll alle Anforderungen des sechsteiligen Kriterienkatalogs vollständig erfüllt und sich als vorteilhaft im Bezug auf eine prototypische Umsetzung im Rahmen dieser Arbeit ausweist, wurde dieses Authentifizierungsverfahren für eine Implementierung ausgewählt.

In einem zweiten Teil der Arbeit wurde eine plattformübergreifende Implementierung des Guillou-Quisquater-Protokolls im Tor-Netzwerk umgesetzt. Somit konnte gezeigt werden, dass eine Integration von Authentifizierungsverfahren in die Hidden Service-Architektur des Tor-Projektes unter Einhaltung der aufgestellten Anforderungen an die Erweiterung, möglich ist. Ausgiebige Tests, insbesondere auf dem Betriebssystem Microsoft Windows XP, haben die Funktionsfähigkeit der Implementierung bewiesen. Dennoch sollte die Umsetzung im Rahmen dieser Arbeit als prototypisch angesehen werden. Zahlreiche Erweiterungen der bestehenden Implementierung sind denkbar. Interessant ist im Besonderen eine weitergehende Untersuchung der aktuellen Implementierung bezüglich einer Unterstützung von anderen Betriebssystemen (weitere Linux-Distributionen, Mac OS, Unix). Da eine Anbindung des implementierten Authentifizierungsverfahrens an die mitgelieferte grafische Benutzeroberfläche bzw. den Tor-Controller im Rahmen dieser prototypischen Umsetzung noch nicht stattgefunden hat, wird ein Betrieb von Hidden Services,

die eine Nutzer-Authentifizierung erfordern, auf Grund einer mangelnden Praktikabilität etwas erschwert. Daher sollte eine Anbindung an die grafische Benutzeroberfläche erfolgen, um eine breite Akzeptanz und eine einfachere Benutzung durch Dienstanbieter zu gewährleisten. Letztendlich ist eine Umsetzung von anderen, teilweise komplexeren Authentifizierungsverfahren (beispielsweise Gruppen-Signatur-Protokolle) im Rahmen des Tor-Projektes denkbar. Die entsprechende generische Datenstruktur der Nachrichten ist hierfür bereits geschaffen worden.

Wenn auch eine Implementierung eines Authentifizierungsverfahrens in Tor stattgefunden hat und somit eine Grundlage für eine Abwehr von Angriffen auf Hidden-Services geschaffen wurde, so sollte dennoch künftig beleuchtet werden, welches Angriffspotential auf Hidden Services weiterhin besteht. Durch den Einsatz von Authentifizierungsverfahren können Angriffe von nicht-berechtigten Nutzern unter der Voraussetzung, dass ein Introduction Point ordnungsgemäß arbeitet, vollständig ausgeschlossen werden. Jegliche Anfragen dieses Personenkreises werden bereits durch den Introduction Point geblockt. Sowohl Verkehrsanalysen als auch Angriffe auf die Erreichbarkeit des Service werden so unterbunden. Dennoch stellt sich die Frage, was passiert, wenn ein Introduction Point nicht ordnungsgemäß arbeitet oder Angriffe von berechtigten Nutzern stattfinden. Ein Zugriff auf den Service durch nicht-berechtigte Nutzer wird durch eine zweistufige Authentifizierung auf der Ebene des Hidden Service zwar unterbunden, dennoch scheinen sowohl Angriffe durch Verkehrsanalysen als auch auf die Erreichbarkeit des Services möglich. Um dies zu verhindern und eventuell korrumpierte Introduction Points bzw. Angriffe von zugangsberechtigten Nutzern zu erkennen, sind weitere Gegenmaßnahmen notwendig. Eine denkbare Erweiterung, mit der die genannten Angriffe u.U. erkannt und unterbunden werden könnten, ist in Form der Integration von Ansätzen zur Vertrauensbildung zu sehen, deren Umsetzung jedoch über den Kontext dieser Arbeit hinausgeht.

Literatur

[1] A. Pfitzmann and M. Köhntopp, "Anonymity, Unlinkability, Unobservability, Pseudeonymity, and Identity Management – A Consolidated Proposal for Terminology," pp. 1–31, 2006. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.28.pdf[Stand:2006.09.27]

[2] I. Goldberg, "A Pseudonymous Communications Infrastructure for the Internet," Ph.D. dissertation, University of California at Berkeley, 2000.

[3] T. Demuth, *Ein Beitrag zur Anonymität in Kommunikationsnetzen.* Shaker Verlag, 2003.

[4] H. Federrath and A. Pfitzmann, "Bausteine zur Realisierung mehrseitiger Sicherheit," in *Mehrseitige Sicherheit in der Kommunikationstechnik*, G. Müller and A. Pfitzmann, Eds. Addison-Wesley-Longman, 1997, pp. 83–104.

[5] V. Scarlata, B. Levine, and C. Shields, "Responder Anonymity and Anonymous Peer-to-Peer File Sharing," in *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols.* IEEE Computer Society, 2001, pp. 272–280.

[6] I. Goldberg and D. Wagner, "TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web," *First Monday*, vol. 3, no. 4, pp. 1–14, 1998. [Online]. Available: http://www.firstmonday.dk/issues/issue3_4/goldberg/index.html[Stand:2006.09.27]

[7] R. Dingledine, "Tor Hidden Services." Presentation 'What The Hack', 2005. [Online]. Available: http://freehaven.net/~arma/wth3.pdf[Stand:2006.09.27]

[8] jrandom (Pseudonym), *Introducing I2P - a scalable framework for anonymous communication.* [Online]. Available: http://dev.i2p.net/cgi-bin/cvsweb.cgi/i2p/router/doc/techintro.html?rev=HEAD[Stand:2006.09.27]

[9] B. Schneier, *Applied Cryptography. Protocols, Algorithms and Source Code in C*, 2nd ed. John Wiley & Sons, Inc, 1996.

[10] R. L. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret," in *Advances in Cryptology – ASIACRYPT '01*, C. Boyd, Ed. Springer-Verlag, LNCS 2248, 2001, pp. 552–565.

[11] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[12] L. Øverlier and P. Syverson, "Locating Hidden Servers," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy.* IEEE CS, May 2006.

[13] M. J. Freedman and R. Morris, "Tarzan: A Peer-to-Peer Anonymizing Network Layer," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, V. Atluri, Ed. ACM Press, 2002, pp. 193–206.

[14] O. Landsiedel, H. Niedermeyer, and K. Wehrle, "An Infrastructure for Anonymous Internet Services," 1st International Workshop on Innovations In Web Infrastructure, pp. 1–5, 2005.

[15] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, 5th ed. CRC Press, 2001.

[16] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[17] D. Chaum and E. van Heyst., "Group Signatures," in *Advances in Cryptology – EUROCRYPT '91*, D. W. Davies, Ed. Springer-Verlag, LNCS 547, 1991, pp. 257–265.

[18] Secretariat ISO/IEC JTC 1/SC 27, "ISO IS 15408 – Evaluation criteria for IT security – Part 2: Security functional requirements," 2003.

[19] B. Flinn and H. Maurer, "Levels of Anonymity," *Journal of Universal Computer Science*, vol. 1, no. 1, pp. 35–47, 1995.

[20] O. Berthold, A. Pfitzmann, and R. Standtke, "The Disadvantages of Free MIX Routes and how to Overcome Them," in *Workshop on Design Issues in Anonymity and Unobservability*, H. Federath, Ed. Springer-Verlag, LNCS 2009, 2000, pp. 30–45.

[21] A. Serjantov, "On the Anonymity of Anonymity Systems," Ph.D. dissertation, University of Cambridge, 2004.

[22] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards Measuring Anonymity," in *Privacy Enhancing Technologies (PET 2002)*, R. Dingledine and P. Syverson, Eds. Springer-Verlag, LNCS 2482, 2002, pp. 54–68.

[23] T. Y. Woo and S. S. Lam, "Authentication for Distributed Systems," *Computer*, vol. 25, no. 1, pp. 39–52, 1992.

[24] C. Eckert, *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftsverlag, 2001.

[25] National Institute of Standards and Technology, "Federal Information Processing Standards 83 – Guideline on User Authentication Techniques for Computer Network Access Control," 1980.

[26] M. Benantar, *Access Control Systems: Security, Identity Management and Trust Models*. Springer-Verlag, 2006.

[27] R. Oppliger, *Authentication Systems for Secure Networks*. Artech House, Inc., 1996.

[28] H. Delfs and H. Knebl, *Introduction to Cryptography. Principles and Applications*. Springer Verlag, 2002.

[29] M. R. A. Huth, *Secure Communicating Systems*. Cambridge University Press, 2001.

[30] M. K. Reiter and A. D. Rubin, "Anonymous Web transactions with Crowds," *Communications of the ACM*, vol. 42, no. 2, pp. 32–48, 1999.

[31] P. A. Karger, "Non-Discretionary Access Control for Decentralized Computing Systems," Massachusetts Institute of Technology, Tech. Rep., 1977.

[32] D. Farber and K. Larson, "Network Security Via Dynamic Process Renaming," in *Fourth Data Communications Symposium*, 1975.

[33] H. Federrath and A. Pfitzmann, "'Neue' Anonymitätstechniken," *Datenschutz und Datensicherheit*, vol. 22, no. 11, pp. 628–632, 1998.

[34] D. Chaum, "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability ," *Jornal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[35] D. L. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[36] H. Federrath and K. Martius, "Anonymität und Authetizität im World Wide Web," in *Internet - frischer Wind in der Telekommunikation*. VDE-Verlag, 1998, pp. 91–101.

[37] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[38] R. Dingledine, "The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven," Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2000.

[39] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an Analysis of Onion Routing Security," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, 2000, pp. 96–114.

[40] M. Rennhard and B. Plattner, "Practical Anonymity for the Masses with Mix-Networks," in *Proceedings of the IEEE 8th Intl. Workshop on Enterprise Security (WET ICE 2003)*. IEEE Computer Society, 2003, pp. 255–260.

[41] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous Connections and Onion Routing," in *IEEE Symposium on Security and Privacy*, 1997, pp. 44–54.

[42] R. Dingledine, N. Mathewson, and P. Syverson, "Challenges in deploying low-latency anonymity (DRAFT)," 2005, unpublished Manuscript. [Online]. Available: http://tor.eff.org/cvs/tor/doc/design-paper/challenges.pdf[Stand:2006.09.27]

[43] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley, "Protecting Free Expression Online with Freenet," *IEEE Internet Computing*, vol. 6, no. 1, pp. 40–49, 2002.

[44] M. Waldman, A. D. Rubin, and L. F. Cranor, "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System," in *Proceedings of the. 9th USENIX Security Symposium*, 2000, pp. 59–72.

[45] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding Routing Information," in *Information Hiding, First International Workshop*, R. Anderson, Ed. Springer-Verlag, LNCS 1174, 1996, pp. 137–150.

[46] R. Dingledine and N. Mathewson, *Tor Rendezvous Spezification, v. 1.31*, 2006. [Online]. Available: http://tor.eff.org/cvs/tor/doc/rend-spec.txt[Stand:2006.09.27]

[47] S. J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2005, pp. 183 –195.

[48] M. J. Freedman, E. Sit, J. Cates, and R. Morris, "Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer," in *Peer-to-Peer Systems, First International Workshop, IPTPS*, P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, Eds. Springer-Verlag, LNCS 2429, 2002, pp. 121–129.

[49] A. Bosselaers and B. Preneel, *Integrity Primitives for Secure Information Systems: Final Ripe Report of Race Integrity Primitives Evaluation.* Springer-Verlag, LNCS 1007, 1995.

[50] International Organization for Standardization, "ISO/IEC 9798-2 Information technology – Security techniques – Entity authentication mechanisms – Part 2: Mechanisms using symmetric encipherment algorithms," 1994.

[51] U. Feige, A. Fiat, and A. Shamir, "Zero-Knowledge Proofs of Identity," *Journal of Cryptology*, vol. 1, no. 2, pp. 77 – 94, 1988.

[52] L. C. Guillou and J.-J. Quisquater, "A "Paradoxical" Identity-based Signature Scheme Resulting from Zeroknowledge," in *Advances in Cryptology –CRYPTO '88*, S. Goldwasser, Ed. Springer-Verlag, LNCS 403, 1990, pp. 216–231.

[53] C. P. Schnorr, "Efficient Identification and Signatures for Smart Cards," in *Advances in Cryptology – CRYPTO '89*, G. Goos and J. Hartmanis, Eds. Springer-Verlag, LNCS 435, 1989, pp. 239–252.

[54] K. Ohta and T. Okamoto, "A Modification of the Fiat-Shamir Scheme," in *Advances in Cryptology – CRYPTO '88*, S. Goldwasser, Ed. Springer-Verlag, LNCS 403, 1990, pp. 232–243.

[55] E. F. Brickell and K. S. McCurley, "An Interactive Identification Scheme Based on Discrete Logarithms and Factoring," *Journal of Cryptology*, vol. 5, no. 1, pp. 29–39, 1992.

[56] M. Girault, "An Identity-Based Identification Scheme Based on Discrete Logarithms Modulo a Composite Number," in *Advances in Cryptology – EUROCRYPT '90*, I. B. Damgård, Ed. Springer-Verlag, LNCS 473, 1991, pp. 481–486.

[57] G. Brassard and C. Crepeau, "Sorting out zero-knowledge," in *Advances in Cryptology – EUROCRYPT '89*, J. Quisquater and J. Vandewalle, Eds. Springer-Verlag, LNCS 434, 1990, pp. 181–191.

[58] J. J. Quisquater, L. C. Guillou, M. Annick, and T. A. Berson, "How to Explain Zero-Knowledge Protocols to Your Children," in *Advances in Cryptology – CRYPTO '89*, G. Brassard, Ed. Springer-Verlag, LNCS 435, 1989, pp. 628–631.

[59] A. Fiat and A. Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems," in *Advances in Cryptology – CRYPTO '86*, A. M. Odlyzko, Ed. Springer-Verlag, LNCS 263, 1987, pp. 186–194.

[60] L. C. Guillou and J. J. Quisquater, "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory," in *Advances in Cryptology – EUROCRYPT '88*, C. G. Günther, Ed. Springer-Verlag, LNCS 330, 1988, pp. 123–128.

[61] W. Mao, *Modern Cryptography: Theory and Practice*. Prentice Hall Professional Technical Reference, 2003.

[62] J. Camenisch and J. Groth, "Group Signatures: Better Efficiency and New Theoretical Aspects," in *Security in Communication Networks, 4th International Conference, Revised Selected Papers*, C. Blundo and S. Cimato, Eds. Springer-Verlag, LNCS 3352, 2004, pp. 120–133.

[63] J. L. Camenisch, "Efficient and Generalized Group Signatures," in *Advances in Cryptology – EUROCRYPT '97*, W. Fumy, Ed. Springer-Verlag, LNCS 1233, 1997, pp. 465–479.

[64] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology – CRYPTO '84*, G. R. Blakley and D. Chaum, Eds. Springer-Verlag, LNCS 196, 1985, pp. 10–18.

[65] J. L. Camenisch and M. A. Stadler, "Efficient Group Signature Schemes for Large Groups," in *Advances in Cryptology – CRYPTO '97*, B. Kaliski, Ed. Springer-Verlag, LNCS 1294, 1997, pp. 410–424.

[66] G. Ateniese, D. X. Song, and G. Tsudik, "Quasi-Efficient Revocation in Group Signatures," in *Financial Cryptography, 6th International Conference, Revised Papers*, M. Blaze, Ed. Springer-Verlag, LNCS 2357, 2002, pp. 183–197.

[67] E. Bresson and J. Stern, "Efficient Revocation in Group Signatures," in *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, K. Kim, Ed. Springer-Verlag, LNCS 1992, 2001, pp. 190–206.

[68] J. Camenisch and A. Lysyanskaya, "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials," in *Advances in Cryptology – CRYPTO '02*, M. Yung, Ed. Springer-Verlag, LNCS 2442, 2002, pp. 61–76.

[69] A. D. Santis, G. D. Crescenzo, and G. Persiono, "Communication-Efficient Anonymous Group Identification," in *5th A.C.M. Conference on Computer and Communications Security (ACM CCS'98)*, L. Gong and M. Reiter, Eds. ACM Press, 1998, pp. 73–82.

[70] S. Schechter, T. Parnell, and A. Hartemink, "Anonymous Authentication of Membership in Dynamic Groups," in *Proceedings of the International Conference on Financial Cryptography 99*, M. Franklin, Ed. Springer Verlag, LNCS 1648, 1999, pp. 184–195.

[71] B. Handley, "Resource-Efficient Anonymous Group Identification," in *FC '00: Proceedings of the 4th International Conference on Financial Cryptography*, Y. Frankel, Ed. Springer-Verlag, LNCS 1962, 2001, pp. 295–312.

[72] S. Gohil and S. T. Fleming, "Attacks on Homage Anonymous Group Authentication Protocol," Department of Computer Science, University of Otago, Tech. Rep., 2004.

[73] S. T. Fleming and S. Gohil, "Further Reflection on Homage Anonymous Group Authentication Protocol," Department of Computer Science, University of Otago, Tech. Rep., 2004.

[74] R. Dingledine and N. Mathewson, *Tor directory protocol for 0.1.1.x series, v. 1.48*, 2006. [Online]. Available: http://tor.eff.org/cvs/tor/doc/dir-spec.txt[Stand: 2006.09.27]

[75] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL*. O'Reilly & Associates, Inc, 2002.

[76] J. Viega and M. Messier, *Secure Programming Cookbook for C and C++*. O'Reilly & Associates, Inc, 2003.

A Aktivitätsdiagramme

.

B Datenobjekte

V		Format Oktette (255)	[1 Oktett]	signiert
V		Versions Oktette (1)	[1 Oktett]	
KL		Schlüssellänge	[2 Oktette]	
PK		Public Key von HS	[128 Oktette]	
TS		Zeitstempel	[4 Oktette]	
PROTO		Rendezvous Protokoll Ver. (Bitmaske)	[2 Oktette]	
NA		Nummer der Authentifizierungsmech.	[1 Oktett]	
Für jeden A. Mechanismus	AUTHT	Authentifizierungstyp (hier: 1)	[2 Oktette]	
	AUTHL	Länge der Auth. Daten (hier: 0)	[2 Oktette]	
	AUTHD	Authentifizierungsdaten (hier: NULL)	[AUTHL Oktette]	
NI		Anzahl der Introduction Points	[2 Oktette]	
Für jeden IPT	ATYPE	Addresstyp (hier: 4)	[1 Oktett]	
	ADDR	IP-Addresse von IPT	[4 Oktette]	
	PORT	OR port von IPT	[2 Oktette]	
	AUTHT	Authentifizierungstyp	[2 Oktette]	
	AUTHL	Läge der Authentifizierungsdaten	[1 Oktett]	
	AUTHD	Authetifizierungsdaten	[variabel]	
	ID	Rendezvous point ID	[20 Oktette]	
	KLEN	Länge des onion key	[2 Oktette]	
	KEY	Rendezvous point onion key	[KLEN Oktette]	
SIG		Signatur der obigen Felder	[variabel]	

Tabelle 4: Datenobjekt RENDEZVOUS_SERVIVE_DESCRIPTOR

KL		Schlüssellänge	[2 Oktette]	signiert
PK		Public Key von HS	[128 Oktette]	
HS		Hash der Sessioninformation	[20 Oktette]	
AUTHT		Authentifizierungstyp (hier: 1)	[2 Oktette]	
AUTHL		Länge der Auth. Daten (hier: 256 + ATLEN * 20)	[2 Oktette]	
AUTHD	MOD	Modul für GQ-Berechnung	[256 Oktette]	
	ATLEN	Anzahl der Elemente in der AT	[2 Oktette]	
	AT	Accesstable (IDs von Clients)	[ATLEN*20 Oktette]	
SIG		Signatur der obigen Felder	[variabel]	

Tabelle 5: Datenobjekt RELAY_ESTABLISH_INTRO_CELL

HS_ID	ID Public Key von HS	[20 Oktette]	nicht verschl.
PUK	Public Userkey	[256 Oktette]	
TEST	Testnummer	[256 Oktette]	

Tabelle 6: Datenobjekt RELAY_ACCESS_CELL

QUES	Zufallszahl / Challenge	[2 Oktette]	nicht verschl.

Tabelle 7: Datenobjekt RELAY_CHALLENGE_CELL

PK_ID	ID Public Key von HS	[20 Oktette]	nicht verschl.
AUTHT	Authentifizierungstyp	[2 Oktette]	
AUTHL	Länge der Auth. Daten (hier:256)	[2 Oktette]	
AUTHD	Authentifizierungsdaten (hier: GQ-Beweis	[AUTHL Oktette]	
VER	Versionsnummer (hier: 4)	[1 Oktett]	verschlüsselt mit public key von HS
ATYPE	Addresstyp (hier: 4)	[1 Oktett]	
ADDR	IP-Addresse von RPT	[4 Oktette]	
PORT	OR port von RPT	[2 Oktette]	
AUTHT	Authentifizierungstyp (hier: 1)	[2 Oktette]	
AUTHL	Länge der Auth. Daten (hier: 284)	[2 Oktette]	
AUTHD	Authentifizierungsdaten (hier: User_ID, Timestamp, Signaturdaten[d, D])	[AUTHL Oktette]	
ID	Rendezvous point ID	[20 Oktette]	
KLEN	Länge des onion key	[2 Oktette]	
KEY	Rendezvous point onion key	[KLEN Oktette]	
RC	Rendezvous cookie	[20 Oktette]	
g^x	Diffie-Hellman Daten, Teil 1	[128 Oktette]	

Tabelle 8: Datenobjekt RELAY_INTRODUCE1_CELL

PK_ID	ID Public Key von HS	[20 Oktette]	nicht verschl.
VER	Versionsnummer (hier: 4)	[1 Oktett]	verschlüsselt mit public key von HS
ATYPE	Addresstyp (hier: 4)	[1 Oktett]	
ADDR	IP-Addresse von RPT	[4 Oktette]	
PORT	OR port von RPT	[2 Oktette]	
AUTHT	Authentifizierungstyp (hier: 1)	[2 Oktette]	
AUTHL	Länge der Auth. Daten (hier: 284)	[2 Oktette]	
AUTHD	Authentifizierungsdaten (hier: User_ID, Timestamp, Signaturdaten[d, D])	[AUTHL Oktette]	
ID	Rendezvous point ID	[20 Oktette]	
KLEN	Länge des onion key	[2 Oktette]	
KEY	Rendezvous point onion key	[KLEN Oktette]	
RC	Rendezvous cookie	[20 Oktette]	
g^x	Diffie-Hellman Daten, Teil 1	[128 Oktette]	

Tabelle 9: Datenobjekt RELAY_INTRODUCE2_CELL

C Quellcode ausgewählter Erweiterungen

Listing 1: Erweiterung Rendezvous Service Descriptor

```
/* encoding of RSD */

[...]
if (version == 1) {
        *(uint16_t*)ptr = (uint16_t)desc->auth_mech;
        ptr += 2;
        if (desc->auth_mech == GQ_AUTH){
                *(uint16_t*)ptr = (uint16_t)0;
                ptr += 2;
        }
}
[...]



/* decoding of RSD */

[...]
 if (version == 1) {
        if (end-ptr < 2) goto truncated;
        result->auth_mech = *(uint16_t*)(ptr);
        ptr +=2;
        if (result->auth_mech == GQ_AUTH){
                if (end-ptr < 2) goto truncated;
                result->auth_length = 0;
                result->auth_data = NULL;
                ptr +=2;
        }
}
[...]
```

Listing 2: Generieren von Schlüsseln für das Verfahren

```
/** Generate a new public/private userkeypair in <b>env</b>.  Return 0 on
 * success, -1 on failure.
 */
int
crypto_gq_generate_accesskey(crypto_accessk_env_t *env)
{
        BIGNUM *exp = BN_new();
        BIGNUM *tmp = BN_new();
        BN_CTX *c = BN_CTX_new();

        /* this is a secure generated prime of LK_BYTES Bytes length */
        BN_generate_prime(env->priv_key, LK_BYTES*8 , 0, NULL, NULL, NULL, NULL);
        BN_set_word(exp, GQ_EXP);
        BN_mod_exp(tmp, env->priv_key,  exp, env->mod, c);

        /* generate the inverse of the priv_key^exp as pub_key*/
        BN_mod_inverse(env->pub_key, tmp , env->mod, c);
        if (!env->pub_key || !env->priv_key) {
                crypto_log_errors(LOG_WARN, "generating auth_key (user)");
                goto error;
        }

        /* free everything used */
        [...]
}
```

Listing 3: Generierung einer Signatur

```
/** Generates a signature for the accesskey-environment with an actual time-stamp.
 * Return the sig. on success, NULL on error.
 */
char *crypto_gq_generate_signature(crypto_accessk_env_t *env){

        /* typedefinitions */
        time_t now = time(NULL);
        BIGNUM *exp = BN_new();
        BIGNUM *testnumber = BN_new();
        BIGNUM *random = BN_new();
        BIGNUM *tmp = BN_new();
        BIGNUM *d = BN_new();
        BIGNUM *d2 = BN_new();
        BN_CTX *ctx = BN_CTX_new();
        SHA_CTX sctx;
        char *buf = tor_malloc(LK_BYTES * 2 + 4 + DIGEST_LEN);
        char *buf2 = tor_malloc(20);
        char *buf3;
        char *sig_out = tor_malloc(4+4+DIGEST_LEN+LK_BYTES*2);
        char *ptr;

        /* Generate the testnumber*/
        BN_rand_range(random, env->mod);
        BN_set_word(exp, GQ_EXP);
        BN_mod_exp(testnumber, random, exp, env->mod, ctx);
        ptr = buf;

        /* M == time_t + userid */
        set_uint32(buf, htonl((uint32_t)now));
        buf += 4;
        strncpy(buf, env->user_id, DIGEST_LEN);
        buf += DIGEST_LEN;

        /* Testnumber T */
        buf3 = BN_bn2hex(testnumber);
        strncpy(buf, buf3, LK_BYTES *2);
        buf = ptr;

        /* Generate the Hash */
        SHA1_Init(&sctx);
        SHA1_Update(&sctx, buf, DIGEST_LEN+4+LK_BYTES*2);
        SHA1_Final(buf2, &sctx);

        /* Hash has to be < exp */
        BN_bin2bn(buf2, DIGEST_LEN, tmp);
        BN_mod(d, tmp, exp, ctx);

        /* Generate d2 */
        BN_mod_exp(tmp, env->priv_key, d, env->mod, ctx);
        BN_mod_mul(d2, random, tmp, env->mod, ctx);

        /* put it all together for the signature: M, d, d2 */
        ptr = sig_out;
        strncpy(sig_out, buf, DIGEST_LEN + 4);
        sig_out += (DIGEST_LEN + 4);
        strncpy(sig_out, BN_bn2hex(d), 4);
        sig_out += 4;
        strncpy(sig_out, BN_bn2hex(d2), LK_BYTES * 2);
        sig_out += (LK_BYTES * 2);
        sig_out = ptr;
        strcpy(sig_out+4+4+DIGEST_LEN+LK_BYTES*2, "\0");

        /* free everything used */
        [...]

        return sig_out;
}
```

Listing 4: Zugangskontrolle beim Hidden Service

```
if (service->auth_mech == (get_uint16(buf+pos))){
        if (service->auth_mech == GQ_AUTH){
                /* authentication mech is GQ_AUTH */
                crypto_usersk_env_t *env;
                time_t now = time(NULL);
                pos += 4;
                if ((int)len < pos + DIGEST_LEN + LK_BYTES *2 + 4 +4) {
                        log_warn(LD_PROTOCOL, "Bad length for version 2 INTRODUCE2
                                cell. Dropping cell");
                        return -1;
                }

                /* 20 Bytes user_id */
                env = ((crypto_usersk_env_t *)digestmap_get(service->user_keys,
                         buf+pos+4));
                if (!env){
                        log_warn(LD_REND, "User not in access-list, Rejecting
                                Access, Dropping cell" );
                        return -1;
                }

                /* 4 Byte timestamp */
                if ((time_t) ntohl(get_uint32(buf+pos)) + GQ_AUTH_MAX_AGE < now ||
                        ((time_t) ntohl(get_uint32(buf+pos)) > now)) {

                        log_warn(LD_REND, "Timestamp dirty: to old or in the
                                future. Rejecting Access, Dropping cell");
                        return -1;
                }
                /* Check the signature */

                if (crypto_gq_check_signature(env, service->modulus, buf+pos)){
                        log_warn(LD_REND, "Signature not valid. Rejecting Access,
                                Dropping cell" );
                        return -1;
                }
                pos += DIGEST_LEN + LK_BYTES *2 + 4 + 4;
        }else {
                /* authentication mech is NONE */
                pos += 4;
        }
}else{
        log_warn(LD_GENERAL, "required and given auth.mech differ, Rejecting
                access, dropping cell");
        goto err;
}
```

Listing 5: Überprüfen einer Signatur

```
/** Checks if the given siganture is ok for the user-key of the environment env.
 * Return 0 if the signature is ok, -1 if not.
 */
int crypto_gq_check_signature(crypto_usersk_env_t *env, BIGNUM *mod, char* sig){

        /* declaration */
        BIGNUM *d = BN_new();
        BIGNUM *d2 = BN_new();
        BIGNUM *exp = BN_new();
        BIGNUM *testnumber = BN_new();
        BIGNUM *tmp = BN_new();
        BIGNUM *tmp2 = BN_new();
        BN_CTX *ctx = BN_CTX_new();
        SHA_CTX sctx;
        char *buf = tor_malloc(LK_BYTES *2);
        char *buf2 = tor_malloc(4);
        char *buf3 = tor_malloc(LK_BYTES *2 + DIGEST_LEN + 4);
        BN_set_word(exp, GQ_EXP);

        /*Generate the Testnumber*/
        strncpy(buf, sig+DIGEST_LEN+8, LK_BYTES*2);
        BN_hex2bn(&d2, buf);
        BN_mod_exp(tmp, d2, exp, mod, ctx);
        strncpy(buf2, sig+DIGEST_LEN+4, 4);
        BN_hex2bn(&d, buf2);
        BN_mod_exp(tmp2, env->pub_key, d, mod, ctx);
        BN_mod_mul(testnumber, tmp, tmp2, mod, ctx);
        buf2 = BN_bn2hex(testnumber);

        /* M,TN -> input for h(x) */
        strncpy(buf3, sig, DIGEST_LEN + 4);
        strncpy(buf3+DIGEST_LEN+4, buf2, LK_BYTES *2);

        /* Generate the Hash */
        SHA1_Init(&sctx);
        SHA1_Update(&sctx, buf3, DIGEST_LEN+4+LK_BYTES*2);
        SHA1_Final(buf2, &sctx);

        /* Hash has to be < exp */
        BN_bin2bn(buf2, DIGEST_LEN, tmp);
        BN_mod(tmp2, tmp, exp, ctx);

        /* free everything that is used */
        [...]

        /* check if d == d' */
        if (strncmp(sig+DIGEST_LEN+4, BN_bn2hex(tmp2), 4)){
                log_info(LD_GENERAL, "Signature that is received is incorrect");
                BN_free(tmp2);
                return -1;
        }else{
                log_info(LD_GENERAL, "Signature that ist received is correct");
                BN_free(tmp2);
                return 0;
        }
}
```

Listing 6: Übersicht der CUnit-Testfälle

```
1
2  /** Test of crypto_gq_generate modulus(). Checks if the modul is produced
3   * or if it is read from a file and is equal to the produced.
4   */
5  void test_crypto_gq_generate_modulus(void)
6  {
7          CU_ASSERT_FALSE(crypto_gq_generate_modulus(test1, "../test"));
8          CU_ASSERT_STRING_NOT_EQUAL(BN_bn2dec(test1), "0");
9
10         CU_ASSERT_FALSE(crypto_gq_generate_modulus(accesskey->mod, "../test"));
11         CU_ASSERT_STRING_EQUAL(BN_bn2dec(test1), BN_bn2dec(accesskey->mod));
12
13         BN_set_word(test1, 0);
14         CU_ASSERT_FALSE(unlink("../md"));
15 }
16
17 /** Test of crypto_gq_generate accesskey(). Checks if access-keys are produced
18  * correctly so that S^v*J = 1 mod n.
19  */
20 void test_crypto_gq_generate_accesskey(void)
21 {
22         CU_ASSERT_FALSE(crypto_gq_generate_accesskey(accesskey));
23
24         CU_ASSERT_STRING_NOT_EQUAL(BN_bn2dec(accesskey->priv_key), "0");
25         CU_ASSERT_STRING_NOT_EQUAL(BN_bn2dec(accesskey->pub_key), "0");
26         BN_set_word(test1, GQ_EXP);
27         CU_ASSERT_TRUE(BN_mod_exp(test2, accesskey->priv_key, test1,
28                 accesskey->mod, ctx));
29         CU_ASSERT_TRUE(BN_mod_mul(test3, test2, accesskey->pub_key,
30                 accesskey->mod, ctx));
31         CU_ASSERT_STRING_EQUAL(BN_bn2dec(test3), "1");
32
33         BN_set_word(test1, 0);
34         BN_set_word(test2, 0);
35         BN_set_word(test3, 0);
36 }
37
38 /** Test of crypto_gq_write_access_key_to_filename(). Checks if no errors occured
39  * writing the access-keys. If the keys are stored correctly is checked at
40  * test_crypto_gq_read_userkey_from_file().
41  */
42 void test_crypto_gq_write_access_key_to_filename(void)
43 {
44         CU_ASSERT_FALSE(crypto_gq_write_access_key_to_filename(accesskey,
45                 "../access"));
46 }
47
48 /** Test of crypto_gq_read_userkey_from_file(). Checks if keys are written
49  * and read correctly from files.
50  */
51 void test_crypto_gq_read_userkey_from_file(void)
52 {
53         CU_ASSERT_FALSE(crypto_gq_read_userkey_from_file(test1, "../access"));
54         CU_ASSERT_FALSE(crypto_gq_read_userkey_from_file(test2, "../access_pub"));
55
56         CU_ASSERT_STRING_EQUAL(BN_bn2dec(test1), BN_bn2dec(accesskey->priv_key));
57         CU_ASSERT_STRING_EQUAL(BN_bn2dec(test2), BN_bn2dec(accesskey->pub_key));
58
59         CU_ASSERT_FALSE(unlink("../access_pub"));
60         CU_ASSERT_FALSE(unlink("../access"));
61
62         BN_set_word(test1, 0);
63         BN_set_word(test2, 0);
64 }
65
66 /** Test of crypto_gq_generate userid(). Checks if user-IDs are produced
67  * correctly and are the same if same keys are given.
68  */
69 void test_crypto_gq_generate_userid(void)
70 {
71         CU_ASSERT_STRING_NOT_EQUAL(BN_bn2dec(accesskey->pub_key), "0");
72         CU_ASSERT_FALSE(crypto_gq_generate_userid(accesskey->pub_key,
73                 accesskey->user_id));
74
75         CU_ASSERT_FALSE(crypto_gq_generate_userid(accesskey->pub_key, test5));
76         CU_ASSERT_STRING_EQUAL(test5, accesskey->user_id);
77
78         CU_ASSERT_FALSE(crypto_gq_generate_userid(accesskey->priv_key, test6));
79         CU_ASSERT_STRING_NOT_EQUAL(test6, accesskey->user_id);
80
81 }
82
```

```
83 /** Test of crypto_gq_generate_signature(). Checks if a signature was produced. If
84  * this signature is valid is check at test_crypto_gq_check_signature().
85  */
86 void test_crypto_gq_generate_signature(void)
87 {
88         crypto_gq_generate_signature(accesskey, test4);
89         CU_ASSERT_NSTRING_NOT_EQUAL(test8, test4, 1);
90 }
91
92 /** Test of crypto_gq_check_signature(). Checks if correct produced
93  * signatures are accepted and if the signature is manipulated or the key
94  * is corrupted it is rejected.
95  */
96 void test_crypto_gq_check_signature(void)
97 {
98         userkey->pub_key = BN_dup(accesskey->pub_key);
99
100         CU_ASSERT_FALSE(crypto_gq_check_signature(userkey, accesskey->mod,
101                 test4));
102
103         CU_ASSERT_TRUE(crypto_gq_check_signature(userkey, accesskey->mod,
104                 test4+1));
105
106         BN_lshift1(userkey->pub_key, userkey->pub_key);
107
108         CU_ASSERT_TRUE(crypto_gq_check_signature(userkey, accesskey->mod, test4));
109 }
110
111 /** Test of crypto_gq_generate_testnumber(). Checks if a testnumber was produced
112  * and that the random number is within the range 0 < r < n.
113  */
114 void test_crypto_gq_generate_testnumber(void)
115 {
116         CU_ASSERT_STRING_NOT_EQUAL((crypto_gq_generate_testnumber(accesskey,
117                 data)), "0");
118         CU_ASSERT_EQUAL(BN_cmp(data->random, accesskey->mod), -1);
119         CU_ASSERT_EQUAL(BN_cmp(data->random, test1),1);
120         CU_ASSERT_STRING_NOT_EQUAL(BN_bn2dec(data->testnumber), "0");
121 }
122
123 /** Test of crypto_gq_calculate_proof(). Checks if a proof was produced. The
124  * correctness of the proof is checked at test_crypto_gq_validate_proof().
125  */
126 void test_crypto_gq_calculate_proof(void)
127 {
128         crypto_gq_calculate_proof(accesskey, data, test7);
129         CU_ASSERT_NSTRING_NOT_EQUAL(test8, test7, 1);
130
131 }
132 /** Test of crypto_gq_validate_proof(). Checks if correct produced
133  * proofs are accepted and if the proof is manipulated or the key
134  * is corrupted it is rejected.
135  */
136 void test_crypto_gq_validate_proof(void)
137 {
138         data->pub_user_key = BN_dup(accesskey->pub_key);
139         CU_ASSERT_FALSE(crypto_gq_validate_proof(test7, data, accesskey->mod));
140
141         CU_ASSERT_TRUE(crypto_gq_validate_proof(test7+1, data, accesskey->mod));
142
143         BN_lshift1(data->pub_user_key, data->pub_user_key);
144         CU_ASSERT_TRUE(crypto_gq_validate_proof(test7, data, accesskey->mod));
145 }
```

Erklärung gem. §27 Abs. 2 APO

Ich erkläre hiermit gemäß §27 Absatz 2 APO, dass ich die vorstehende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Bamberg, den 28.09.2006 _____