Universität Hamburg
Fachbereich Informatik

Implementation eines Anonymisierungsverfahrens für WWW-Zugriffe

Diplomarbeit

Ulf Möller
<3umoelle@informatik.uni-hamburg.de>

16. Juli 1999

Betreuer: Prof. Dr. K. Brunnstein
Zweitbetreuer: Prof. Dr. M. Kudlek

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ACI	Anonymous Connection Identifier
DES	Data Encryption Standard
ftp	File Transfer Protocol
HMAC	Hash-based MAC
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IV	Initial Value
MAC	Message Authentication Code
MDC	Modification Detection Code
OAEP	Optimal Asymmetric Encryption Padding
PGP	Pretty Good Privacy
PKCS	Public Key Cryptography Standard
PRNG	Pseudo-Random Number Generator
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WWW	World Wide Web

Kapitel 1

Einleitung

Seit 1993 hat sich das WWW zu einem der wichtigsten Internet-Dienste entwickelt. Mit eigenen Angeboten sind im World Wide Web mittlerweile nahezu alle Zeitungen und Zeitschriften vertreten; eine zunehmende Anzahl kommerzieller Datenbanken ist über das WWW zu erreichen. Hinzu kommen neue Multimedia-Dienste und ungezählte private Angebote.

Bei traditionellen Medien ist dem Verlag nur bekannt, wer die Abonnenten einer Publikation sind; über deren Lesegewohnheiten liegen keine Daten vor. Es ist möglich, Zeitschriften am Kiosk zu erwerben, ohne daß eine Datenspur entsteht. Auch über die Zuhörer bzw. Zuschauer von Radio- und Fernsehprogrammen fallen normalerweise keine personenbezogenen Daten an.

Im Internet ist dies anders. Wenn der WWW-Browser des Nutzers eine Verbindung zum Server des Anbieters aufbaut, um Informationen abzurufen, fallen dort personenbezogene Daten an, die von vielen WWW-Servern in Dateien protokolliert werden. Internet-Provider können die Daten über sämtliche Netz-Zugriffe ihrer Kunden speichern und auswerten und so umfangreiche Persönlichkeitsprofile gewinnen.

Das Interesse der Anbieter an Daten über ihre Nutzer wird schon an dem großen Aufwand deutlich, der für die Ermittlung von Einschaltquoten von Fersehsendern und für Leserbefragungen von Zeitschriften getrieben wird. Wenn Betreiber von WWW-Servern personenbezogene Daten ohne Einwilligung oder sogar ohne das Wissen der Nutzer sammeln und auswerten, wird deren Recht auf informationelle Selbstbestimmung verletzt. Die Rechte der Nutzer sind in Datenschutzgesetzen festgeschrieben; der Nutzer hat aber keine Möglichkeit zu überprüfen, ob der Anbieter diese Regelungen tatsächlich einhält. Bei der Vielzahl der Anbieter ist auch keine umfassende Kontrolle durch Datenschutzbehörden möglich. Zudem befinden sich viele Server in Staaten wie den USA, in denen die Auswertung personenbezogener Daten im allgemeinen zulässig ist.

Zuverlässig läßt sich der Mißbrauch personenbezogener Daten im Internet daher nur mit technischen Mitteln verhindern. Erfolgt der Zugriff des Nutzers auf WWW-Server anonym, fallen dort gar keine personenbezogenen Daten an. Anonymisierungsverfahren für WWW-Zugriffe sind daher ein wirkungsvolles Verfahren, um den Datenschutz in diesem neuen Medium durchzusetzen.

David Chaum hat 1981 mit dem Mix-Netz das erste kryptographische Anonymisierungsverfahren für E-Mail entwickelt. In der Folgezeit wurden Verfahren vorgeschlagen, die anonyme Zugriffe auf das WWW ermöglichen. Der derzeit meistgenutzte Anonymisierungsdienst ist ein von der Firma *The Anonymizer Inc.* betriebener Server. Eine größere Sicherheit bieten auf dem Mix-Netz beruhende Verfahren wie *Onion-Routing* und *Freedom*. Die Implementation eines derartigen Anonymisierungsverfahrens ist Gegenstand dieser Arbeit.

Kapitel 2

Grundlagen

2.1 WWW-Zugriffe

Bei Zugriffen auf das WWW ruft der Nutzer mit Hilfe eines als *Browser* bezeichneten Programms Daten von *Servern* verschiedener Anbieter ab. Als Adresse eines WWW-Dokuments dient ein eindeutiger Bezeichner, üblicherweise ein *Uniform Resource Locator (URL),* der das Übertragungsprotokoll, eine Internet-Adresse sowie einen Pfadnamen für das Dokument angibt.

WWW-Browser beherrschen eine Vielzahl von Übertragungsprotokollen und Dateiformaten bzw. Datentypen. Für die Übertragung von WWW-Seiten wird üblicherweise das *Hypertext Transfer Protocol (HTTP)* [Fielding 1997] auf einer TCP/IP-Verbindung verwendet. Der Browser baut eine Verbindung zum HTTP-Port des Servers auf und sendet eine Anfrage *(Request)* mit dem URL der gewünschten WWW-Seite. Der Server antwortet mit einer Nachricht, die das gewünschte Dokument enthält *(Response).* HTTP-Nachrichten bestehen aus einem *Message Header,* der Fehlermeldungen und Angaben zum Datentyp des übertragenen Dokuments enthalten kann, sowie einem *Message Body,* der das zu übertragende Dokument enthält. Dokumente in der Beschreibungssprache HTML [Berners-Lee 1995] enthalten neben dem eigentlichen Text Informationen darüber, wie der Text darzustellen ist, und Verknüpfungen zu weiteren Daten, die zum Teil vom Benutzer ausgewählt werden können, zum Teil aber auch – wie beispielsweise in einem HTML-Dokument enthaltene Grafiken – vom Browser automatisch abgerufen und dargestellt werden.

Statt direkte HTTP-Zugriffe zum jeweiligen Server durchzuführen, werden *Proxies* eingesetzt, wenn eine direkte Verbindung nicht möglich oder nachteilig ist. Der Browser sendet die Anfragen dabei an den voreingestellten Proxy, dieser leitet sie an den WWW-Server weiter. Proxy-Caches speichern oft abgerufene Seiten lokal, um schnellere Antwortzeiten zu erreichen und das Übertragungsvolumen zu reduzieren. Proxies werden auch eingesetzt, um Clients bei der Verwendung eines Firewall den Zugriff auf externe Daten zu ermöglichen, ohne ihnen direkte HTTP-Verbindungen zu externen Servern erlauben zu müssen.

2.2 Personenbezogene Daten bei WWW-Zugriffen

Bei WWW-Zugriffen entsteht eine Vielzahl von Daten, die Rückschlüsse über Interessen und Verhaltensweisen des Nutzers geben. Zu unterscheiden ist dabei, welche Daten ein Informationsanbieter und welche ein Netzbetreiber erhält.

2.2.1 Anbieter

Der Betreiber eines WWW-Servers erhält verschiedene Daten über die Personen, die sein Angebot nutzen. Diese Daten werden ohne Kontrollmöglichkeit und oft sogar ohne Wissen der Nutzer übermittelt.

2.2.1.1 IP-Adressen

Wird zwischen dem Browser des Nutzers und dem WWW-Server eine HTTP-Verbindung aufgebaut, erfährt der Anbieter notwendigerweise, von welcher IP-Adresse der Abruf durchgeführt wird. Arbeitsplatzrechner mit permanenter Internetverbindung haben in der Regel eine fest zugeordnete IP-Nummer, so daß der Nutzer eines Ein-Personen-Systems durch diese Angabe eindeutig identifiziert ist.

Bei der Modem-Einwahl, wie Privatkunden sie üblicherweise nutzen, wird die IP-Nummer dagegen in der Regel dynamisch vergeben. Aus der Adresse läßt sich dann nur entnehmen, über welchen Internet-Provider der Zugriff durchgeführt wurde.

Auch wenn ein Proxy verwendet wird, ist der Nutzer nicht an der IP-Nummer zu erkennen.[1] Da die Verbindung zum Server durch den Proxy hergestellt wird, ist aus der IP-Nummer nur zu erkennen, daß der Zugriff von einem Angehörigen derjenigen Organisation bzw. einem Kunden desjenigen Providers erfolgt ist, der den Proxy betreibt. Welche Auswirkungen dies hat, hängt von der Anzahl der Benutzer ab. Bei einer kleinen Zahl von Nutzern ist damit zu rechnen, daß nur einer von ihnen auf einen bestimmten Server zugreift, so daß die Adreß-Angabe auch in diesem Fall auf die Person zurückschließen läßt.

2.2.1.2 Identifikations-Dienste

Einige frühe HTTP-Implementationen sendeten bei Zugriffen auf WWW-Server eine `From`-Zeile mit der E-Mail-Adresse des Nutzers, ähnlich wie

[1] Manche Proxies fügen allerdings eine Header-Zeile `X-Forwarded-For:` ein, aus der die IP-Nummer des Nutzers hervorgeht.

es bei dem älteren ftp-Protokoll üblich ist, die E-Mail-Adresse als Paßwort anzugeben. In [Fielding 1997] ist die Übertragung dieser Header-Zeile aus Datenschutzgründen allerdings nur nach ausdrücklicher Bestätigung des Nutzers vorgesehen.

Es gibt jedoch weitere Internet-Dienste, die der Identifikation von Nutzern dienen. Über den `finger`-Dienst [Zimmerman 1991] lassen sich Namen, E-Mail-Adressen und weitere Informationen über die auf einem Computer eingeloggten Personen feststellen. Sofern der Betreiber des Systems diesen Dienst aktiviert hat, läßt sich so der Bezug zwischen einer IP-Nummer und einer Person herstellen. Nur wenn der Zugriff über einen Proxy erfolgt ist, führt eine `finger`-Anfrage nicht zu der gewünschten Information. Dann ist es jedoch möglich, Anfragen an alle Systeme in dem jeweiligen Subnetz zu richten, in dem sich der Proxy befindet. In vielen Fällen ist unter den Personen, deren Namen so festgestellt werden, auch der tatsächliche Nutzer. Wenn zu verschiedenen Zeiten Zugriffe erfolgen, läßt sich schließen, daß der fragliche Nutzer derjenige sein muß, der während aller Zugriffe eingeloggt war.

Das für Audit-Zwecke gedachte Identifikations-Protokoll [Johns 1993] übermittelt eine den Eigentümer einer bestimmten TCP-Verbindung identifizierende Zeichenkette. Wenn der `ident`-Dienst auf dem System des Nutzers installiert ist, kann ein Server-Betreiber so feststellen, wer einen Zugriff durchgeführt hat. Nur in wenigen Fällen ist die Kooperation des Systembetreibers erforderlich, um aus der `ident`-Zeichenkette auf die Identität des Nutzers zu schließen; üblicherweise wird der Login-Name des Nutzers übermittelt. Bei T-Online, dem größten Internet-Zugangsanbieter Deutschlands, ist dies in der Regel die Telefonnummer des Kunden. Auf Antrag teilt T-Online auch eine andere Nummer zu; da diese jedoch auch in allen E-Mail-Nachrichten und Usenet-Artikeln des Nutzers übertragen wird, ist es häufig möglich, dessen Namen beispielsweise über eine Suchmaschine festzustellen [Möller 1998].

2.2.1.3 Vom Browser übertragene Daten

Die verbreitetsten Browser *Internet Explorer* und *Netscape Navigator* übertragen als Bestandteil einer HTTP-Anfrage neben dem URL eine Reihe weiterer Daten. übermittelt werden beispielsweise Angaben zur verwendeten Browser- und Betriebssystemversion und der Bildschirmauflösung, eine Liste akzeptierter Datentypen und Angaben über die Sprache des Nutzers. Diese Informationen können dazu dienen, den Nutzer zu identifizieren. Da Proxies diese Angaben im Normalfall weiterleiten, ist die Identifikation des Nutzers auch möglich, wenn er einen Proxy verwendet.

Wenn der Nutzer einem Hypertext-Link folgt, übertragen Browser zudem einen `Referer`-Header, der den URL der Seite enthält, von der der Nutzer gekommen ist. Damit läßt sich verfolgen, wie er sich durch das

```
GET / HTTP/1.0
If-Modified-Since: Tuesday, 20-Oct-98 15:07:06 GMT; length=7000
Connection: Keep-Alive
User-Agent: Mozilla/4.07 [en] (X11; I; SunOS 5.6 sun4u)
Host: echo.znet.de:8888
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
        image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Abbildung 2.1: Beispiel für einen HTTP-Request

WWW-Angebot bewegt, mit welcher Such-Anfrage er auf einen Server gelangt ist oder welchen Namen eine private Datei hat, von der aus er den Server aufruft.

WWW-Browser übertragen auf Anforderung des Servers hin während einer Sitzung oder auch über mehrere Sitzungen hinweg in den HTTP-Anfragen Zustandsinformationen, die als *Cookies* bezeichnet werden [Kristol 1997]. Dadurch können beispielsweise personalisierte Einstiegsseiten implementiert werden. Häufig werden Cookies allerdings im Zusammenhang mit Werbeeinblendungen verwendet. Deren Betreiber erfährt im `Referer`-Header, welche Seite abgerufen wurde, und kann die Anfrage durch einen Cookie einem einzelnen Nutzer zuordnen. Da viele Suchmaschinen und Inhalteanbieter an solche Werbeanbieter angeschlossen sind, entstehen dort umfassende Nutzerprofile, in denen unter Umständen sämtliche Suchanfragen des Nutzers enthalten sind.

Netscape 4 bietet die Möglichkeit, zu WWW-Seiten themenverwandte Seiten anderer Anbieter zu finden. Dazu sendet der Browser einen HTTP-Request mit dem URL der jeweils abgerufenen Seite an einen Netscape-eigenen Server. In der Default-Einstellung wird diese Anfrage während des Ladens jeder WWW-Seite automatisch durchgeführt, nachdem der Nutzer den „What's related"-Button einmal benutzt hat. Die Firma Netscape erhält dadurch eine vollständige Liste aller von dem Nutzer gelesenen Seiten.

Weitere Möglichkeiten zur Übertragung persönlicher Daten des Nutzers ergeben sich daraus, daß WWW-Browser aus dem Netz geladene Programme ausführen können. ActiveX-Controls haben alle Zugriffsrechte des Nutzers auf System-Ressourcen. Bei anderen Konzepten, wie zum Beispiel Java-Applets und JavaScript, sind Zugriffsbeschränkungen vorgesehen; es wurden jedoch verschiedene Fehler bekannt, durch die nicht vorgesehene Zugriffe auf Daten des Nutzers möglich sind. Beispielsweise ist es in vielen Browser-Versionen möglich, mit JavaScript-Programmen ohne Wissen des Nutzers eine E-Mail-Nachricht in dessen Namen zu verschicken.

2.2.1.4 Server-Protokollierung

Der derzeit meistverwendete WWW-Server Apache protokolliert bei jedem Zugriff standardmäßig das aufgerufene Objekt, die Internet-Adresse des Abrufers sowie Datum und Uhrzeit. Auch die Protokollierung von `Referer`-Zeilen ist weit verbreitet.

2.2.2 Netzbetreiber

Der Netzbetreiber, über den der Nutzer Zugang zum Internet erhält, kann die gesamte Kommunikation des Nutzers beobachten. Für Proxies sind umfangreiche Protokolldateien nicht ungewöhnlich. Der Proxy *Squid* speichert in der Datei `access.log` beispielsweise zu jedem Zugriff den Zeitpunkt, die Adresse des Clients sowie auf Wunsch das Ergebnis einer `ident`-Abfrage, den abgerufenen URL, und die Größe der übertragenen Datei [Wierda 1997].

Auch wenn der Nutzer keinen Proxy nutzt, kann der Netzbetreiber auf der TCP-Ebene auf die Kommunikation zugreifen. Nur wenn die Übertragung verschlüsselt erfolgt (beispielsweise mittels SSL [Freier 1996]), hat er keinen Zugriff auf die Inhaltsdaten. Die Adressen der Kommunikationspartner und die Länge der übertragenen Daten liegen allerdings auch dann vor und können zur Identifikation der abgerufenen Dateien verwendet werden.

Eine Verbindung zu einem WWW-Server führt üblicherweise durch die Netze vieler private Betreiber; welche Betreiber in welchen Ländern beteiligt sind, ist dem Nutzer zumeist nicht bekannt. Die Übertragung kann an jedem beteiligten Netzknoten abgehört und automatisch nach interessanten Daten durchsucht werden.

2.3 Datenschutzanforderungen

Eine Anonymisierung des Zugriffs auf WWW-Server kann wirksamen Datenschutz für den Nutzer gewährleisten. Zumeist benötigt der Anbieter keine Informationen über seine individuellen Kunden: Statistiken über die Nutzung seines Angebots und sogar die Bezahlung kostenpflichtiger Dienste sind auch dann möglich, wenn der Zugriff anonym erfolgt. Es ist somit möglich, eine datenschutzfreundliche Infrastruktur aufzubauen, die auch die Interessen der Anbieter berücksichtigt. Ziel dieser Infrastruktur ist demzufolge, daß der Nutzer gegenüber dem Betreiber *anonym* bleibt.

Um zu verhindern, daß der Betreiber Wissen über eine Person ansammeln kann, das deren Anonymität gefährden würde, sollten verschiedene Zugriffe eines Nutzers auf den Server in der Regel *unverkettbar* sein. In bestimmten Fällen ist es jedoch wünschenswert, personalisierte Zugriffe zu ermöglichen, ohne daß der Betreiber dabei die Identität des Nutzers kennen sollte *(Pseudonymität)*. Die Pseudonyme eines Nutzers für verschiedene Rollen sollten dann unverkettbar sein.

Zudem sollten Zugriffe auf WWW-Server für Unbeteiligte *unbeobachtbar* erfolgen. Dies ist auch dann wünschenswert, wenn der Nutzer sich dem Betreiber des Servers gegenüber identifiziert. Unbeobachtbarkeit beinhaltet nicht nur die Vertraulichkeit der *Inhaltsdaten,* sondern auch die der Information, wer wann von wem Daten abruft *(Verkehrsdaten)*.

Ein datenschutzkonformes Kommunikationssystem soll somit die Anforderungen nach Anonymität, Unverkettbarkeit, Pseudonymität und Unbeobachtbarkeit erfüllen [Pfitzmann 1990, Rannenberg 1995].

2.4 Angreifermodelle

Um die Sicherheit eines Verfahrens zu bewerten, werden verschiedene Angreifermodelle betrachtet. Ein **Angreifer** ist jemand oder etwas – zum Beispiel eine Person oder ein Computer –, der versucht, nicht für ihn bestimmte Informationen zu erhalten, falsche Nachrichten zu senden oder die Erbringung eines Dienstes zu verhindern. Angreifer kann sowohl ein Außenstehender als auch ein Nutzer oder Betreiber eines Netzdienstes sein.

Ein **passiver** Angreifer kann einen Teil oder sämtliche Kommunikationsverbindungen abhören. Ein **aktiver** Angreifer kann darüberhinaus übertragene Nachrichten verfälschen, verzögern und unterdrücken und eigene Nachrichten übertragen, indem er versucht, als legitimer Sender zu erscheinen.

Ein Angreifer, der nur Zugriff auf die Kommunikationsverbindungen hat, heißt **externer** Angreifer. Sind legitime Kommunikationsteilnehmer, also Nutzer oder Betreiber eines Dienstes, an einem Angriff beteiligt, spricht man von einem **internen** Angreifer.

Häufig ist es möglich, durch die Auswertung der übertragenen Datenmengen und der Sende- und Empfangszeitpunkte Informationen über kryptographisch gesicherte Kommunikation zu gewinnen. Solche Angriffe werden als **Verkehrsanalyse** *(traffic analysis)* bezeichnet.

Wenn der Angreifer sämtliche Kommunikationsverbindungen abhören kann, heißt er **omnipräsenter** Angreifer. Häufig ist die Annahme sinnvoll, daß bestimmte Verbindungen dem Angreifer nicht zugänglich sind. Auch bei omnipräsenten Angreifern ist vorausgesetzt, daß ein Teil der geheimen Schlüssel dem Angreifer nicht bekannt ist; andernfalls wäre kein Schutz durch kryptographische Verfahren möglich.

Weiter werden Angreifer danach unterschieden, welche Berechnungen sie durchführen können, um beispielsweise eine Verschlüsselung zu brechen.

2.5 Definitionen

R_I bezeichne, daß der Nutzer I eine Rolle R wahrnimmt, beispielsweise Sender oder Empfänger einer Nachricht ist. $\mathrm{P}(A)$ ist die Wahrscheinlichkeit eines Ereignisses A, $\mathrm{P}(A|B)$ die bedingte Wahrscheinlichkeit für A unter der Bedingung, daß das Ereignis B eingetreten ist.

Ein Nutzer A ist in der Rolle R **anonym** bezüglich eines Angreifers E und einer Menge $\mathcal{A}$ von nicht mit E kooperierenden Nutzern *(Anonymitätsmenge)*, wenn E nicht feststellen kann, daß A die Rolle R wahrgenommen hat.

A heißt **perfekt anonym** in der Rolle R bezüglich E und $\mathcal{A}$, wenn nach jeder für E möglichen Beobachtung B gilt: $\forall I \in \mathcal{A}\colon\ \mathrm{P}(R_I) = \mathrm{P}(R_I|B)$, d. h. keine Beobachtung B liefert E zusätzliche Informationen darüber, wer die Rolle R wahrnimmt [Pfitzmann 1990].

Je nach der dabei betrachteten Rolle spricht man von *Sender-Anonymität* und *Empfänger-Anonymität*. Um einen WWW-Zugriff durchzuführen, muß eine Anfrage gesendet und die Antwort des Servers empfangen werden; damit der Zugriff anonym bleibt, sind also Sender- und Empfänger-Anonymität notwendig.

Ein Ereignis X heißt für einen Angreifer E **unbeobachtbar**, wenn für jede für E mögliche Beobachtung B gilt: $\mathrm{P}(X) = \mathrm{P}(X|B)$, d. h. daß die Beobachtung E keine zusätzlichen Informationen über X liefert.

Kapitel 3

Protokolle zur Anonymisierung

Kryptographische Verfahren dienen dem Schutz der Vertraulichkeit und Authentizität von Daten. Durch Protokolle, in denen kryptographische Operationen verwendet werden, kann die anonyme Übertragung von Nachrichten realisiert werden.

3.1 Kryptographische Verfahren

3.1.1 Vertraulichkeit

Zum Schutz vor unbefugter Kenntnisnahme können Daten verschlüsselt werden. Ein Kryptosystem besteht aus einem Klartextraum $\mathcal{M}$, einem Geheimtextraum $\mathcal{C}$, einem Schlüsselraum $\mathcal{K}$, einer Menge $\{E_e\colon\ e \in \mathcal{K}\}$ von Verschlüsselungsfunktionen $E_e\colon\ \mathcal{M} \to \mathcal{C}$ (bei *probabilistischen* Kryptosystemen ist E_e statt dessen ein nichtdeterministischer Algorithmus) und einer korrespondierenden Menge $\{D_d\colon\ d \in \mathcal{K}\}$ von Entschlüsselungsfunktionen $D_d\colon\ \mathcal{C} \to \mathcal{M}$ mit der Eigenschaft, daß es zu jedem $e \in \mathcal{K}$ einen Schlüssel d gibt, so daß $D_d(E_e(m)) = m$ für alle $m \in \mathcal{M}$ [Menezes 1997]. $\mathcal{M}, \mathcal{C}, \mathcal{K}, \{E_e\}$ und $\{D_d\}$ müssen nicht geheimgehalten werden.

Ein Verschlüsselungsverfahren ist **sicher,** wenn es einem auf effiziente Berechnungen beschränkten Angreifer nicht möglich ist, ohne Kenntnis des Schlüssels d aus einem Geheimtext systematisch Informationen über den entsprechenden Klartext – über die Tatsache hinaus, daß eine Nachricht der gegebenen Länge vorliegt – zurückzugewinnen.

Informationstheoretisch oder **perfekt sicher** ist ein Verschlüsselungsverfahren, wenn ein Angreifer mit unbegrenzter Rechenkapazität von der Länge abgesehen keine Informationen über den Klartext erhalten kann. Perfekte Sicherheit ist nur möglich, wenn der Schlüssel mindestens so lang ist wie die (eventuell vorher komprimierte) zu übertragende Nachricht. Durch das daraus resultierende Schlüsselverteilungsproblem sind perfekt sichere Verfahren für die Anwendung in offenen Netzen unpraktikabel. Die Bewertung von in der Praxis eingesetzen Verfahren als sicher beruht auf der Annahme, daß für bestimmte Probleme keine effizienten Algorithmen existieren und daß bestimmte Berechnungen in der einem Angreifer zur Verfügung stehenden Zeit nicht praktikabel sind. Wenn gezeigt werden kann, daß das Brechen eines Systems so schwer ist, wie die Lösung eines wohlbekannten und als schwierig angesehenen Problems zu finden, wird es auch als **kryptographisch sicher** bezeichnet.

Angriffe auf Verschlüsselungsverfahren werden danach unterschieden, ob der Angreifer nur Geheimtexte beobachtet *(ciphertext-only attack),* Paare von Klartexten und zugehörigen Geheimtexten kennt *(known-plaintext attack),* Klartexte selbst auswählen kann und die zugehörigen Geheimtexte erhält *(chosen-plaintext attack)* oder Geheimtexte auswählen kann und die entsprechenden Klartexte erhält *(chosen-ciphertext attack).*

3.1.1.1 Symmetrische Verschlüsselung

In einem **symmetrischen** Verschlüsselungsverfahren kann d einfach aus e bestimmt werden, üblicherweise ist $e = d$. Dieser **geheime Schlüssel** muß vorab über einen sicheren Kanal ausgetauscht werden.

Ein symmetrisches Verschlüsselungsverfahren, durch das ein n-Bit-Klartext auf einen n-Bit-Geheimtext abgebildet wird, heißt **Blockchiffre** mit Blocklänge n. Um Klartexte beliebiger Länge zu verschlüsseln, können Blockchiffren in verschiedenen **Betriebsmodi** eingesetzt werden [Menezes 1997]. Im *ECB-Modus (Electronic Code Book)* wird der Klartext in Blöcke der Größe n unterteilt und alle Blöcke unabhängig voneinander verschlüsselt. Identische Klartextblöcke ergeben daher identische Geheimtextblöcke, so daß Datenmuster im Geheimtext erkennbar bleiben und Angreifer Blöcke unbemerkt vertauschen oder einfügen können. Für Nachrichten, die länger als n Bits sind, ist dieser Modus somit unsicher. Im *CBC-Modus (Cipher Block Chaining)* wird ein Initialisierungswert *(initial value, IV)* verwendet, von dem der erste Geheimtextblock neben dem Schlüssel und dem ersten Klartextblock zusätzlich abhängt; alle folgenden Geheimtextblöcke hängen vom jeweils vorhergehenden Geheimtextblock, dem Schlüssel und dem jeweiligen Klartextblock ab. Damit ergeben sich für die Verschlüsselung eines Klartextes mit demselben Schlüssel genau dann identische Geheimtexte, wenn auch der IV identisch ist. Der IV muß im Gegensatz zum Schlüssel nicht geheimgehalten werden.

Das bekannteste symmetrische Verschlüsselungsverfahren ist der **Data Encryption Standard (DES)** [FIPS 1980]. DES ist eine Blockchiffre mit 56-Bit-Schlüsseln. Aufgrund des kleinen Schlüsselraums ist es mit begrenzten Ressourcen möglich, den Klartext zurückzugewinnen, indem der Geheimtext systematisch mit allen denkbaren Schlüsseln entschlüsselt wird; DES ist daher unsicher [EFF 1998]. Durch die Verwendung von **Triple-DES** [Menezes 1997] kann dieser Angriff verhindert werden. Praktikable Angriffe auf Triple-DES sind nicht bekannt; dieser Algorithmus wird daher allgemein als sicher angesehen. Ein *Advanced Encryption Standard (AES)* wird derzeit entwickelt.

3.1.1.2 Asymmetrische Verschlüsselung

Ein Kryptosystem, das auch dann sicher ist, wenn der Angreifer E_e kennt, heißt **asymmetrisch** *(Public-Key-Verschlüsselung)*. e kann in dem Fall veröffentlicht werden und heißt daher auch **öffentlicher Schlüssel**; d ist der **private Schlüssel.**

Asymmetrische Kryptosysteme beruhen auf *Einwegfunktionen mit Falltür*. Eine **Einwegfunktion** ist eine effizient berechenbare Funktion, deren Umkehrung sich nur mit vernachlässigbarer Wahrscheinlichkeit effizient berechnen läßt; bei einer Einwegfunktion mit Falltür ist die Umkehrung nur mit Kenntnis eines geheimen Parameters *(Falltürinformation)* effizient zu berechnen. Ob es derartige Funktionen wirklich gibt, ist nicht bekannt. Funktionen, von denen angenommen wird, daß sie diese Eigenschaft erfüllen, beruhen beispielsweise auf der Schwierigkeit, Zahlen in ihre Primfaktoren zu zerlegen oder diskrete Logarithmen zu berechnen.

Damit ein asymmetrischer Verschlüsselungsalgorithmus sicher ist, muß er einen von verschiedenen möglichen Geheimtexten zufällig auswählen *(probabilistische Verschlüsselung),* wobei die Menge der möglichen Geheimtexte so groß ist, daß die Zuordnung eines Geheimtextes zu einem gegebenen Klartext nicht durch systematisches Ausprobieren aller Verschlüsselungen dieses Klartextes festgestellt werden kann. Bei *deterministischen* asymmetrischen Kryptosystemen ist die Entschlüsselung von Nachrichten aus einem kleinen, bekannten Klartextraum trivial [Goldreich 1997, Bellare 1997].

Ein Kryptosystem heißt *plaintext aware,* wenn es ohne Kenntnis des Klartextes nicht praktikabel ist, einen gültigen Geheimtext zu erzeugen. Bei Kryptosystemen ohne *plaintext awareness* besteht die Möglichkeit, daß ein Angreifer einen Geheimtext durch adaptive *chosen ciphertext*-Angriffe entschlüsseln kann. Bei *chosen ciphertext*-Angriffen erzeugt der Angreifer Geheimtexte, deren Klartext in einer vorhersagbaren Weise vom gesuchten Klartext abhängt, und veranlaßt den Schlüsselinhaber dazu, ihm Informationen über den Klartext mitzuteilen. Da die Entschlüsselung in vielen Anwendungen automatisch erfolgt, sind derartige Angriffe oft praktikabel. Dies ist zum Beispiel bei RSA-Verschlüsselung nach PKCS #1 Version 1.5 der Fall [Bleichenbacher 1998].

Das verbreitetste asymmetrische Kryptosystem ist **RSA** [Kaliski 1998]. Ein öffentlicher RSA-Schlüssel besteht aus zwei natürlichen Zahlen (n, e), wobei n das Produkt zweier verschiedener Primzahlen p und q ist und $\mathrm{ggT}\big(e, (p-1)(q-1)\big) = 1$ gilt. Zu diesem Schlüssel gehört ein geheimer Schlüssel d mit $ed \equiv 1 \mod (p-1)(q-1)$. d kann bei Kenntnis von p und q effizient berechnet werden.

Um eine Nachricht zu verschlüsseln, wird anhand einer *Kodierungsvorschrift* eine den Klartext repräsentierende natürliche Zahl m gebildet und der Geheimtext durch die Verschlüsselungsprimitive $c := m^e \mod n$ berechnet. Mit dem geheimen Schlüssel kann $m = c^d \mod n$ berechnet werden. Durch eine Dekodierungsvorschrift wird daraus der Klartext gewonnen oder aber ein Entschlüsselungsfehler festgestellt.

Wenn es möglich ist, große Zahlen zu faktorisieren, es also einen effizienten Algorithmus gibt, um p und q aus dem öffentlichen Modulus n zu errechnen, ist RSA unsicher. Der beste hierfür bekannte Algorithmus ist das *General Number Field Sieve (GNFS)*. Die Faktorisierung eines 1024 Bit großen RSA-Modulus würde damit $3 \cdot 10^{11}$ MIPS-Jahre benötigen, so daß RSA bei dieser Schlüssellänge als sicher angesehen wird [Wiener 1998].

Wenn RSA mit geeigneten Kodierungsvorschriften verwendet wird und bei der Schlüsselerzeugung bestimmte Randbedingungen [Menezes 1997] beachtet werden, ist kein effizienterer Angriff als die Faktorisierung von n bekannt. Um ein sicheres Kryptosystem zu erhalten, muß die Kodierungsvorschrift probabilistisch sein. Die Dekodierungsvorschrift muß zur Verhinderung von *chosen plaintext*-Angriffen sicherstellen, daß ein gültiger Geheimtext nicht erzeugt werden kann, ohne den dazugehörigen Klartext zu kennen.

OAEP *(Optimal Asymmetric Encryption Padding)* [Bellare 1995, Kaliski 1998] ist eine probabilistische Kodierungsvorschrift, für die Sicherheit von RSA unter der Annahme bewiesen werden kann, daß die verwendete Hashfunktionen ideal ist (*random oracle*-Modell) und daß die Verschlüsselungsprimitive eine Einwegfunktion ist.

Das **ElGamal**-Kryptosystem beruht dagegen auf der Schwierigkeit, diskrete Logarithmen zu berechnen. Auch hierfür werden Schlüssellängen ab 1024 Bit als sicher angesehen; für auf elliptischen Kurven basierende Verfahren gelten Schlüssellängen ab 160 Bit als sicher [Wiener 1998].

Der Aufwand für asymmetrische Ver- und Entschlüsselung ist i. a. höher als der für symmetrische. Asymmetrische Verfahren werden in der Praxis mit symmetrischen zu *Hybridverfahren* kombiniert, indem nur ein zufällig gewählter Sitzungsschlüssel für das symmetrische Verfahren asymmetrisch verschlüsselt wird.

Eine Anforderung, die man an Kryptosysteme zusätzlich stellen kann, besteht darin, daß ein Angreifer im Nachhinein selbst dann keine Informationen über in der Vergangenheit übertragene Nachrichten gewinnen darf, wenn einer der dauerhaften privaten Schlüssel zu einem späteren Zeitpunkt kompromittiert wird. Diese als **forward secrecy** bezeichnete Eigenschaft kann durch ein Protokoll zum Vereinbaren geheimer Sitzungsschlüssel, beispielsweise durch den Diffie-Hellman-Algorithmus, erreicht werden [Goldwasser 1997]. Sie läßt sich auch bei RSA erzielen, indem Schlüssel nur eine kurze Gültigkeitsdauer erhalten und die geheimen Schlüssel nach deren Ablauf gelöscht werden.

3.1.2 Integrität und Authentizität

Unter *Integrität* versteht man, daß Informationen vor unbefugter Veränderung geschützt sind; *Authentizität* ist die Gewährleistung, daß Daten tatsächlich vom angegebenen Absender stammen.

Zum Erreichen dieser Ziele werden **kryptographische Hashfunktionen** eingesetzt. Eine *Hashfunktion* ist eine effizient berechenbare Funktion, die Zeichenketten beliebiger Länge auf Zeichenketten einer festen Länge („Hash-Werte") abbildet. Eine Hashfunktion h heißt *sicher* oder *kryptographische Hashfunktion,* wenn es nicht praktikabel ist, zwei Eingaben x und y zu finden, so daß $h(x) = h(y)$ gilt *(Kollisionsresistenz)*, oder zu einem gegebenen Hash-Wert y ein x mit $h(x) = y$ zu finden. Verbreitete Hash-Algorithmen sind MD5, in dem jedoch Schwächen gefunden wurden, und SHA-1 [FIPS 1995]. Die Hash-Werte von SHA-1 sind 160 Bit groß.

Hash-Werte können als **Modification Detection Codes (MDC)** die Integrität verschlüsselter Daten sicherstellen. Hierfür berechnet der Sender einer Nachricht x den Hash-Wert $h(x)$ und überträgt $x \,||\, h(x)$ verschlüsselt. Hierbei bezeichne $||$ die Konkatenation von Bitketten. Der Empfänger entschlüsselt die empfangene Nachricht und berechnet ebenfalls $h(x)$. Wenn die Werte übereinstimmen, ist die Nachricht nicht verfälscht worden.

Zur Sicherung der Authentizität von Daten können **Message Authentication Codes (MAC)** eingesetzt werden. Der Sender einer Nachricht x berechnet den MAC $h_k(x)$ für einen ihm und dem Empfänger bekannten geheimen Schlüssel k und überträgt $x \,||\, h_k(x)$. Der Empfänger berechnet ebenfalls $h_k(x)$. Wenn die Werte übereinstimmen, ist die Nachricht authentisch, da ein Angreifer ohne Kenntnis von k keinen gültigen MAC bilden kann. HMAC [Krawczyk 1997] ist eine Möglichkeit, MACs aus Hashfunktionen zu konstruieren.

Auch **digitale Signaturen** dienen der Gewährleistung der Authentizität. Zusätzlich ermöglichen sie es dem Empfänger, gegenüber einem Dritten zu beweisen[1], daß der Sender die Nachricht signiert hat. Der Sender kann mit seinem privaten Schlüssel d und der Signierfunktion $S_d\colon\ \mathcal{M} \to \mathcal{C}$ (oder einem nichtdeterministischen Signieralgorithmus S_d) Signaturen für seine Nachrichten generieren. Mit dem zu d gehörigen öffentlichen Schlüssel e und der Verifikationsfunktion $V_e\colon\ \mathcal{M} \times \mathcal{C} \to \{\mathsf{wahr}, \mathsf{falsch}\}$ kann der Empfänger die Authentizität der signierten Nachricht prüfen.

Digitale Signaturen werden auch zur Zertifizierung öffentlicher Schlüssel eingesetzt, die dann auch über unsichere Kanäle ausgetauscht werden können. Der Anwender benötigt nur den öffentlichen Schlüssel einer vertrauenswürdigen Instanz, die mit ihrer Signatur die Zuordnung anderer öffentlicher Schlüssel zu Personen bestätigt.

[1] Die Geheimhaltung der privaten Schlüssel ist dabei vorausgesetzt. Um zu verhindern, daß der Schlüsselinhaber seinen geheimen Schlüssel verbreiten kann, wurden Chipkarten-basierte Verfahren vorgeschlagen.

3.2 Das ideale Modell der Anonymisierung

Im idealen Modell der anonymen Kommunikation existiert eine vertrauenswürdige Instanz, die Nachrichten anonymisiert weiterleitet. Jeder Teilnehmer verfügt über einen sicheren Kanal zur vertrauenswürdigen Instanz. Diese gibt die Nachrichten, die sie von den Teilnehmern erhält, anonymisiert weiter *(Sender-Anonymität)* und stellt Nachrichten an anonyme Adressen dem Empfänger zu *(Empfänger-Anonymität)*. Da die Instanz vertrauenswürdig ist und es Angreifern weder möglich ist, die sicheren Kanäle abzuhören, noch sich über einen sicheren Kanal als die vertrauenswürdige Instanz auszugeben, ist das Senden und Empfangen von Nachrichten sowie die Beziehung von Sender und Empfänger unbeobachtbar. Somit erreicht dieses Verfahren perfekte Unbeobachtbarkeit und Anonymität. Alle ehrlichen (nicht mit dem Angreifer kooperierenden) Teilnehmer bilden die Anonymitätsmenge; über den Absender einer Nachricht ist nur bekannt, daß er dieser Menge angehört.

Der Angreifer kennt im idealen Modell die Nachrichten, die die mit ihm kooperierenden Instanzen gesendet haben. Nachrichten, die an eine mit dem Angreifer kooperierende Instanz gesendet oder an alle Teilnehmer verteilt werden, kann er beobachten und ggf. beeinflussen. Beobachtet der Angreifer eine ihm nicht bekannte Nachricht, muß somit einer der ehrlichen Teilnehmer der Absender sein. Jede dem Angreifer nicht bekannte anonyme Adresse muß einem der ehrlichen Teilnehmer gehören. Es ist nicht geheim, wer an dem Verfahren teilnimmt. Je weniger ehrliche Teilnehmer es gibt, desto größer ist die Wahrscheinlichkeit, daß der Angreifer eine anonyme Nachricht korrekt ihrem Absender zuordnet.

Ein reales Verfahren kann keine besseren Ergebnisse erzielen als das ideale [Micali 1992]; die Informationen, die der Angreifer im idealen Modell kennt, stehen ihm auch bei jedem realen Verfahren zur Verfügung.

Ein reales Anonymisierungsverfahren ist sicher, wenn es das ideale Modell mit kryptographischen Methoden so simuliert, daß ein Angreifer keine zusätzlichen Informationen gewinnen kann, obwohl keine physikalisch gesicherten Kanäle zur Verfügung stehen und keine einzelne Instanz als vertrauenswürdig feststeht [Beaver 1992, Goldreich 1997].

Ein sicherer Kanal kann simuliert werden, indem die Kommunikation zwischen zwei Instanzen so verschlüsselt wird, daß der Angreifer aus den abgehörten Nachrichten keine Informationen gewinnen kann. Wenn bei der Verschlüsselung die Länge der Nachrichten erkennbar bleibt, müssen zusätzlich bedeutungslose Nachrichten übertragen werden, um dem Angreifer keine Informationen darüber zu geben, ob Kommunikation stattfindet.

Eine vertrauenswürdige Instanz kann simuliert werden, indem mehrere Instanzen zusammenarbeiten, von denen zumindest ein Teil in Bezug auf eine bestimmte Funktion vertrauenswürdig ist. Den Teilnehmern muß dabei nicht bekannt sein, welche Instanzen tatsächlich die ehrlichen sind.

In realen Verfahren muß der Sender alle Nachrichten zusätzlich zu der in Anonymisierungsverfahren eingesetzten Verschlüsselung mit einem Schlüssel des Empfängers verschlüsseln, um die Vertraulichkeit der Inhaltsdaten sicherzustellen (Ende-zu-Ende-Verschlüsselung).

3.3 Das Mix-Netz

Ein **Mix** ist ein Netzknoten, der Nachrichten weiterleitet, wobei er den Zusammenhang zwischen seinen Ein- und Ausgaben durch Umkodieren verbirgt [Chaum 1981]. Wenn eine Nachricht eine *Sequenz* von unabhängig betriebenen Mixen durchläuft, reicht es aus, daß einer der Mixe vertrauenswürdig ist, um Anonymität und Unbeobachtbarkeit der Kommunikationsbeziehung zu gewährleisten, auch wenn der Angreifer mit allen anderen Mixen kooperiert und sämtliche Kommunikationsverbindungen abhören kann. Realisiert werden Mixe durch den Einsatz asymmetrischer Verschlüsselung.

Hierdurch ergibt sich ein effizientes, gegen mächtige Angreifer sicheres Anonymisierungsverfahren. In seiner ursprünglichen Form ist das Mix-Netz insbesondere zur Übertragung von E-Mail geeignet; derartige Mixe werden auch als **Remailer** bezeichnet. Für interaktive Anwendungen muß das Verfahren modifiziert werden.

3.3.1 Anonymisierung von Nachrichten

Zu jedem Mix gehört eine Adresse M_j, ein öffentlicher Schlüssel e_j und ein privater Schlüssel d_j. Die Adressen und öffentlichen Schlüssel aller Mixe sind allen potentiellen Sendern bekannt. Will Sender A dem Empfänger B eine Nachricht m zukommen lassen, wählt er einen beliebigen Pfad $\mathsf{M}_1, \mathsf{M}_2, \mathsf{M}_3, \ldots \mathsf{M}_\lambda$ durch das Mix-Netz, wobei ein Mix auch mehrfach vorkommen kann, und bildet die Nachricht N_1:
$N_\lambda = E_{e_\lambda}(\mathsf{B} \,||\, m)$
$N_i = E_{e_i}(\mathsf{M}_{i+1} \,||\, N_{i+1}) \quad \text{für } i = \lambda - 1, \ldots, 1$
A sendet also $N_1 = E_{e_1}(\mathsf{M}_2 \,||\, E_{e_2}(\mathsf{M}_3 \,||\, E_{e_3}(\ldots(E_{e_\lambda}(\mathsf{B} \,||\, m))\ldots)))$ an M_1.

E ist dabei ein gegen aktive Angriffe sicherer probabilistischer Verschlüsselungsalgorithmus; $||$ bezeichnet die Konkatenation von Bitketten. Da Mixe beliebige Nachrichten automatisch entschlüsseln und das Ergebnis weiterleiten, ist Sicherheit gegen *chosen-ciphertext*-Angriffe erforderlich.

Mix M_j sammelt die empfangenen Nachrichten, bis hinreichend viele Nachrichten vorliegen. Für jede Nachricht berechnet er dann $D_{d_j}(N_j) = (\mathsf{M}_{j+1} \,||\, N_{j+1})$ und sendet die resultierenden Nachrichten in veränderter Reihenfolge an den jeweils angegebenen Mix weiter. Der letzte Mix der Sequenz erhält $D_{d_j}(N_j) = (\mathsf{B} \,||\, m)$ und sendet die Nachricht m an B.

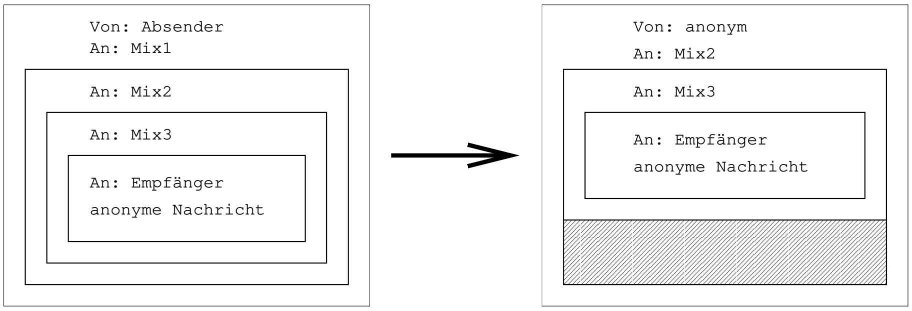

Abbildung 3.1: Anwendung einer Mix-Sequenz

3.3.2 Zuordbarkeit der Nachrichten

Für einen nicht mit M_j kooperierenden passiven Angreifer E sind die Nachrichten durch ihre *Länge,* durch die verschlüsselten – ohne Kenntnis der geheimen Schlüssel wie zufällig wirkenden – *Daten* und ihren *Ein-* und *Ausgangszeitpunkt* im Mix charakterisiert. Um Sicherheit gegen diesen Angreifertyp zu erreichen, muß verhindert werden, daß die Ein- und Ausgaben eines Mixes einander anhand dieser Merkmale zugeordnet werden können.

3.3.2.1 Umkodierung

Könnte der Angreifer eine Ausgabe mit größerer Wahrscheinlichkeit als durch zufälliges Auswählen einer Eingabe zuordnen, wären die Annahmen über das zugrundegelegte Kryptosystem verletzt [Rackoff 1993]. Einem Geheimtext, also der Eingabe eines Mixes sind ohne Kenntnis des Schlüssels keinerlei Informationen über den zugehörigen Klartext – die Ausgabe des Mixes – entnehmen, wenn das Kryptosystem sicher ist.

Umgekehrt ist es nicht möglich, den zu einem Klartext gehörenden Geheimtext zu identifizieren, weil die Verschlüsselung probabilistisch ist. Wenn allerdings ein Angreifer eine Nachricht aufzeichnet und mehrfach an den Mix sendet, ist die Voraussetzung, daß der Geheimtext zufällig ausgewählt wird, nicht erfüllt. Das wiederholte Senden identischer Eingaben *(Replay-Angriff)* an einen Mix führt zu identischen Ausgaben; damit kann ein aktiver Angreifer trivial feststellen, welcher Klartext zu einem beobachteten Geheimtext gehört. Um dies zu verhindern, muß der Mix Wiederholungen ignorieren.

3.3.2.2 Wiederholungen

Zum Erkennen von Wiederholungen kann der Mix ein eindeutiges oder jedenfalls nur mit vernachlässigbarer Wahrscheinlichkeit mehrfach auftretendes Merkmal jeder bearbeiteten Nachricht in einer Datenbank speichern. Das Prüfen auf Wiederholung entspricht dann einem Datenbankzugriff. Da alle während der Gültigkeitsdauer eines Schlüsselpaares weitergeleiteten Nachrichten in der Datenbank vermerkt werden müssen, ist der Speicherplatzbedarf dieses Verfahrens groß. Wenn Nachrichten jeweils innerhalb eines bestimmten Zeitraums eintreffen, kann dieser Aufwand reduziert werden, indem der Absender den mit E_{e_j} verschlüsselten Teil der Nachricht mit einem Zeitstempel versieht. Der Mix akzeptiert dann nur Nachrichten mit gültigem Zeitstempel nach Abgleich mit der Datenbank [Gülcü 1996]. Da alte Nachrichten aufgrund ihres Zeitstempels ignoriert werden, können die entsprechenden Einträge in der Datenbank nach Ablauf der zulässigen Zeitspanne gelöscht werden.

Wenn der Zeitstempel den exakten Zeitpunkt der Nachrichtenerzeugung angibt, kann dies Rückschlüsse auf den Absender erlauben, die ansonsten aufgrund der Verzögerung durch die vorangegangenen Mixe nicht möglich wären. Um internen Angreifern hierdurch keine zusätzlichen Informationen zukommen zu lassen, sollte der Zeitstempel keine zu feine Auflösung haben. Um Rückschlüsse auf den tatsächlichen Erzeugungszeitpunkt zu verhindern, kann der Absender den Zeitstempel zudem um einen zufällig gewählten Wert zurückdatieren. Wenn die Zeitspanne, nach der die Datenbankeinträge gelöscht werden, größer ist als die tatsächliche Verzögerung der Nachrichten, hat die Rückdatierung keinen Einfluß auf die Gültigkeitsprüfung.

Als Merkmal zur Wiederholungsprüfung kann ein Hash-Wert der gesamten Nachricht verwendet werden. Auch wenn der Absender aus einem hinreichend großen Merkmalsraum eine *Message-ID* zufällig wählt, treten Kollisionen nur mit geringer Wahrscheinlichkeit auf.

3.3.2.3 Nachrichtenlänge

Bei der Entschlüsselung entfernt der Mix den Sitzungsschlüssel sowie die Adreßinformation aus dem Datenpaket. Die Nachricht wird dadurch kürzer. Sofern nicht für alle Nachrichten ein vorgegebener Pfad *(Kaskade)* verwendet wird, muß die Umkodierung *längentreu* erfolgen, um Angreifern keine Information über den Weg der Nachricht zu geben [Pfitzmann 1990]. Dazu ergänzt der Mix die Nachricht bis zur ursprünglichen Länge mit zufälligen Bits *(padding)*. Da weder ein passiver Angreifer noch die folgenden Mixe – bei unverschlüsseltem m mit Ausnahme des letzten oder der mit ihm kooperierenden Mixe – die zur Entschlüsselung nötigen Schlüssel kennen, können sie das Padding nicht von verschlüsselten Nutzdaten unterscheiden. Um zu verhindern, daß die Padding-Länge nach dem Durchlaufen der Mix-Sequenz Informationen über die Länge des Pfades enthält, kann eine maximale Pfadlänge für das Gesamtsystem festgelegt werden. Die reservierten und nicht genutzten Einträge füllt dabei schon der Sender mit zufälligen Bits auf.

Durch diese Umkodierung werden die Nachrichten somit nach ihrer Länge in Klassen eingeteilt, innerhalb derer sie anonym und unbeobacht-

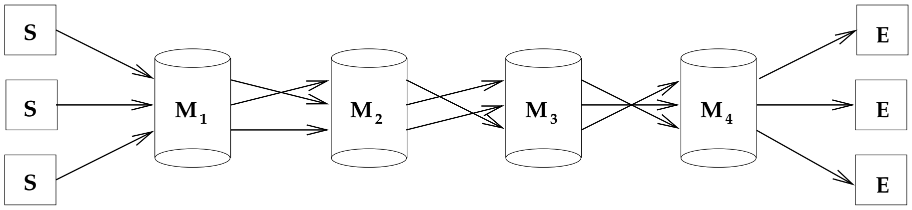

Abbildung 3.2: Schutz der Kommunikationsbeziehung durch eine Mix-Kaskade

bar sind. Um eine von der Nachrichtenlänge unabhängige Anonymitätsmenge zu erhalten, kann für alle Nachrichten eine einheitliche Länge festgelegt werden. Längere Nachrichten muß der Absender dann in mehrere Teile aufteilen; alle Nachrichten werden bis zur festgesetzten Länge mit zufälligen Bits aufgefüllt.

3.3.2.4 Umsortierung

Um die Unsicherheit über den Zusammenhang zwischen Ein- und Ausgabe zu erhöhen, müssen genügend viele Nachrichten gesammelt werden. Die Reihenfolge des Aussendens darf keine Informationen über die Reihenfolge des Empfangens der Nachrichten preisgeben. Für das Sammeln und Umsortieren der Nachrichten sind verschiedene Strategien möglich.

Im *Batch-Betrieb* sammelt ein Mix mindestens N Nachrichten an, die dann in beispielsweise lexikographischer Reihenfolge in einem Schub ausgegeben werden [Chaum 1981]. Im *Intervall-Batch-Betrieb* [Gülcü 1996] wird jeweils nach Ablauf eines festgelegten Zeitraums T ein Nachrichtenschub ausgegeben. Falls während des Intervalls nur $n < N$ Nachrichten eingetroffen sind, erzeugt der Mix $N - n$ bedeutungslose Nachrichten *(Dummies)*. Damit ist T die maximale und $T/2$ die durchschnittliche Verzögerungszeit für eine Nachricht.

Für nicht interaktive Anwendungen sind weitere Strategien möglich. Bei *Stop-and-Go-Mixen* [Kesdogan 1998] legt der Absender die Verzögerung der Nachricht gemäß einer Exponentialverteilung zufällig fest. Um zu verhindern, daß durch zu frühes oder zu spätes Weiterleiten der vorhergehenden Mixe Angriffsmöglichkeiten entstehen, gibt der Absender zusätzlich ein *Zeitfenster* an, in dem die Nachricht eintreffen muß. Falls die Ankunftszeit außerhalb des Zeitfensters liegt, wird die Nachricht verworfen.

Im *Pool-Betrieb* wird, sobald die N-te Nachricht eintrifft, zufällig eine Nachricht ausgewählt und weiterverarbeitet [Franz 1997]. In diesem Fall ist die Verzögerung hinsichtlich eines Zählers, der jeweils beim Eintreffen einer neuen Nachricht erhöht wird, geometrisch verteilt: $\mathrm{P}(\{T = k\}) = (1 - \frac{1}{N})^{k-1}\frac{1}{N}$. Der Erwartungswert ist $E(T) = N$, d. h. eine Nachricht bleibt im Durchschnitt bis zum Eintreffen N weiterer Nachrichten gespeichert.

Möglich ist auch der *Intervall-Pool-Betrieb.* In dieser Betriebsart wird jeweils nach Ablauf eines festgelegten Zeitraums geprüft, wieviele Nachrichten im Pool gespeichert sind. Enthält der Pool $n > N$ Nachrichten, werden nacheinander $n - N$ Nachrichten zufällig ausgewählt und bearbeitet. Andernfalls werden keine Nachrichten bearbeitet. In einer Variante mit *variabler Pool-Größe* werden in jedem Intervall nur $c(n - N)$ Nachrichten bearbeitet, $0 < c < 1$. Der Pool hat somit eine Mindestgröße; wenn mit gleichbleibender Häufigkeit Nachrichten eintreffen, ergibt sich für eine (möglicherweise größere) Poolgröße ein Gleichgewicht. Bei einem plötzlichen Ansteigen des Netzverkehrs wird der Pool größer.

Der Nachteil der poolbasierten Betriebsarten besteht darin, daß keine maximale Verzögerungszeit angegeben werden kann. Jedoch bleibt die Sicherheit des Verfahrens auch dann unverändert hoch, wenn wenige Nachrichten gesendet werden.

Ein Angreifer kann die Eingabe und die Ausgabe des Mixes beobachten. Anhand dieser Beobachtung kann er eine Nachricht mit einer Wahrscheinlichkeit $\mathcal{G}$ korrekt dem Sender A zuordnen. Je größer die vom Mix erzielte *Konfusion,* desto kleiner ist $\mathcal{G}$.

In allen Betriebsarten beruht die Sicherheit des Mixes darauf, daß ausreichend viele Nachrichten gesammelt werden. Wird bei gleich vielen eintreffenden Nachrichten eine größere Anzahl von Nachrichten gesammelt, erhöht sich jedoch die Verzögerungszeit. Je mehr Nachrichten durch den Mix anonymisiert werden, desto mehr Nachrichten können gesammelt werden, ohne daß dadurch eine für den Nutzer inakzeptable Verzögerung auftritt.

3.3.3 Mix-Sequenzen

Wird mehr als ein Mix eingesetzt, erhöht sich die Zahl der geheimen Schlüssel, die kompromittiert werden müssen, um den Sender zu identifizieren.

Die Übertragungskapazität des zugrundeliegenden Kommunikationsnetzes kann besser ausgenutzt werden, wenn für alle Nachrichten ein fester Pfad vorgegeben ist, da dann keine Adreßangaben für die folgenden Mixe notwendig sind und es – da alle Nachrichten durch die Entschlüsselung gleichermaßen kürzer werden – nicht erforderlich ist, Nachrichten mit Padding aufzufüllen.

Da jeder Mix innerhalb der Kaskade nur Nachrichten von seinem Vorgänger erhält und an seinen Nachfolger sendet, ist das Mix-Netz einfacher zu verwalten und zuverlässiger, als wenn jeder Mix mit jedem anderen Nachrichten austauschen müßte [Gülcü 1996]. Auch ergeben sich kürzere Übertragungszeiten, da jeder Mix seinem Nachfolger einen bereits vollständigen Batch sendet.

Eine Mix-Kaskade mit Batchgröße N wird von N Nachrichten parallel durchlaufen. Somit ist die Wahrscheinlichkeit $\mathcal{G}$ genauso groß wie bei einem einzelnen Mix. Durch die Verwendung einer Kaskade erhöht sich also – sofern sie mehrere vertrauenswürdige Mixe enthält, die Sicherheit – nicht aber die Konfusion.

In einem System mit vielen Mixen ist es ineffizient, jede Nachricht durch alle Mixe zu senden. In dem Fall sollte der Nutzer die Möglichkeit haben, Mixe auf der Basis der Topologie des zugrundeliegenden Kommunikationsnetzes und seines Vertrauens zu wählen. Die Eingaben der Mixe sind dann voneinander verschieden. Die resultierende Konfusion kann daher bei gegebener Batchgröße größer sein als bei einem einzelnen Mix. In [Rackoff 1993] wird gezeigt, daß eine vor passiven Angreifern verborgene Permutation von n Nachrichten, also $\mathcal{G} = 1/n$, bei n Mixen der Batchgröße $N = 2$ bei in $\log(n)$ polynomiellen Pfadlängen erreicht werden kann.

In großen verteilten Mix-Netzen sind die meisten Betreiber dem Nutzer nicht bekannt; daher ist eine Auswahl nach Vertrauensgesichtspunkten nicht sinnvoll. Es kann jedoch davon ausgegangen werden, daß jedenfalls ein Teil der Mixe vertrauenswürdig ist. Die Anonymität kann dann durch die zufällige Auswahl einer hinreichend großen Anzahl von Mixen gewährleistet werden. Wenn die verwendeten Mixe aus der Menge der vertrauenswürdigen zufällig ausgewählt werden, werden Angriffe durch die Unsicherheit über den jeweils verwendeten Pfad zusätzlich erschwert. Wenn ein Teil der Mixe nicht vertrauenswürdig ist, besteht bei zufälliger Auswahl jedoch das Risiko, einen Pfad auszuwählen, in dem keine vertrauenswürdigen Mixe enthalten sind. In dem Fall wäre die Anonymität des Nutzers im Einzelfall kompromittiert. Falls die Zugriffe des Nutzers verkettbar sind, hat ein solches Ereignis weitreichende Folgen. Die Vorteile beider Auswahlmethoden lassen sich kombinieren, indem festgelegte, vermutlich vertrauenswürdige Mixe in jedem Pfad verwendet werden, zusätzliche Mixe jedoch zufällig ausgewählt werden.

Im Intervall-Batch-Betrieb ergeben sich wie bei Mix-Kaskaden kurze Übertragungszeiten, da die Takte der aufeinander folgenden Mixe aufeinander abgestimmt werden können. Im ereignisgesteuerten Batch-Betrieb und im Pool-Betrieb werden Nachrichten in jedem Mix verzögert. Der Erwartungswert der Verzögerungszeit steigt dabei proportional zur Pfadlänge.

3.3.4 Aktive Angriffe

Ein aktiver Angreifer kann Nachrichten unterdrücken und verfälschen. Gegen Angriffe durch verfälschte Nachrichten kann das Mix-Netz gesichert werden; Angriffe durch das Unterdrücken von Nachrichten sind nicht vollständig zu verhindern.

3.3.4.1 Verändern von Nachrichten

Vor dem Weiterleiten einer Nachricht prüft der Mix, daß er die jeweilige Nachricht noch nicht bearbeitet hat (Replay-Angriff, Abschnitt 3.3.2.2).

Möglicherweise kann ein Angreifer jedoch zu einer an den Mix gerichteten Nachricht weitere davon abgeleitete Geheimtexte erzeugen und an den Mix senden, ohne daß der Zusammenhang für den Mix zu erkennen ist. Wenn dieser die Eingaben in Ausgaben transformiert, die für den Angreifer erkennbar mit der echten Ausgabenachricht zusammenhängen, kann der Angreifer die zu der Nachricht gehörende Ausgabe identifizieren und so den Mix überbrücken. Bei einer direkten RSA-Implementierung von Mixen ist dieser Angriff möglich [Pfitzmann 1990b], nicht jedoch bei gegen *chosen ciphertext*-Angriffe sicherer Verschlüsselung.

Auch muß es unmöglich sein, Teile der Nachricht so zu ändern, daß sich das zur Prüfung auf Wiederholung (Abschnitt 3.3.2.2) verwendete Merkmal ändert. Der Mixmaster-Remailer [Cottrell 1995] verwendet für diese Prüfung eine 16 Bytes große *Message-ID,* die den Anfang des symmetrisch verschlüsselten Teils der Nachricht bildet. Da diese Verschlüsselung im CBC-Modus erfolgt, in dem ein Fehler im Geheimtext nur zu Fehlern im entsprechenden und dem darauffolgenden Klartextblock führt, bewirkt eine Verfälschung in den ersten acht Bytes, daß die entschlüsselte Nachricht eine andere *Message-ID* erhält, ansonsten aber identisch ist. Um diesen Angriff zu verhindern, muß der Mix die Integrität aller empfangenen Nachrichten prüfen.

Wenn, wie bei Verschlüsselung im ECB-Modus, identische Geheimtextblöcke zu identischen Klartextblöcken führen, ist ein aktiver Angriff durch Verfälschen von an Mixe gerichteten Nachrichten möglich: Der Angreifer dupliziert einen Block einer Nachricht an einer anderen Stelle und prüft, welche der Ausgabenachrichten das entsprechende Muster aufweist. Falls der Angreifer eine Nachricht verkürzt, indem er beispielsweise einen Teil des Paddings abschneidet, darf der Mix nicht eine entsprechend verkürzte umkodierte Nachricht ausgeben. Sonst könnte der Angreifer den Weg der Nachricht anhand ihrer Länge verfolgen.

Verhindert werden muß auch, daß ein Angreifer in einer Nachricht Informationen kodiert, die er nach dem Durchlaufen von Mixen der Ausgabenachricht wieder entnehmen kann, um Ein- und Ausgabe einander zuzuordnen. Statt des zufälligen Paddings könnte ein Mix von ihm gewählte Daten in eine Nachricht einfügen oder einen Teil der Nachricht durch von ihm gewählte Daten ersetzen. Informationen lassen sich auf diese Weise nicht in der Nachricht transportieren, da der erste ehrliche Mix der Sequenz die Nachricht mit einem dem Angreifer nicht bekannten Sitzungsschlüssel umkodiert; die Daten sind für den Angreifer nach der Umkodierung also nicht mehr lesbar. Das Verfälschen des Geheimtextes führt aber dazu, daß der zugehörige Klartext von der veränderten Position an nicht mehr lesbar ist; die resultierenden Daten sind pseudozufällig. Schon die Information, daß der Text an einer gegebenen Position verändert wurde, kann einem Angreifer die Zuordnung der Nachricht ermöglichen. Erkennen kann der Angreifer die Veränderung, wenn er den an den Empfänger gesendeten Klartext beobachtet, sowie in den Nachrichtenköpfen, die die mit ihm kooperierenden Mixe empfangen. Somit können ehrliche Mixe überbrückt werden.

Diese Angriff lassen sich verhindern, indem die Mixe die Integrität der von ihnen bearbeiteten Nachrichten anhand eines MAC prüfen. Im Nachrichtenkopf müssen dazu die vom Absender erzeugten MACs für die verschlüsselte Nachricht gespeichert sein. Wenn ein vom Mix berechneter MAC nicht dem vom Sender angegebenen entspricht, wurde die Nachricht verfälscht und muß verworfen werden. Diese Prüfung ist nicht möglich, wenn die Mixe zufällig erzeugtes Padding in weitergeleitete Nachrichten einfügen. Daher muß das Padding auf eine zwischen dem Absender und den jeweiligen Mixen vereinbarte Weise deterministisch erzeugt werden.

3.3.4.2 Deterministisches Padding im Mix-Netz

Wenn der Mix die als Padding einzufügenden Daten mit einem Pseudozufallszahlengenerator erzeugt, dessen Initialisierungswert vom Absender der Nachricht im jeweiligen Header vorgegeben wird, ist dem Absender die Form der Nachricht in jedem Schritt bekannt, und er kann entsprechende MACs erzeugen. Wenn die Nachricht den letzten Mix der Sequenz erreicht, enthält sie demzufolge die von allen vorangehenden Mixe eingefügten Padding-Daten in der Form, die durch „Entschlüsselung“ der Daten mit den jeweiligen Sitzungsschlüsseln entsteht, so daß der MAC von all diesen Daten abhängt.

Wenn der Nachrichtenkopf dem Nachrichtenkörper vorausgeht, hängt das Padding nach der Transformation durch die Entschlüsselungsfunktion seinerseits vom Inhalt der Nachrichtenköpfe – also von den verschlüsselten Sitzungsschlüsseln, MACs usw. – ab, da die Entschlüsselung in einem Modus erfolgen muß, in dem ein Geheimtextblock von allen vorangehenden Klartextblöcken abhängt (Abschnitt 3.1.1.1). Um dies zu verhindern, sollte der Nachrichtenkopf dem Nachrichtenkörper nicht vorangestellt werden, sondern ihm folgen. Alternativ dazu wäre es auch möglich, bei der symmetrischen Verschlüsselung der Nachricht in einem Feedback-Modus die Reihenfolge der Textblöcke umzukehren.

Bei der Erzeugung des MAC für einen Mix i muß die Form des Paddings, das von den Mixen 1 bis $i-1$ eingefügt und durch Entschlüsselung umkodiert wird, bekannt sein. Dieses Padding muß der Absender zunächst durch wiederholte Verschlüsselung des Nachrichtenkörpers und darauf folgende Entschlüsselung des Nachrichtenkörpers mit dem jeweils bereits bekannten Padding erzeugen, bevor er die endgültige Form der Nachricht durch erneutes Verschlüsseln erzeugen kann. Dies ist möglich, da bei der ersten Verschlüsselungsoperation der Inhalt der Nachrichtenköpfe noch nicht bekannt sein muß, und auch bei der Entschlüsselungsoperation der Inhalt der für die Mixe i bis n bestimmten Nachrichtenköpfe für das Ergebnis keine Relevanz hat.

Für die Verschlüsselung der Nachricht m ergibt sich damit bei einer Mix-Sequenz der Länge n folgender Algorithmus:

- Erzeuge alle Sitzungsschlüssel k_i
- Verschlüssele den Nachrichenkörper $N'_n := E_{k_n}(m)$
- Für Mix i von $n-1$ bis 1:

  Berechne die Nachrichtenkörper: $N'_i := E_{k_i}(N'_{i+1})$
- Setze $N^p_1 := N'_1$
- Für Mix i von 1 bis $n-1$:

  Erzeuge durch Entschlüsselung die Nachricht mit Padding $N^P_i := D_{k_i}(N^P_{i-1})$
- Setze $N_n := E_{k_n}(N^P_n) \,||\, MAC(N^P_n)$
- Für Mix i von $n-1$ bis 1:

  Erzeuge $N_i := E_{k_i}(N_{i+1}) \,||\, MAC(N_{i+1})$

3.3.4.3 Isolieren von Nachrichten

Ein Angreifer kann die von einem Mix erzielte Konfusion reduzieren, indem er eine große Anzahl von Nachrichten an den Mix sendet *(flooding)*. Da er seine eigenen Nachrichten nach der Umkodierung wiedererkennt, besteht über diese Nachrichten keine Unsicherheit. Kennt er $N-1$ der in einem Schub bearbeiteten N Nachrichten, ist die Zuordnung der N-ten Nachricht trivial. Daher sollte nach Möglichkeit verhindert werden, daß ein großer Teil der gleichzeitig bearbeiteten Nachrichten von einem Angreifer stammt.

Wie groß der Aufwand für einen *flooding*-Angriff ist, hängt von der Betriebsart der Mixe ab. Um eine bestimmte Nachricht zu verfolgen, kann der Angreifer sie abfangen und zusammen mit von ihm erzeugten Nachrichten an den Mix senden. Bei einer Mix-Kaskade im Batch-Betrieb muß er dafür $N-1$ Nachrichten erzeugen, die dann zusammen mit der fraglichen Nachricht die Kaskade durchqueren. Bei einem frei wählbaren Pfad müssen in jedem Mix $N-1$ Nachrichten vorliegen, wenn die Nachricht dort eintrifft. Der Angreifer muß daher insgesamt eine sehr große Zahl von Nachrichten erzeugen. Besonders viele Nachrichten muß der Angreifer im Intervall-Pool-Betrieb mit variabler Größe senden, bis alle in einem Mix gespeicherten Nachrichten von ihm stammen. Je mehr Nachrichten der Angreifer erzeugen muß, desto größer ist die Wahrscheinlichkeit, daß der Angriff entdeckt wird.

Eine weitere mögliche Maßnahme gegen das Isolieren von Nachrichten besteht darin, mit digitalen Signaturen sicherzustellen, daß die in einem

Schub bearbeiteten Nachrichten von verschiedenen Absendern stammen [Franz 1997]. Treffen zu viele Nachrichten von einem Absender ein, oder sind Nachrichten nicht mit einem von einem vertrauenswürdigen Dritten zertifizierten Signaturschlüssel signiert, lehnt der erste Mix diese Nachrichten ab. Die anderen Mixe können an dieser Stelle ebenfalls prüfen, um sicherzustellen, daß der erste Mix korrekt vorgeht. Mixe im Batch-Betrieb können mit einem Zero-Knowledge-Beweis zeigen, daß sie die Nachrichten protokollkonform umkodiert haben [Rackoff 1993].

Allerdings läßt sich, auch wenn mit kryptographischen Methoden sichergestellt wird, daß die gleichzeitig bearbeiteten Nachrichten von verschiedenen Absendern stammen, nicht feststellen oder verhindern, daß diese untereinander Informationen austauschen. Insbesondere in einem weltweiten offenen Netz, in dem die meisten Beteiligten den Mix-Betreibern nicht bekannt sind, bietet dieses Verfahren somit kaum Schutz.

Ein Mix kann die Unsicherheit über die zu einer Eingabe gehörende Ausgabenachricht erhöhen, indem er Dummy-Nachrichten erzeugt. Gegen den Empfänger der Nachricht bieten vom letzten vertrauenswürdigen Mix der Sequenz erzeugte Dummies keinen Schutz, da sie keine Unsicherheit über die Eingabe bewirken [Franz 1997].

Gibt ein vertrauenswürdiger Mix die umkodierten Nachrichten seinerseits unbeobachtbar aus, kann der Angreifer seine eigenen Nachrichten nicht mehr erkennen. Flooding-Angriffe werden dadurch wirkungslos. Sofern nicht alle Mixe gleichzeitig angegriffen werden, läßt sich dies erreichen, indem der Mix seine Ausgabe durch eine von ihm gewählte Mix-Sequenz weiterleitet (Umleitung, *Inter-Mix Detour)* [Gülcü 1996]. Wenn die von einem vertrauenswürdigen Mix ausgewählten Mixe mit dem Angreifer kooperieren, verursacht die Umleitung jedoch keine zusätzliche Unsicherheit über den Weg der Nachricht. Auch eine von einem mit dem Angreifer kooperierenden Mix trägt nicht zum Schutz der Nachricht bei.

Wenn ein Mix seine Ausgabe nicht nur an den nächsten Mix sendet, sondern an eine Gruppe von Mixen verteilt, kann der Angreifer den Weg der Nachricht nicht beobachten [Wayner 1996]. Sofern dem übertragenen Geheimtext keine Information über zu das verwendete Schlüsselpaar zu entnehmen ist, wird die Konfusion dadurch erhöht.

3.3.4.4 Unterdrücken von Nachrichten

Vereinfacht werden Flooding-Angriffe, wenn der Angreifer auch legitime Nachrichten unterdrücken kann. Außerdem ermöglicht das Unterdrücken von Nachrichten Angriffe gegen einzelne Teilnehmer.

Ein aktiver Angreifer, der den ersten Mix einer Kaskade oder den Übertragungsweg dorthin kontrolliert und die Ausgabe des letzten Mixes beobachten kann, kann Nachrichten isoliert in das Mix-Netz einspielen und die resultierende Ausgabe beobachten [Gülcü 1996]. Dieser zentrale Angriffspunkt existiert bei Mix-Netzen mit frei wählbarem Pfad nicht: Der

Sender kann eine Nachricht bei verschiedenen Mixen einliefern. Erzeugt er Nachrichten für verschiedene Mixe so, daß nach der Entschlüsselung durch einen Teil der Mixe für den letzten Teilpfad in beiden Fällen dieselbe Nachricht entsteht, erkennt der erste Mix dieses Teilpfads die Wiederholung und ignoriert die als zweite eintreffende Nachricht. Durch die redundante Übertragung wird ein höherer Aufwand zum Unterdrücken der Nachricht erforderlich.

Wenn der Nutzer vom ersten Mix eine signierte Empfangsbestätigung für die eingelieferten Nachrichten erhält, kann er nachweisen, daß seine Eingabe nicht korrekt bearbeitet wurde [Chaum 1981]; die Mixe zeigen mit einem Zero-Knowledge-Beweis ihr protokollkonformes Verhalten. Für den Pool-Betrieb ist keine maximale Verzögerungszeit festgelegt; daher kann dem Mix nicht nachgewiesen werden, daß er einen Angriff versucht hat [Franz 1997].

Es besteht jedoch auch die Gefahr, daß ein Nutzer aufgrund eines aktiven Angriffs oder auch eines Fehlers des zugrundeliegenden Kommunikationsnetzes keine Verbindung zu Mixen aufbauen kann. Die Möglichkeit, sich dagegen zu schützen, indem im Fehlerfall das gesamte Protokoll abgebrochen wird, ist bei einer großen Teilnehmerzahl nicht praktikabel.

Wenn der Angreifer Nachrichten eines Teilnehmers unterdrückt und zu einem späteren Zeitpunkt gesammelt an den Mix sendet, kann er aus den dann in entsprechend großer Zahl auftretenden anonymen Nachrichten auf die Zuordnung schließen. Die Wirksamkeit dieses Angriffs können Mixe durch das Prüfen eines Zeitstempels in der Nachricht einschränken.

3.4 Mix-Netze in interaktiven Anwendungen

Das Mix-Netz-Konzept ist besonders für die Versendung von E-Mail geeignet, für interaktive Anwendungen jedoch ineffizient. Da jeweils eine Nachricht vollständig übertragen und dann gegebenenfalls durch eine neue Nachricht beantwortet werden muß, sind viele asymmetrische Verschlüsselungsoperationen erforderlich. Bei kurzen Anfragen nimmt das erforderliche Padding einen großen Teil der Übertragungskapazität in Anspruch. Auch die Verzögerungen, die bei der anonymen Übertragung von Nachrichten durch ein Mix-Netz entstehen, sind für interaktive Dienste wie das WWW nicht akzeptabel.

Daher muß das nachrichtenbasierte Mix-Netz-Konzept für interaktive Anwendungen zu einem auf **anonymen Verbindungen** beruhenden Verfahren abgeändert werden. Um Verzögerungen bei der Nachrichtenübertragung zu verhindern, sind auch Kompromisse zwischen Sicherheit und Effizienz des Verfahrens nötig. Der Einsatz von *Mix-Kanälen* im Telefonnetz wurde in [Pfitzmann 1989] vorgeschlagen; eine Umsetzung für Internet-Dienste ist *Onion Routing* (Abschnitt 3.6.4). Bei diesem Verfahren bestehen Netzverbindungen zwischen Mixen, über die mit fester Rate verschlüsselte Daten übertragen werden. Diese Netzverbindung wird im Multiplexverfahren für die anonymen Verbindungen genutzt; wenn keine Nutzdaten zu senden sind, wird Padding übertragen. Auf entsprechenden Prinzipien beruht auch der *PipeNet*-Vorschlag [Dai 1998].

3.4.1 Verbindungsaufbau

Der Sender wählt für seine anonyme Verbindung einen Pfad durch das Mix-Netz aus. Zwischen den aufeinander folgenden Mixen muß jeweils eine Netzverbindung bestehen.

Um eine anonyme Verbindung durch einen Mix aufzubauen, erzeugt der Sender eine Nachricht, in der die Adresse des folgenden Mixes oder des Empfängers sowie kryptographische Parameter für die anonyme Verbindung angegeben sind, und verschlüsselt sie mit dem öffentlichen Schlüssel des Mixes. Diese Nachricht wird als *Verbindungsaufbaunachricht* bezeichnet.

Ein Mix, der eine Verbindungsaufbaunachricht erhält, verknüpft die ankommende mit der abgehenden Netzverbindung zu einer anonymen Verbindung. Solange diese Verbindung besteht, werden über sie mit einer festen Rate verschlüsselte Daten übertragen. Die Verbindungsaufbaunachricht enthält symmetrische Schlüssel für die Übertragung von Daten in Hin- und Rückrichtung, für die Berechnung eines MAC und ggf. für die Erzeugung von Padding.

Im Onion-Routing-Verfahren enthält die Verbindungsaufbaunachricht die schichtenweise für die folgenden Mixe verschlüsselten Informationen; aufgrund dieser Form wird sie auch als *Onion* bezeichnet. Ihr Aufbau entspricht genau dem einer Nachricht im ursprünglichen Mix-Netz. In diesem Fall muß die Umkodierung der durch das Mix-Netz wandernden Verbindungsaufbaunachricht längentreu erfolgen.

Da der Verbindungsaufbau in diesem Fall bei allen Mixen unmittelbar nacheinander erfolgt, wird es internen Angreifern erleichtert, die Verbindung durch das Mix-Netz zu verfolgen. Es ist jedoch nicht erforderlich, die Verbindungsaufbaunachrichten für alle Mixe gleichzeitig zu übertragen. Stattdessen kann der Sender auch eine bereits bestehende anonyme Verbindung nutzen, um nach einer beliebig langen Verzögerung eine Verbindungsaufbaunachricht an den folgenden Mix zu senden.

Zum Erkennen vor Wiederholungen können Verbindungsaufbaunachrichten Message-IDs und Zeitstempel enthalten.

3.4.2 Anonyme Verbindung

Nachdem die anonyme Verbindung durch ein Mix-Netz aufgebaut ist, verschlüsselt der Sender die zu übertragenden Daten mit den symmetrischen Schlüsseln, die er beim Verbindungsaufbau mit den Mixen vereinbart hat.

Diese verschlüsselten Daten sendet er an den ersten Mix. Der Mix wartet, bis er auf allen anonymen Verbindungen je Richtung eine vorgegebene Menge von Daten empfangen hat; dann entschlüsselt er die vom anonymen Sender an den Empfänger gerichteten Daten und verschlüsselt die Antworten des Empfängers mit den in der Verbindungsaufbaunachricht angegebenen Schlüsseln und leitet die umkodierten Daten weiter. Nach der Weiterleitung und Anonymisierung durch das Mix-Netz erhält der Empfänger die Daten unverschlüsselt. Die Antworten erreichen den anonymen Nutzer in verschlüsselter Form; er entschlüsselt sie der Reihe nach mit den von ihm für den Rückkanal angegebenen Schlüsseln.

Tritt in einer anonymen Verbindung eine Verzögerung auf, führt dies somit zu einer entsprechenden Verzögerung bei allen Ausgaben des Mixes. Dies ist erforderlich, da Angreifer andernfalls anhand der Übertragungszeiten erkennen könnten, welche Ausgabe zu einer Eingabe gehört.

Das Verändern von Daten durch einen aktiven Angreifer würde dazu führen, daß unentschlüsselbare Daten an den Empfänger weitergeleitet werden. Da dies zur Identifizierung des anonymen Absenders führen könnte, müssen die Mixe die Authentizität der Daten anhand von durch den Sender erzeugten MAC prüfen. Schlägt die Prüfung fehl, müssen sämtliche anonymen Verbindungen abgebrochen werden, um dem Angreifer keine Informationen darüber zu geben, welche Ausgabe des Mixes zu der von ihm manipulierten Eingabe gehört. Es besteht die Gefahr, daß diese Maßnahme von maliziösen Sendern für *denial of service*-Angriffe mißbraucht wird. Diesem Problem kann begegnet werden, indem die Möglichkeit vorgesehen wird, im Fall einer fehlgeschlagenen MAC-Prüfung die betroffene Nachricht bis zu der Stelle zurückzuverfolgen, an der der Fehler aufgetreten ist.

In der Rückrichtung ist es nicht möglich, MACs durch die Mixe prüfen zu lassen. Hier kann nur der anonyme Nutzer selbst nach dem Entschlüsseln der Nachricht von den Mixen erzeugte MACs überprüfen.

3.4.3 Schutz vor Verkehrsanalyse

Da es jederzeit möglich sein soll, Verbindungen auf- und abzubauen, sind interaktive Mix-Netz-Varianten anfällig für Verkehrsanalyse.

Im Gegensatz zu den Nachrichten fester Länge im originalen Mix-Netz sind anonyme Verbindungen durch ihre Länge zu unterscheiden. Zudem ist es nicht möglich, Nachrichten zu sammeln und nach erst nach einer gewissen Verzögerung in geänderter Reihenfolge weiterzuleiten. Auch wenn die Übertragung paketweise erfolgt, müssen die Daten ohne nennenswerte Verzögerung weitergeleitet werden. Zudem sind die zu einer anonymen Verbindung gehörenden Pakete verkettbar. Um dies zu verhindern, müßte für jedes Paket ein neuer Pfad durch das Mix-Netz gewählt werden, was mit einem erheblichen Aufwand für die asymmetrische Verschlüsselung verbunden wäre. Aber selbst dann bleiben die Anfangs- und Endpunkt der

Pakete, die in direkter Folge gesendet und kurz darauf empfangen werden, gleich, so daß weiterhin Angriffe möglich sind.

Hat ein Mix, der nicht nur lokal von einem Nutzer verwendet wird, mehrere ehrliche Nachbarn, so ist für Angreifer nicht nachvollziehbar, ob ein Mix Daten weiterleitet, oder ob sie vom lokalen Nutzer stammen. Bei einem Mix, der über eine Telefonverleitung angebunden und daher nur temporär aktiv ist, ist jedoch unwahrscheinlich, daß er neben dem Eigentümer auch von anderen Anwendern genutzt wird.

Zum Schutz gegen Verkehrsanalyse muß daher Padding eingesetzt werden. Wenn sich die Menge der übertragenen Daten zwischen den Mixen und gegebenenfalls zwischen dem Nutzer und dem von ihm verwendeten Mix sich nicht in Abhängigkeit von den Nutzdaten ändert, können externe Angreifer nicht mehr beobachten, wann Verbindungen auf- und abgebaut werden oder wie viel Daten zu welchem Zeitpunkt übertragen werden. Dies läßt sich erreichen, indem das Datenvolumen mittels Padding auf eine konstante Größe gebracht wird. Anpassungen an die tatsächlich benötigte Kapazität sind möglich, wenn sie nicht in unmittelbarem zeitlichem Zusammenhang mit dem Auf- und Abbau anonymer Verbindungen erfolgen.

Wird das Padding auf den Verbindungen zwischen je zwei benachbarten Mixen eingefügt, sind passive externe Angriffe verhindert. Die Mixe selbst kennen in dem Fall jedoch die Menge der übertragenen Nutzdaten, so daß interne Angriffe möglich sind.

Ein aktiver Angreifer kann so viele anonyme Verbindungen aufbauen, bis die vorab festgelegte konstante Größe erreicht ist. Wenn das Mix-Netz dann weitere Verbindungen akzeptiert, ändert sich die Menge der übertragenen Daten wieder in Abhängigkeit von den Nutzdaten. Um aktive externe Angriffe zu verhindern, müssen weitere anonyme Verbindungen in diesem Fall also abgelehnt werden, oder das Datenvolumen muß so weit erhöht werden, daß wieder Padding eingefügt wird.

3.4.4 Schutz gegen interne Angreifer

Um sich gegen interne Angreifer zu schützen, muß Padding auf der anonymen Verbindung zwischen dem Nutzer und dem letzten von ihm verwendeten Mix eingefügt werden. Dadurch können die dazwischen liegenden Mixe keine zeitlichen Veränderungen der übertragenen Datenmengen beobachten.

Der Auf- und Abbau anonymer Verbindungen ist notwendigerweise allen beteiligten Mixen bekannt. Um zu verhindern, daß die Anonymität hierdurch kompromittiert wird, sollten die Zeitpunkte des Verbindungsaufbaus bei den verschiedenen Mixen eines Pfades möglichst weit auseinander liegen. Daraus folgt, daß der Nutzer nicht erst dann einen Pfad durch das Mix-Netz aufbauen sollte, wenn er tatsächlich anonym kommunizieren will. Stattdessen sollte er eine Verbindung durch das Mix-Netz aufrechterhalten, wann immer das möglich ist. Die Zeitpunkte, zu denen der Pfad durch das Mix-Netz verlängert oder verkürzt wird, sollten zufällig gewählt werden, damit die einzelnen Schritte des Aufbaus einer anonymen Verbindung nicht verkettet und so einem einzelnen Nutzer zugeordnet werden können. Darüberhinaus ist es möglich, zufällig Verbindungen auf- und wieder abzubauen, auch wenn gar keine Daten gesendet werden sollen.

3.5 Anwendung für das WWW

Für die Akzeptanz der Nutzer ist es entscheidend, daß Anonymisierungsverfahren benutzerfreundlich realisiert werden. Die für anonyme Zugriffe notwendigen kryptographischen Operationen müssen durch ein vertrauenswürdiges System durchgeführt werden; die Kommunikation des Nutzers mit diesem System darf für Außenstehende nicht zu beobachten sein. Die nötige Software muß somit auf dem PC des Nutzers installiert werden.

Ein transparenter Zugriff über die gewohnte Benutzerschnittstelle ist möglich, indem der WWW-Browser des Nutzers die Anonymisierungs-Software als Proxy verwendet. Aufgabe eines solches Proxies ist es neben der Durchführung kryptographischer Operationen auch, identifizierende Daten aus dem Dialog zwischen Browser und Server herauszufiltern.

Gelegentlich ist es auch wünschenswert, Informationen anonym anbieten zu können. Auch zu diesem Zweck können Mix-basierte Verfahren [Goldberg 1997, Demuth 1998] angewendet werden (Empfänger-Anonymität). Ein Angebot muß jedoch beliebig oft abgerufen werden können, so daß Replay-Angriffe (Abschnitt 3.3.2.2) nicht verhindert werden können. Ein gewisser Schutz gegen Angriffe ist durch den Einsatz von Proxy-Caches möglich, da Anfragen dann schon aus dem Cache beantwortet werden können und nicht bis zum anonymen Anbieter weitergeleitet werden müssen. Eine höhere Sicherheit ist gegeben, wenn der Anbieter seine Informationen anonym an einen Server überträgt, auf dem sie zum Abruf bereitgehalten werden, ohne daß der Betreiber des Servers die Identität des Anbieters kennen muß. Protokolle zur Anonymisierung von WWW-Zugriffen können so auch zum Schutz von Informationsanbietern eingesetzt werden.

3.6 Bestehende Systeme

Die meisten praktisch einsetzbaren Anonymisierungsverfahren sind Mix-Netz-basiert. Mit dem DC-Netz [Chaum 1988] existiert ein Verfahren, das informationstheoretisch sichere Sender-Anonymität gewährleistet, sofern die Schlüssel zufällig gewählt und über einen sicheren Kanal ausgetauscht werden und bei Schlüsselvereinbarung mittels asymmetrischer Verschlüsselung kryptographisch sicher ist. Der Aufwand für dieses Verfahren ist jedoch sehr hoch, da alle Teilnehmer einen konstanten Strom verschlüsselter Daten senden und empfangen, wobei nur ein geringer Teil der Bandbreite für tatsächliche Kommunikation eingesetzt wird. Für den Einsatz in einem weltweiten Netz wie dem Internet ist dieses Verfahren daher nicht praktikabel.

Sofern darauf verzichtet wird, alle Teilnehmer in einer Anonymitätsmenge zusammenzufassen, können DC-Netze jedoch eingesetzt werden, um *lokale Anonymität* [Martin 1998] zu erzielen. Wenn die Benutzer anhand der Netztopologie in Gruppen eingeteilt werden, ist allerdings auch der Nutzen der Anonymisierung deutlich reduziert.

3.6.1 Crowds

Das *Crowds*-System [Reiter 1997] ermöglicht anonyme WWW-Zugriffe. Es bietet eine effiziente Möglichkeit, Anonymität gegenüber dem WWW-Server zu erreichen, wobei die Kommunikationsbeziehung für lokal abhörende Angreifer unbeobachtbar bleibt, schützt aber nicht gegen globales Abhören.

3.6.1.1 Verfahren

Der Nutzer ist Mitglied einer als *Crowd* bezeichneten Gruppe. Jeder Teilnehmer der Crowd betreibt einen *Jondo* genannten Prozeß. Um einen anonymen WWW-Zugriff durchzuführen, sendet der lokale Jondo die Anfrage an einen zufällig ausgewählten Jondo seiner Gruppe, möglicherweise an sich selbst. Ein Jondo, der eine Anfrage von einem anderen Teilnehmer empfängt, leitet sie mit Wahrscheinlichkeit p_f wiederum an einen zufällig ausgewählten Jondo weiter, andernfalls an den Server. Die Antwort des Servers wird auf dem so ausgewählten Pfad in der umgekehrten Richtung zum Nutzer übertragen. Die Kommunikation zwischen dem ersten und dem letzten Jondo wird mit einem vom Initiator erzeugten Schlüssel verschlüsselt, der in jedem Schritt verschlüsselt weitergegeben wird, also allen Beteiligten bekannt ist. Die weitergeleiteten Nachrichten werden von den Jondos nicht umkodiert.

3.6.1.2 Sicherheit

Für den WWW-Anbieter ist jedes Mitglied der Crowd als Sender gleich wahrscheinlich. Die Wahrscheinlichkeit, daß er den Zugriff dem Nutzer A korrekt zuordnet, ist $\mathcal{G} = \frac{1}{n}$. Würde der lokale Jondo sich nicht selbst auswählen, wäre $\mathcal{G} = \frac{1}{n-1}$, was bei kleinen Crowds nachteilig wäre.

Ein Angreifer, der die Kommunikation des Nutzers abhört, kann zwar feststellen, daß der Nutzer einen WWW-Abruf durchführt; wer der Empfänger ist, erfährt er aber nicht, da die Kommunikation mit den folgenden Jondo verschlüsselt wird. Wenn der Jondo des Nutzers den Zugriff jedoch selbst durchführt, kann der Angreifer die unverschlüsselte Kommunikation mit dem Server beobachten. Zusätzlich erfährt er, daß der Jondo des Nutzers die Anfrage nicht von einem anderen Jondo erhalten hat und die Antwort nicht weiterleitet. Wenn der lokale Jondo sich, wie in [Reiter 1997], auch selbst auswählen kann, führt er den Zugriff mit Wahrscheinlichkeit $\frac{1-p_f}{n}$ selbst durch. Zusätzlich besteht die Möglichkeit, daß der zufällige Pfad im Jondo des Initiators endet. Dieser Fall ist für den Angreifer erkennbar, da die Nachrichten zwischen den Jondos nicht umkodiert werden.

Wenn der Anteil ehrlicher Teilnehmer um einen von p_f abhängenden Faktor größer ist als der der kompromittierten, dann ist es für interne Angreifer, der eine Anfrage erhält, wahrscheinlicher, daß der Jondo die Anfrage von einem anderen ehrlichen Jondo erhalten hat, als daß es sich um eine Anfrage des lokalen Nutzers handelt.

Gegen einen Angreifer, der mit einem der beteiligten Jondos kooperiert und zusätzlich die Kommunikationsverbindung des Nutzers abhören kann, bietet das Verfahren keinen Schutz. Dieser Angreifer kann feststellen, daß der Nutzer den Zugriff selbst veranlaßt hat und, da er den verwendeten Schlüssel kennt, auch die Kommunikation zwischen Nutzer und Server entschlüsseln.

Ein Angreifer, der die Kommunikation des Nutzers und die Kommunikation zwischen dem letzten Jondo und dem Server abhören kann, kann die unverschlüsselten Daten anhand ihrer Länge und der Zugriffszeit den verschlüsselten zuordnen. Auch in diesem Fall ist die Anonymität des Nutzers somit nicht gewährleistet.

3.6.2 JANUS

Das System JANUS [Demuth 1998] soll zum einen die Anonymisierung von Zugriffen auf WWW-Server ermöglichen, zum anderen aber auch das anonyme Anbieten von Informationen anhand verschlüsselter URLs, die vom JANUS-Server entschlüsselt werden.

Bei der Anonymisierung von WWW-Zugriffen entspricht die Sicherheit dieses Verfahrens nur der eines anonymisierenden Proxys: Die Verbindung zwischen dem Nutzer und dem JANUS-Server kann mit SSL verschlüsselt werden; personenbezogene Daten werden aus dem HTTP-Dialog entfernt. Eine Verkettung von JANUS-Servern ist jedoch nicht möglich. Die RSA-Verschlüsselung des JANUS-Systems dient ausschließlich dem Verschleiern der URLs anonymer Informationsanbieter.

3.6.3 Freedom

Freedom [ZKS 1999] ist ein kommerzielles Anonymisierungssystem der kanadischen Firma *Zero-Knowledge Systems, Inc.*, das an das Mix-Netz-Prinzip angelehnt ist.

3.6.3.1 Aufbau des Netzes

Das *Freedom*-Netz besteht aus Servern, die von verschiedenen Internet-Providern, Organisationen und Personen im Auftrag von *Zero-Knowledge Systems* betrieben werden.

Aufgabe dieser Server ist zum einen die Anonymisierung auf IP-Ebene, so daß TCP-Verbindungen, aber auch andere IP-basierte Protokolle anonymisiert werden können *(Anonymous Internet Proxy, AIP),* und zum anderen die Anonymisierung von E-Mail *(Anonymous Mail Proxy, AMP).*

Zusätzlich zu den Servern existiert eine zentrale Datenbank *(Freedom Network Information Database),* in der Informationen über die Topologie des Netzes sowie über Status und Performanz der einzelnen Server gespeichert sind, und ein *Keyserver,* der die öffentlichen Schlüssel der Freedom-Server und aller Pseudonyme enthält.

Jeder Freedom-Server hat fünf benachbarte Server; die Kommunikation zwischen benachbarten Servern wird mit einem gemeinsamen Sitzungsschlüssel verschlüsselt. In der Initialisierungsphase ruft ein Server die Schlüssel seiner Nachbarn aus der zentralen Datenbank ab, sofern sie nicht bereits vorliegen. Danach beginnt er, mit dem verbindungslosen *User Datagram Protocol (UDP)* Daten an seine Nachbarn zu senden. Wenn keine Nutzerdaten zu übertragen sind, werden Padding-Pakete gesendet. Durch einen *Traffic Shaper* wird das Padding so gewählt, daß eine vom Betreiber des Mixes vorgegebene maximale Bandbreite nicht überschritten wird; die übertragene Datenmenge ist unabhängig von der Menge der jeweils gesendeten Nutzdaten. Sie kann aber in Abhängigkeit von Statistiken über die zu erwartende Auslastung variieren.

Die Funktionsweise der *Anonymous Mail Proxies* ähnelt derjenigen herkömmlicher Remailer [Möller 1998]. Damit ist es möglich, E-Mail unter Pseudonym zu versenden und mit Hilfe von *Reply-Blocks* auch zu empfangen. Reply-Blocks sind vom anonymen Empfänger vorab erzeugte Nachrichten, in denen ein Pfad durch das Mix-Netz zum Empfänger kodiert ist. Während der Reply-Block bei der Weiterleitung durch das Mix-Netz entschlüsselt wird, wird die Antwort bei jedem Schritt mit einem vom Empfänger angegebenen symmetrischen Schlüssel verschlüsselt.

3.6.3.2 Client-Software

Eine auf dem Computer des Nutzers installierte Client-Software sorgt für transparenten anonymen Zugang zu den Netzdiensten DNS, HTTP, HTTPS, SMTP, POP3, Telnet, SSH, IRC (bis auf direkte Verbindungen zwischen Clients) sowie über ein WWW-Interface zum Usenet. über die Client-Software kann der Nutzer die Pfadwahl durch das Mix-Netz steuern und seine digitalen Pseudonyme verwalten.

Der Nutzer kann die Pfadlänge innerhalb bestimmter Grenzen auswählen; die voreingestellte Länge beträgt 3. Die Server können auf Wunsch manuell ausgewählt werden, andernfalls wählt die Client-Software sie zufällig aus, wobei kein Server mehrfach verwendet wird und als erster ein dem Nutzer topologisch naher Server gewählt wird.

Als lokaler Proxy entfernt der Client personenbezogene Daten aus dem Datenstrom. Ausgehende Daten werden mit dem Schlüssel eines Pseudonyms signiert. Anhand dieser Signatur prüft *Freedom*, ob die anonyme Verbindung von einem berechtigten Nutzer stammt. Ein Nutzer erhält gegen Zahlung einer Gebühr die Berechtigung, eine bestimmte Anzahl von Pseudonymen zu verwenden [Lahey 1999]. Dadurch soll auch Mißbrauch verhindert werden, indem Pseudonyme bei mißbräuchlicher Verwendung gesperrt werden.

Anschließend werden die Daten über eine anonyme Verbindung durch das *Freedom*-Netz gesendet.

3.6.3.3 Anonyme Verbindungen

Ein Freedom-Server, der von einem Nachbarn ein Datenpaket empfängt, entschlüsselt zunächst mit dem gemeinsamen Schlüssel den Header des Pakets.

Handelt es sich um eine Verbindungsaufbaunachricht (`CREATE`), so wird sie mit dem privaten Schlüssel des Servers entschlüsselt. Verbindungsaufbaunachrichten enthalten einen *Anonymous Connection Identifier (ACI),* die Adresse des folgenden Servers, Angaben über die zu verwendenden kryptographischen Algorithmen für die Hin- und Rückrichtung, die symmetrischen Schlüssel sowie eine Gültigkeitsdauer. Diese Daten werden für die Dauer der Verbindung gespeichert. Alle weiteren Pakete werden dann mit dem zur Verbindung gehörenden symmetrischen Schlüssel entschlüsselt.

Bei Nutzdaten-Paketen werden der Schlüssel und die kryptographische Operation (Ver- oder Entschlüsselung) anhand des ACI festgestellt. Nach der Umkodierung wird ein neuer Header erzeugt und das Paket an den folgenden Server weitergeleitet, wobei wiederum Verbindungsverschlüsselung eingesetzt wird. Der letzte vom Nutzer ausgewählte Freedom-Server leitet die Daten zum gewünschten Ziel weiter. Durch ein `DESTROY`-Paket wird die Verbindung wieder geschlossen.

3.6.3.4 Sicherheit des Verfahrens

Die *Freedom*-Software liegt nicht im Quellcode vor. [ZKS 1999] enthält nicht genügend Informationen über das Verfahren, um die Sicherheit beurteilen zu können.

Durch die obligatorische Verwendung von Pseudonymen besteht die Möglichkeit, anonyme Nutzer mit Kooperation der Server-Betreiber zurückzuverfolgen, da zu jedem Pseudonym drei *Reply-Blocks* gespeichert sind, in denen jeweils ein Pfad durch das Mix-Netz zur Adresse des Nutzers kodiert ist. *Forward secrecy* ist somit aufgrund der mit temporären Sitzungsschlüsseln verschlüsselten Datenübertragung nur für die Vertraulichkeit der Nutzdaten gegenüber Außenstehenden gegeben, nicht aber in Bezug auf die Identität des Nutzers.

Da kein Padding nur auf den Verbindungen zwischen den Mixen eingesetzt wird, nicht aber zwischen dem Nutzer und den Mixen, ist Verkehrsanalyse durch interne und durch aktive externe Angreifer möglich (Abschnitt 3.4.3).

3.6.4 Onion Routing

Ein Onion-Routing-Netz [Reed 1998] besteht aus einer Anzahl von als *Onion-Router* bezeichneten Mixen, zwischen denen dauerhafte TCP-Verbindungen bestehen. Ein solches System befindet sich in dem Firewall, der das sichere Netz des Anwenders vom Internet trennt.

Der Zugriff auf das Onion-Routing-Netz erfolgt über *Anwendungs-Proxies,* so daß die Anonymisierung von Proxy-fähigen Anwendungen ohne Modifikation genutzt werden kann. Diese bestehen aus einem anwendungsspezifischen *Client-Proxy* und einem für das Onion-Routing-Protokoll zuständigen *Core-Proxy.* Die anonymisierte Verbindung endet in einem *Responder-Proxy,* der über eine TCP-Verbindung mit dem gewünschten Server kommuniziert.

Der Client-Proxy teilt dem Core-Proxy die Zieladresse mit und leitet Anfragen weiter, wobei er – sofern nicht nur Unbeobachtbarkeit, sondern auch Anonymität erreicht werden soll – identifizierende Daten wie Cookies entfernt. Gegebenenfalls erzeugt er anwendungsspezifische Fehlermeldungen. Client-Proxies existieren für HTTP, ftp, SMTP, und rlogin. Es ist auch möglich, mit einem IP-Tunnel beliebige Internet-Verbindungen zu anonymisieren.

Der Core-Proxy wählt einen Pfad durch das Onion-Routing-Netz, baut die Verbindung auf und leitet daraufhin die Daten vom Client-Proxy verschlüsselt an das Onion-Routing-Netz weiter.

3.6.4.1 Anonyme Verbindungen

Der Core-Proxy erzeugt eine Verbindungsaufbaunachricht („Onion"), die dem Mix-Protokoll entsprechend mehrfach verschlüsselt wird. Sie enthält für jeden Onion-Router eine Versionsnummer, Bezeichner für die Algorithmen, mit denen die Verbindung in Hin- und Rückrichtung verschlüsselt werden soll, die Adresse des folgenden Onion-Routers (oder 0, um den Responder-Proxy des Onion-Routers zu bezeichnen), die Gültigkeitsdauer der Verbindungsaufbaunachricht sowie einen Wert, aus dem die Sitzungsschlüssel abgeleitet werden. Im ersten Schritt wird zufälliges Padding angehängt; in den folgenden Schritten die für den jeweils nächsten Onion-

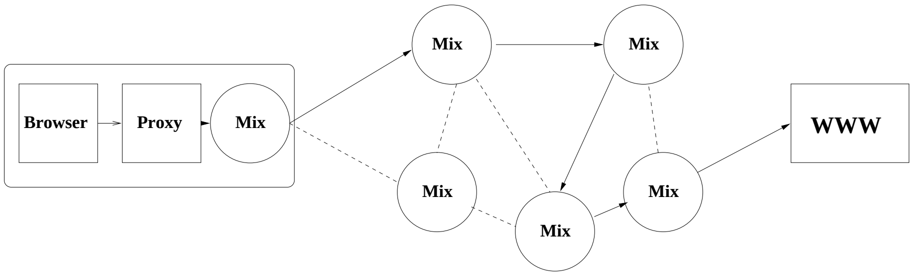

Abbildung 3.3: Anonyme Verbindung

Router bestimmte Nachricht. In Version 1 des Protokolls werden die ersten 1024 Bit der Nachricht direkt mit RSA verschlüsselt, die weiteren Daten mit DES im OFB-Modus. Die Pfadlänge ist auf 5 festgelegt. Für die Verbindungsverschlüsselung ist neben DES auch RC4 mit 128-Bit-Schlüsseln vorgesehen [Reed 1998]. Der beim Aufbau gewählte Pfad wird für die Dauer der gesamten Verbindung verwendet.

Ein Onion-Router, der eine Verbindungsaufbaunachricht erhält, entschlüsselt sie und prüft ihre Gültigkeit anhand der angegebenen Gültigkeitsdauer und eines Hash-Wertes, der zur Verhinderung von Replay-Angriffen gespeichert wird. Er erzeugt eine Datenstruktur, in der die ankommende und die ausgehende Verbindung sowie die Verschlüsselungssysteme für beide Richtungen miteinander verknüpft sind. Die Nachricht wird bis zu ihrer ursprünglichen Länge mit zufälligem Padding aufgefüllt und an den nächsten Onion-Router weitergeleitet.

Sobald die anonyme Verbindung besteht, können die mit den in der Verbindungsaufbaunachricht vereinbarten Schlüsseln stromchiffrierten Daten transportiert werden.

3.6.4.2 Verbindung zwischen Onion-Routern

Die Topologie des Netzes ist im voraus definiert. Jeder Onion-Router verfügt über DES-verschlüsselte Verbindungen zu einer festgelegten Menge von benachbarten Onion-Routern. Diese Verbindung wird im Multiplexverfahren für die anonymen Verbindungen sowie Steuerinformationen genutzt. Sie werden in *Zellen* fester Größe übertragen. Jede Zelle besteht aus einem für die betreffende dauerhafte Verbindung eindeutigen Bezeichner der anonymen Verbindung *(Anonymous Connection Identifier, ACI)*, der Angabe ihres Typs, der Längenangabe der folgenden Nutzdaten sowie bis zu 44 Bytes Nutzdaten, die mit Padding auf die Maximallänge aufgefüllt werden. Es gibt Zellen der Typen `Padding`, `Create`, `Data` und `Destroy`.

Die Verbindungsaufbaunachrichten werden in `Create`-Zellen transportiert, die Nutzdaten in `Data`-Zellen. Wenn eine Verbindung abbricht, wird eine `Destroy`-Zelle mit der entsprechenden ACI übertragen, die der Empfänger an den folgenden Onion-Router weiterleitet und mit einer weiteren

Destroy-Zelle bestätigt; die zugehörigen Datenstrukturen werden dann gelöscht. Padding-Zellen werden eingefügt, um die Verkehrsanalyse zu erschweren; sie werden vom Empfänger ignoriert. Die Verbindungsverschlüsselung wird für die bereits verschlüsselten Bestandteile der Create- und Data-Zellen ausgelassen.

Würde die Reihenfolge der ausgehenden Data-Zellen der der ankommenden entsprechen, wäre die Zuordnung der Verbindungen für einen Angreifer, der mit den benachbarten Onion-Routern zusammenarbeitet, trivial. Daher wird eine begrenzte Umsortierung vorgenommen: Zellen, die zu einem bestimmten Zeitpunkt von unterschiedlichen Onion-Routern gesendet wurden, können in beliebiger Reihenfolge weiterbearbeitet werden, ohne die Reihenfolge der Daten in einer Verbindung zu zerstören oder die Fairneß der Datenvermittlung zu beeinträchtigen. Solange diese Bedingungen erhalten bleiben, können mehrere Zellen einer Verbindung gesammelt und in geänderter Reihenfolge bearbeitet werden. Wenn zwischen interaktiven und nicht zeitkritischen Verbindungen unterschieden wird, bestehen bei letzteren verbesserte Möglichkeiten zur Umsortierung.

3.6.4.3 Sicherheit des Verfahrens

Zusätzlich zu den im einfachen Mix-Netz möglichen Angriffen sind bei Onion Routing einige weitere zu beachten.

Da die Verzögerung durch das Verfahren gering ist, wird die Verbindung zwischen Responder-Proxy und Empfänger nahezu gleichzeitig zur Verbindung zwischen Absender und Onion-Routing-Netz aufgebaut. Damit keine statistische Verkehrsanalyse möglich ist, muß die Kommunikationsverbindung zwischen dem Teilnehmer und seinem lokalen Onion Router also sicher sein, und der lokale Onion Router muß auch fremde Daten transportieren. Eine solche Konfiguration ist in vielen Fällen aber nicht gegeben [Reed 1998], insbesondere bei Privatanwendern.

Da die dauerhaften Verbindungen verschlüsselt und mit Padding aufgefüllt werden, ist die Verkehrsanalyse für externe Angreifer trotz der kurzen Verzögerungszeiten gegenüber dem einfachen Mix-Netz sogar erschwert. Die Nachbarn eines ehrlichen Onion-Routers können das Padding jedoch erkennen. Die Sicherheit gegen interne Angreifer ist also deutlich geringer. Für Onion-Router ist die Dauer einzelner Verbindungen zu erkennen. Kooperierende Onion-Router am Anfang und Ende einer anonymen Verbindung können die Anonymität daher auch aufdecken, wenn in der Verbindung zusätzlich ehrliche Onion-Router verwendet wurden. Sicherheit gegen diesen Angriff besteht, wenn der erste – definitionsgemäß vertrauenswürdige – Onion-Router ehrliche Nachbarn hat, so daß der Angreifer nicht feststellen kann, woher die Verbindung stammt. Auch in diesem Fall ist die erzielte Konfusion erheblich geringer als im Mix-Netz.

Die in [Reed 1998] vorgeschlagenen Verschlüsselungsverfahren sind nicht optimal: Um *forward secrecy* zu ermöglichen, sollte entgegen

[Reed 1998] Verbindungsverschlüsselung für alle Zellen eingesetzt werden. Anstelle direkter RSA-Verschlüsselung wäre ein Verfahren wie OAEP zu bevorzugen. Die Schlüsselgröße des verwendeten DES ist mit 56 Bit nicht ausreichend; für langfristige Sicherheit sollten RSA-Schlüssel mit mehr als 1024 Bit verwendet werden können.

3.7 Entwurf eines Protokolls

Onion Routing hat die meisten Eigenschaften, die für eine interaktive Mix-Netz-Anwendung sinnvoll sind. Einige Verbesserungen sind jedoch möglich. Wenn anonyme Verbindungen mittels einer mehrfach verschlüsselten *Onion* aufgebaut werden, kann ein interner Angreifer durch den zeitlichen Zusammenhang feststellen, wie sich diese Nachricht durch das Mix-Netz bewegt. Besser ist es, dem Client die Möglichkeit zu geben, die anonyme Verbindung schrittweise auf- und auch wieder abzubauen. Wenn der Client dabei verzögert vorgeht, ist dieser Angriff nicht mehr möglich.

Um sicherzustellen, daß die Datenübertragung zwischen zwei Mixen nicht vom tatsächlichen Kommunikationsaufkommen abhängt, kann die Verbindung zwischen den Mixen in Kanäle eingeteilt werden, wobei ein Kanal entweder einer anonymen Verbindung zugewiesen oder mit Padding aufgefüllt wird. Da eine anonyme Verbindung durch ihren Kanal identifiziert ist, ist die Übertragung von ACIs damit nicht mehr erforderlich.

Wenn des weiteren geeignete Verschlüsselungalgorithmen eingesetzt werden und die Integrität der Daten anhand eines MAC sichergestellt wird, ergibt sich damit ein sehr sicheres Anonymisierungsverfahren mit vertretbarem Aufwand. Die konkrete Gestaltung eines solchen Verfahrens ist im folgenden Kapitel beschrieben.

Kapitel 4

Implementation

Bei der Anonymisierung von WWW-Zugriffen ist zwischen zwei Aufgaben zu unterscheiden: Ein *Client-Proxy* nimmt die HTTP-Requests des Browsers entgehen, entfernt daraus alle Daten, die den Nutzer identifizieren könnten und sendet den Request durch das Mix-Netz an den Server. Aus Antwort des Servers muß er wiederum alle Daten entfernen, die den Browser veranlassen könnten, die Identität des Nutzers – beispielsweise über ein nicht anonymisiertes Protokoll – zu offenbaren. Die anonyme und unbeobachtbare Verbindung zwischen dem Nutzer und dem Server wird dagegen vom *Mix-Netz* hergestellt.

4.1 Wahl der Programmiersprache

Damit das Anonymisierungssystem von vielen Nutzern verwendet werden kann und möglichst jeder Betreiber eines Internet-Servers, der das möchte, einen Mix betreiben kann, sollten sowohl der Proxy als auch der Mix auf den verbreitetsten Server-Betriebssystemen lauffähig sein. Ein Großteil der Server im Internet verwendet Unix-artige Systeme oder Windows NT. Es ist also sinnvoll, eine Programmiersprache zu wählen, die sowohl für Unix als auch für Windows NT vorhanden ist. Damit Nutzer den Proxy auf ihren privaten PCs installieren können, muß dieser auch unter Windows 95 lauffähig sein. Da das WWW interaktiv benutzt wird, muß die durch die Anonymisierung verursachte Verzögerung so gering sein wie möglich. Auch die Effizienz ist daher ein wichtiges Kriterium.

Perl ist auf vielen Unix-Systemen installiert; Windows NT wird ebenfalls unterstützt. Perl ist besonders für die schnelle Entwicklung kleinerer Programme geeignet. Nachteilig ist jedoch die geringe Ausführungsgeschwindigkeit.

Die Programmiersprache Java ist sehr portabel, und sowohl für Sockets als auch kryptographische Funktionen stehen Klassenbibliotheken zur

Verfügung[1]. Java steht für Windows NT und auch für viele Unix-Systeme zur Verfügung. Obwohl neben Bytecode-Interpretern inzwischen auch Java-Compiler existieren, reicht die Effizienz von Java-Programmen noch nicht an die von C-Programmen heran.

C ist durch seine Maschinennähe sehr effizient und aufgrund seiner Hochsprachеigenschaften gleichzeitig zur Entwicklung auch großer Programme geeignet. Nachteilig ist, daß die Speicherverwaltung zu großen Teilen dem Programmierer überlassen bleibt. Dies macht die Programmierung nicht nur aufwendig, sondern durch den daher notwendigen Umgang mit Zeigern sehr fehleranfällig. Viele Mängel der Sprache C sind in C++ [Stroustrup 1997] behoben. C++ erweitert C unter anderem um Möglichkeiten zur Datenabstraktion. Die Verwendung von Zeigern ist in vielen Fällen nicht mehr nötig; im Gegensatz zu Java sind Zeigeroperationen allerdings möglich. Bereichsprüfungen werden z. B. bei Zugriffen auf Vektoren nicht durchgeführt. Damit zusammenhängende Fehler werden wie in C nicht vom Compiler erkannt und sind auch zur Laufzeit teilweise nicht festzustellen.

Für Windows NT gibt es mehrere C++-Compiler. Der frei verfügbare Compiler GNU C++ wurde an eine Vielzahl von Unix-Systemen angepaßt und gehört zum Lieferumfang von beispielsweise Linux-Systemen. Bei der systemnahen Programmierung bestehen allerdings Unterschiede zwischen Windows NT und Unix. Wo es erforderlich ist, kann der für das jeweilige System vorgesehene Programmcode durch bedingte Kompilierung ausgewählt werden.

Kryptographische Funktionen gehören nicht zur Standardbibliothek von C++. Es gibt jedoch eine Reihe von Bibliotheken, die diese Funktionen zur Verfügung stellen. Weit verbreitet ist die Bibliothek OpenSSL, die Implementationen verschiedener kryptographischer Algorithmen vor C- und C++-Programme zur Verfügung stellt (Abschnitt 4.5).

Unix-Systeme stellen für die Kommunikation die *socket*-Schnittstelle zur Verfügung [Stevens 1998]. Damit ist es möglich, TCP-Verbindungen aufzubauen und wie auf Dateien lesend und schreibend auf sie zuzugreifen. Unter Windows wird diese Funktionalität mit geringen Abweichungen von der WINSOCK-Bibliothek zur Verfügung gestellt.

Für eine effiziente Implementation des Anonymisierungssystems ist C++ demnach in Verbindung mit OpenSSL eine geeignete Programmiersprache. Sie wird dem in dieser Arbeit entwickelten Code zugrundegelegt.

4.2 HTTP-Proxy

Ein anonymisierender HTTP-Proxy hat die Aufgabe, Anfragen von Browsern entgegenzunehmen und so weiterzuleiten, daß so wenig Informatio-

[1]Aufgrund der Exportbeschränkungen für Verschlüsselungssysteme aus den USA gibt es neben der nur in den USA erhältlichen Java Cryptography Extension (JCE) auch eine international frei verfügbare Implementation namens IJCE.

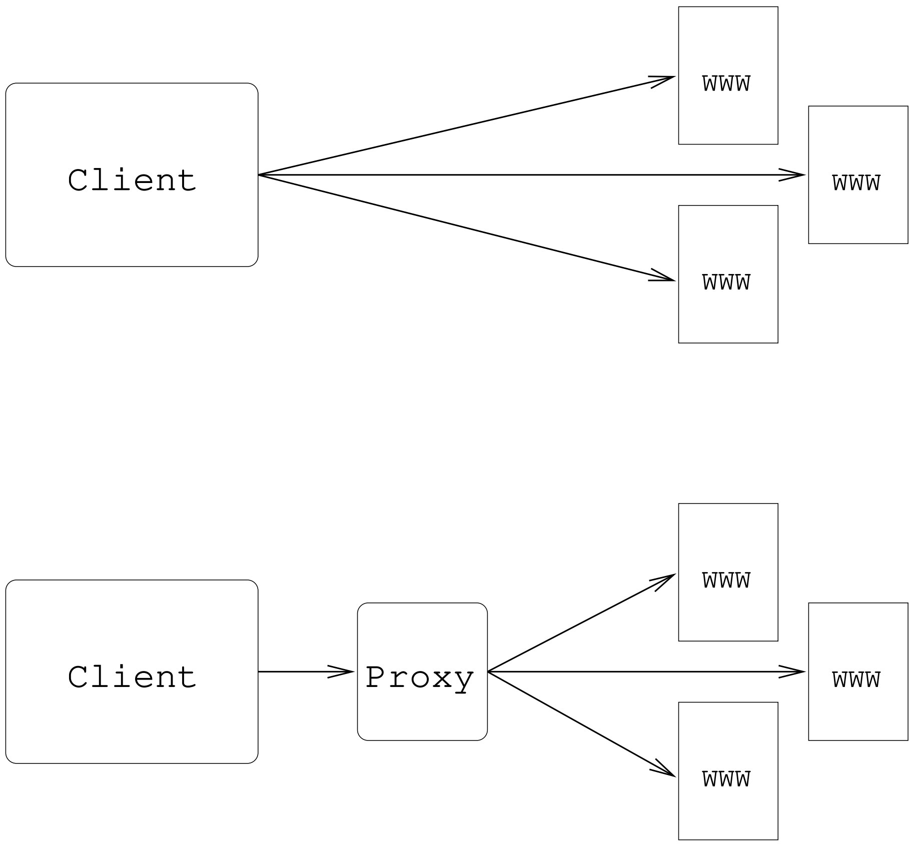

Abbildung 4.1: WWW-Zugriffe: direkt und über einen Proxy

nen über den Nutzer übermittelt werden, wie es möglich ist, ohne die Benutzbarkeit stark zu beeinträchtigen.

4.2.1 Request-Nachrichten

Ein Proxy agiert gegenüber dem Browser als ein HTTP-Server, für den eigentlichen Server hat er die Rolle eines Clients. Wenn der Proxy eine Anfrage erhält, sendet er eine entsprechende Anfrage an den Server.

Transparente Proxies leiten die Anfrage weitgehend unverändert weiter. Bestimmte Header-Zeilen, die sich nur auf die Verbindung zwischen Browser und Proxy beziehen, müssen gelöscht werden. Beim Zugriff auf den endgültigen Server ändert sich auch die Form des des `Request-URL`, da HTTP/1.0-Server anstelle eines vollständigen URL nur die Pfad-Komponente (`abs_path`) [Berners-Lee 1998] erwarten.

Ein anonymisierender Proxy muß darüberhinaus weitere Header-Zeilen entfernen. Je nach Art der Header-Zeile können sich dabei Einschränkungen bei der Nutzung bestimmter WWW-Server ergeben.

Referer-Zeilen ermöglichen Analysen über das Verhalten der Nutzer und sollten daher nicht übertragen werden. Ebenso müssen From-Zeilen herausgefiltert werden, falls ein Browser sie erzeugt. Auch die Übertragung von Cookie-Headern [Kristol 1997] ist aus Datenschutzgründen nicht sinnvoll. Hiervon ausgenommen sind neuere Browser, die Cookies nur auf ausdrückliche Bestätigung des Nutzers anlegen.

User-Agent und weitere Browser-spezifische Header können es ermöglichen, Nutzer zu identifizieren, haben aber keine Bedeutung für die Anfrage selbst. Allerdings verwenden manche Server diese Zeilen, um auf bestimmte Browser abgestimmte Seiten zu übertragen. Die Unterdrückung solcher Header kann dann dazu führen, daß dem Nutzer Daten nicht angezeigt werden, obwohl sein Browser in der Lage ist, sie anzuzeigen. Daher kann es sich anbieten, eine fiktive Versionsangabe zu übertragen.

Die Header Accept, Accept-Charset, Accept-Encoding und Accept-Language übermitteln ebenfalls Informationen über den Nutzer. Die Liste der akzeptierten Datentypen ist nützlich, um Informationen optimal darstellen zu können. Da diese Möglichkeit relativ selten genutzt wird und in den meisten Fällen ein verbreitetes Format als Default ausgewählt wird, ist ihr Fehlen jedoch ein akzeptabler Nachteil. Das Fehlen des Accept-Language-Headers ist unproblematisch, da fast alle mehrsprachigen WWW-Server eine manuelle Auswahl der Sprache ermöglichen. Accept-Encoding ermöglicht die Auswahl von Transportkodierungen, insbesondere von Kompressionsverfahren. Wenn hierauf nicht verzichtet werden soll, könnte der Proxy selbst beispielsweise die gzip-Kompression unterstützen.

Der Authorization-Header ist unproblematisch, da er nur einen vom Nutzer selbst bewußt eingegebenen Namen und ein Paßwort enthält. Der Zugriff auf paßwortgeschützte Seiten ist somit unter einem Pseudonym möglich, wenn der Inhaltsanbieter es zuläßt.

Das Herausfiltern des Headers If-Modified-Since führt dazu, daß Seiten noch einmal übertragen werden, auch der Browser die aktuelle Version bereits gespeichert hat. Wird diese Zeile mitgesendet, werden jedoch auseinanderliegende Zugriffe eines Nutzers auf denselben Server verkettbar. Es ist daher sinnvoll, ihre Übertragung abhängig von den Sicherheitsbedürfnissen des Nutzers konfigurierbar zu machen.

Rückfragen an den Nutzer können auch sinnvoll sein, wenn der Proxy einen unbekannten Header oder eine neue Zugriffsmethode feststellt, die zum Zeitpunkt der Implementation noch nicht bekannt waren.

4.2.2 Response-Nachrichten

Im Gegensatz zu Requests können die Antworten der WWW-Server ohne große Änderungen weitergeleitet werden.

Wenn Cookies herausgefiltert werden sollen, dürfen `Set-Cookie` und in anderen Versionen der Cookie-Spezifikation verwendete Varianten dieses Headers nicht an den Browser weitergeleitet werden. Wenn ein Cookie angelegt und nur bei der Anonymisierung der Anfragen nicht übertragen würde, könnten spätere nicht anonyme Zugriffe die Anonymität des Nutzers beim ursprünglichen Zugriff kompromittieren. Es reicht also nicht aus, nur die vom Browser an den Server übertragenen „Cookie"-Daten zu entfernen.

Wenn ein Browser Java oder JavaScript unterstützt, aber nicht die Möglichkeit gibt, diese Funktionalität zu deaktivieren, dann muß der Proxy entsprechende Elemente aus HTML-Dokumenten entfernen. Entsprechendes gilt für Hypertext-Links, die nicht anonymisierte Protokolle verwenden – insbesondere wenn sie durch das `src`-Attribut eines `<img>`- oder `<frame>`-Tags automatisch ausgeführt werden.

4.2.3 Weitere Protokolle

ftp kann in gleicher Weise über einen Proxy verwendet werden wie HTTP. Wenn der anonymisierende Proxy das ftp-Protokoll nicht selbst unterstützt, kann er die anonymisierte Anfrage seinerseits an einen ftp-Proxy weitergeben.

Aus SSL-verschlüsselte Verbindungen zwischen Browser und Server können Proxies keine Daten herausfiltern. Es besteht jedoch die Möglichkeit, unverschlüsselte Daten an den Proxy zu senden, der dann die gesicherte Verbindung zum Server aufbaut.

4.2.4 Der filternde Client-Proxy AFproxy

AFproxy ist ein filternder HTTP-Proxy, der zum einen direkte Verbindungen zu Servern herstellen kann, und zum anderen den Übergang zu einem Mix-Netz bildet.

Die Mix-Netz-Software stellt eine Socket-basierte Schnittstelle zur Verfügung, über die ein Pfad durch das Mix-Netz gewählt und die Nutzdaten übertragen werden können (Abschnitt 4.3.6). AFproxy ist dafür zuständig, WWW-Zugriffe in gefilterter Form durch das Mix-Netz zu leiten. Aufgabe eines derartigen Client-Proxy kann es zusätzlich sein, dem Nutzer die Konfiguration des Anonymisierungssystems in Form eines HTML-Formulars zu ermöglichen.

AFproxy wird auf dem System des Anwenders installiert und akzeptiert daraufhin auf einem konfigurierbaren Port TCP-Verbindungen. Im WWW-Browser wird `localhost` mit der gewählten Portnummer als Proxy eingetragen. Daraufhin führt der Browser alle HTTP-Requests über diesen Proxy durch, sie verlassen das System des Anwenders also nur noch in anonymisierter Form. Der Übergang zum Mix-Netz erfolgt über die in Abschnitt 4.3.6 beschriebene Schnittstelle. Der Client-Proxy wählt beim Programmstart einen zufälligen Pfad der vom Anwender vorgegebenen Länge durch das Mix-Netz. Alle anonymen WWW-Zugriffe der betreffenden Sitzung erfolgen dann über diesen Pfad.

4.3 WMix: Mix-Funktionalität

Ein Mix-Netz besteht aus einer Vielzahl von Knoten, den Mixen. Damit anonyme Zugriffe auf interaktive Dienste möglich sind, werden an Mixe die in Abschnitt 3.7 genannten Anforderungen gestellt. Das Programm WMix wurde entwickelt, um diese Mix-Funktionalität auf Unix- und Windows-Systemen zur Verfügung zu stellen.

Ein Nutzer, der das Mix-Netz nutzen möchte, installiert auf seinem Computer einen Client-Proxy (Abschnitt 4.2.4) und ebenfalls die Mix-Netz-Software WMix. Solange WMix aktiv ist, ist das System des Nutzers Teil des Mix-Netzes und kann jederzeit anonyme Verbindungen initiieren.

4.3.1 Konfiguration

Bestimmte Daten für die Konfiguration der Mixe werden im voraus festgelegt. Beim Aufbau einer anonymen Verbindung dient eine Liste in der alle aktiven Mixe mit ihrem Host-Namen, der Portnummer und ihrem öffentlichen Schlüssel verzeichnet sind, zur Wahl eines Pfades durch das Mix-Netz. Eine solche Liste kann über das WWW verbreitet werden.

Jeder Mix-Server hat eine Konfigurationsdatei, aus der hervorgeht, von welchen Servern und gegebenenfalls welchen Anwendern Verbindungen akzeptiert werden, und ob der Mix anonyme Verbindungen zu externen Servern durchführt. Außerdem ist in dieser Datei festgelegt, zu welchen anderen Servern der Mix auf welchem Port eine Verbindung aufbaut. In dieser Datei können auch symmetrische Schlüssel für die gesicherte Datenübertragung zwischen den Mixen angegeben und die für die Verbindung vorgesehene Bandbreite festgelegt werden.

4.3.2 Schlüsselverwaltung

Voraussetzung für die Sicherheit der asymmetrischen Verschlüsselung ist, daß die verwendeten öffentlichen Schlüssel zertifiziert werden. Geeignete Zertifizierungsinfrastrukturen stehen im Internet zur Verfügung; die derzeit verbreitetste ist PGP [Callas 1998]. Da WMix ebenso wie PGP RSA-Schlüssel verwendet, wäre es möglich, die öffentlichen Schlüssel der Mixe im PGP-Format zu speichern und entsprechende Zertifikate zu erzeugen. Eine flexiblere Lösung besteht jedoch darin, die Schlüssel als Textdatei zu signieren. Das WMix-Schlüsselformat bleibt dadurch unabhängig von der zur Zertifizierung verwendeten Software. Der Betreiber eines Mixes erzeugt einen PGP-Schlüssel, mit dem er den WMix-Schlüssel signiert; die Authentizität des PGP-Schlüssels wird durch das PGP-*web of trust* gewährleistet.

4.3.3 Algorithmen

Für die Implementation wurden als sicher geltende kryptographische Algorithmen (Abschnitt 3.1) ausgewählt. Die Verwendung der Algorithmen wird jeweils beim Verbindungsaufbau ausgehandelt, so daß die Auswahl der Algorithmen leicht an neue Erkenntnisse über die Sicherheit von Verschlüsselungsverfahren angepaßt werden kann. Zur asymmetrischen Verschlüsselung wurde RSA im OAEP-Modus mit 1024-Bit-Schlüsseln ausgewählt. Der symmetrischen Verschlüsselung dient Triple-DES mit 112-Bit-Schlüsseln und einem Initialisierungswert von 8 Bytes im *CBC*-Modus; die Integrität der Daten wird mit Hilfe von HMAC-SHA-1 gewährleistet. Damit ein Paket-Header einschließlich eines Typ-Bytes nicht mehr als 16 Bytes umfaßt, wird die Variante HMAC-SHA-1-120 eingesetzt, so daß der Hash-Wert nur 15 Bytes groß ist. Bei der Berechnung des MAC wird den eigentlichen Daten eine Sequenznummer des jeweiligen Pakets als Zwei-Byte-Wert vorangestellt, um zu verhindern, daß Pakete vertauscht werden können.

Die Ausgabe des Padding-Algorithmus soll ohne Kenntnis des geheimen Schlüssels nicht von Zufallsdaten zu unterscheiden sein. Falls der Algorithmus dieses Ziel nicht erreicht, erhalten Angreifer nur die zusätzliche Information, ob ein `Data`-Paket vom absendenden Mix weitergeleitet oder aber von einem lokalen Nutzer erzeugt wurde. Da lokale Nutzer eine Sequenz aus mehreren anderen Mixen verwenden, ist diese Information nicht sehr sensitiv. Als effizienter Algorithmus wird für diesen Zweck in WMix der auf einer Hashfunktion beruhende Algorithmus MGF1 [Kaliski 1998] verwendet.

Die verwendeten kryptographischen Algorithmen werden durch numerische Bezeichner repräsentiert. Folgende Algorithmen sind definiert:

NULL	0
RSA	1
NULL	0
3DES	1
NULL	0
HMAC-SHA-1	1
NULL	0
MGF1	1

Ist ein Verschlüsselungsalgorithmus NULL, werden die Daten unverschlüsselt übertragen. Für die MAC- und Padding-Algorithmen bedeutet

NULL, daß alle Ausgabebytes den Wert 0 haben. Die NULL-Algorithmen werden nur beim Verbindungsaufbau eingesetzt, bis die Kommunikationspartner mit Hilfe von `SessionKey`-Paketen Algorithmen und Schlüssel für ihre Verbindung vereinbart haben.

4.3.4 Verbindung zwischen zwei Mixen

Zwischen den Mixen bestehen langfristige Socket-Verbindungen. Das Protokoll wird in Runden ausgeführt; in jeder Runde senden alle Mixe jeweils eine feste Anzahl von *Paketen* vorgegebener Länge. Somit wird eine Verbindung durch Multiplexen für mehrere zur anonymen Kommunikation dienende *Kanäle* genutzt. Benachbarte Mixe vereinbaren die Anzahl der zwischen ihnen bestehenden Kanäle; die Zahl kann im laufenden Betrieb angepaßt werden, so daß immer ausreichend viele Kanäle zur Verfügung stehen, um neue anonyme Verbindungen aufbauen zu können. In nicht genutzten Kanälen wird Padding übertragen.

Sofern die Betreiber nichts anderes vereinbart haben, wird unmittelbar nach dem Aufbau einer Verbindung in jeder Runde ein Paket gesendet; die Verschlüsselungs- und Authentisierungsalgorithmen sind NULL. Die Kommunikationspartner können die Verschlüsselungsparameter und die Anzahl der Pakete je Runde aushandeln, indem sie `SessionKey`-Pakete austauschen. Der die Verbindung aufbauende Mix sendet zunächst `Padding`-Pakete, bis der andere Mix seine Konfigurationsdaten übertragen hat. Solange die Verbindung ungesichert ist, werden nur `Padding`-Pakete und `SessionKey`-Pakete für die Vereinbarung eines geheimen Schlüssels gesendet.

Da ein Mix die Bearbeitung der empfangenen Pakete erst beginnen kann, wenn er alle Pakete der jeweiligen Runde erhalten hat, ist die Geschwindigkeit des Netzes durch den langsamsten Knoten bestimmt. Die Bearbeitung erfolgt daher mit höchster Priorität, um die maximal mögliche Geschwindigkeit zu erreichen. Da das Programm durch einen `select`-Aufruf solange angehalten werden kann, bis auf einem der benutzten Sockets Daten zum Lesen bereitstehen, führt dieses Vorgehen nicht zu einer übermäßigen Auslastung der beteiligten Systeme.

4.3.5 Pakete

Ein Paket ist 256 Bytes groß. Folgende Typen sind definiert:

`Padding`	1	`Data`	9
`Create`	2	`RData`	10
`Connect`	3	`Userdata`	11
`ApplConnect`	4	`RUserdata`	12
`Close`	5	`CreateAck`	13
`Closed`	6	`Error`	14
`ApplClose`	7	`SessionKey`	15
`RApplClosed`	8	`ReqChannel`	16

Die Pakete haben folgenden Aufbau:

Byte	0:	Längenangabe
Bytes	1–127:	Nutzdaten + Padding
Bytes	128–239:	Padding
Byte	240:	Typ
Bytes	241–255:	MAC

Der Nachrichtenkopf bildet also, wie in Abschnitt 3.3.4.2 begründet, das Ende des Pakets. Für Pakettypen, die Daten variabler Länge enthalten können, ist Byte 128 die Längenangabe; die eigentlichen Nutzdaten sind in den Bytes 129–255 enthalten. Andere Pakete, wie etwa `ApplClose` enthalten keine Nutzdaten; Byte 0 hat dann den Wert 0.

Das Padding (Bytes 128–239) ermöglicht es, Pakete längentreu umzukodieren, um sie durch das Mix-Netz zu transportieren. Es wird in die Berechnung der MACs einbezogen, ist aber ansonsten bedeutungslos. Dadurch können Pakete in `Data`- und `RData`-Paketen gekapselt werden können. Damit alle Pakettypen gleich behandelt werden können, ist dieses Padding auch bei Paketen vorhanden, die nicht in anonymisierter Form durch das Mix-Netz übertragen werden.

`Data`- und `RData`-Paketen haben folgenden Aufbau:

Byte	0:	Längenangabe
Bytes	1–127:	Nutzdaten + Padding
Bytes	128–239:	Steuerinformationen für weitere Mixe
Byte	240:	Typ
Bytes	241–255:	MAC

Die Nutzdaten einschließlich Längenangabe und Padding sind dabei verschlüsselt. Bei den Steuerinformationen handelt es sich um aus Typangaben und MACs bestehende Abschnitte, die in der umgekehrten Reihenfolge des Pfades in verschlüsselter Form vorliegen.

Um Modifikationen an Paketen entdecken zu können, enthält jedes Paket einen *Message Authentication Code*. Der MAC richtet sich nach dem zwischen den Mixen vereinbarten Algorithmus und dem hierfür vereinbarten geheimen Schlüssel, bei über eine anonyme Verbindung übertragenen Paketen nach den in der Verbindungsaufbaunachricht angegebenen Parametern.

Wird ein Paket mit ungültigem MAC empfangen, muß der Mix es ignorieren und zur Verhinderung von Angriffen alle aktiven anonymen Verbindungen unterbrechen. Dies gilt auch für `Padding`-Pakete, da diese für Angreifer nicht von Daten zu unterscheiden sein dürfen.

Zusätzlich zur der für eine anonyme Verbindung zwischen dem Nutzer und dem jeweiligen Mix vereinbarten Verschlüsselung werden alle über die Socket-Verbindung zwischen zwei Mixen übertragenen Daten symmetrisch verschlüsselt. Der MAC wird geprüft, nachdem das empfangene Paket mit dem zur Socket-Verbindung gehörenden und gegebenenfalls mit dem vom Absender angegebenen Schlüssel dechiffriert wurde.

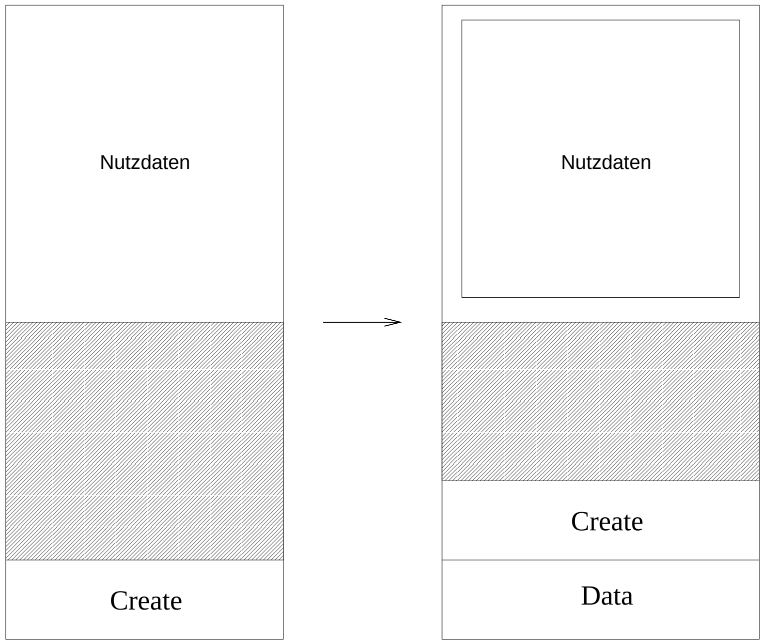

Abbildung 4.2: `Create`-Paket in einem `Data`-Paket

4.3.5.1 Padding

`Padding`-Pakete werden ignoriert. Sie dienen dazu, unabhängig vom tatsächlichen Kommunikationsaufkommen einen konstanten Datenstrom zu senden (siehe Abschnitt 3.4.3).

4.3.5.2 SessionKey

Nach dem Aufbau einer neuen Verbindung zwischen zwei Mixen wird ein `SessionKey`-Paket verwendet, um Algorithmen und Sitzungsschlüssel zu vereinbaren. Es enthält Schlüssel für die Ver- und Entschlüsselung sowie für die Überprüfung von MACs in beiden Richtungen.

Der Inhalt des Pakets ist mit dem öffentlichen Schlüssel des empfangenden Mixes verschlüsselt. Die Daten sind damit nur den beteiligten Mixen bekannt. Das Paket wird vom absendenden Mix nicht authentisiert; *man in the middle*-Angriffe auf das Mix-Netz sind dennoch nicht möglich, da die nachfolgenden Verbindungsaufbaunachrichten jeweils nur vom ausgewählten Mix entschlüsselt werden können.

4.3.5.3 ReqChannel

Ein Mix kann ein `ReqChannel`-Paket erzeugen, wenn die Bandbreite der Verbindung zu einem Nachbarn nicht ausreicht. In der auf das `ReqChannel`-Paket folgenden Runde ist die Anzahl der Kanäle dieser Verbindung um eins erhöht. Dies sollte nicht automatisch beim Aufbau neuer anonymer Verbindungen geschehen, da der Angreifer dann aus der Erhöhung der Anzahl von Kanälen auf den Verbindungsaufbau zurückschließen könnte. Stattdessen kann die Anpassung der Bandbreite in bestimmten Zeitabständen oder zu zufällig gewählten Zeitpunkten erfolgen. Die Mixe können ein Maximum an Kanälen vereinbaren; wenn sämtliche Kanäle durch anonyme Verbindungen belegt sind, können keine weiteren anonymen Verbindungen aufgebaut werden. Indem die Anzahl der Kanäle durch `ReqChannel`-Pakete erhöht, aber nur durch den Neustart des Systems reduziert werden kann, wird verhindert, daß Angreifer aus einer Reduzierung der Kanäle auf die Anzahl der bestehenden anonymen Verbindungen schließen kann.

4.3.5.4 Create

Eine Verbindungsaufbaunachricht wird dem Mix in einem oder mehreren `Create`-Paketen übermittelt. Hat die Längenangabe des Pakets den Wert 255, bedeutet das, daß die Nachricht im folgenden `Create`-Paket fortgesetzt wird; ansonsten gibt sie an, wieviele Bytes folgen.

Die Verbindungsaufbaunachricht beginnt mit einem Byte, das den verwendeten asymmetrischen Algorithmus angibt; de folgenden Daten sind asymmetrisch verschlüsselt. Der verschlüsselte Datenblock enthält Angaben der Algorithmen sowie die Schlüssel für die Verschlüsselung des Datenstroms in Hin- und Rückrichtung sowie zur Bildung der MACs und des Paddings für beide Richtungen. Zum Schutz gegen Replay-Angriffe kann der Mix einen Hash-Wert des Pakets speichern; als Zeitstempel ist das Erzeugungsdatum im `Create`-Paket enthalten.

Im OAEP-Modus können mit RSA bei Verwendung von 1024-Bit-Schlüsseln bis zu 86 Bytes verschlüsselt werden. Diese Größe wird bei der vorliegenden Implementation nicht überschritten. Bei größeren Verbindungsaufbaunachrichten müßten die darüberhinausgehenden Daten mit einem im RSA-verschlüsselten Teil enthaltenen Sitzungsschlüssel symmetrisch verschlüsselt werden.

Ein Mix, der eine Verbindungsaufbaunachricht erfolgreich entschlüsselt und für die neue anonyme Verbindung einen Kanal reserviert hat, bestätigt dies mit einem `CreateAck`-Paket. Nach dem Senden des `CreateAck`-Pakets werden die neu vereinbarten Sitzungsschlüssel verwendet. Während der Entschlüsselung darf die Datenübertragung nicht unterbrochen werden; falls die kryptographische Operation sehr zeitaufwendig ist, muß sie also in einem separaten Thread durchgeführt und auf dem betreffenden Kanal währenddessen `Padding`-Pakete übertragen werden. Andernfalls wäre durch die Verzögerungen für externe Angreifer zu erkennen, wie durch das Mix-Netz eine anonyme Verbindung aufgebaut wird.

4.3.5.5 CreateAck

`CreateAck`-Pakete enthalten lesbaren Text. Die erste Zeile enthält den Namen des Mixes, der die Verbindung bestätigt. Darauf folgen jeweils in einzelnen Zeilen die Namen aller benachbarten Mixe, zu denen dieser Mix eine anonyme Verbindung aufbauen kann.

4.3.5.6 Error

Ein `Error`-Paket wird erzeugt, wenn ein Fehler aufgetreten ist, beispielsweise eine gewünschte anonyme Verbindung nicht aufgebaut werden konnte. Der Inhalt dieses Pakets ist eine Beschreibung der Fehlersituation in natürlicher Sprache.

4.3.5.7 Connect

Ein Paket dieses Typs enthält den Namen des Mixes, zu dem eine anonyme Verbindung aufgebaut werden soll. Üblicherweise besteht der Name eines Mixes aus seinem Domainnamen und, getrennt durch einen Doppelpunkt, der Portnummer.

Nachdem der Mix für die anonyme Verbindung einen Kanal belegt hat, ist dieser ausgehende Kanal für die Dauer der Verbindung dem Kanal zugeordnet, auf dem der Mix das `Connect`-Paket erhalten hat.

4.3.5.8 ApplConnect

Ein `ApplConnect`-Paket teilt dem empfangenden Mix mit, daß eine TCP-Verbindung zu einem WWW-Server oder einem anderen Internet-Dienst aufgebaut werden soll.

Hat die Längenangabe der Nutzdaten den Wert 0, dann baut der Mix eine Verbindung zu seinem vorgegebenen HTTP-Proxy auf. Andernfalls besteht die Adresse aus dem Hostnamen und, getrennt durch einen Doppelpunkt, der Portnummer des gewünschten Dienstes. Ein Mix muß nicht alle Netzdienste unterstützen; Beschränkungen sind sinnvoll, um keine Angriffe auf andere Systeme durch anonymes *remote login* zu ermöglichen.

4.3.5.9 ApplClose, RApplClosed

Mit einem Paket des Typs `ApplClose` wird der empfangende Mix aufgefordert, die TCP-Verbindung zu schließen. Die anonyme Verbindung bleibt dabei jedoch bestehen und kann zum Aufbau neuer TCP-Verbindungen genutzt werden.

Wenn der Mix feststellt, daß die TCP-Verbindung von der Gegenstelle geschlossen wurde, teilt er dies mit, indem er über die anonyme Verbindung ein `RApplClosed`-Paket sendet.

4.3.5.10 Data

In diesen Paketen leiten Mixe die verschlüsselten Daten der anonymen Verbindungen weiter. Ein `Data`-Paket besteht aus den eigentlichen Nutzdaten in den ersten 128 Bytes sowie aus sieben mal 16 Bytes, in denen jeweils das Typ-Byte und der MAC für die folgenden Mixe oder aber zufälliges Padding enthalten sind.

Der Mix entschlüsselt das `Data`-Paket und sendet dem folgenden Mix ein Paket, dessen Typ-Byte und MAC den Bytes 224–239 des entschlüsselten Pakets entnommen sind. Die übrigen aus Typ-Byte und MAC bestehenden Blöcke werden im neuen `Data`-Paket um jeweils eine Postion verschoben übernommen; die Bytes 128–143 werden nach dem vom Absender angegebenen Padding-Algorithmus aufgefüllt. Die Bytes 0–127 des neuen Pakets sind die entschlüsselten Bytes 0–127 des `Data`-Pakets.

4.3.5.11 RData

Der Rückkanal zum anonymen Nutzer wird durch `RData`-Pakete implementiert. Um ein Paket in ein `RData`-Paket zu transportieren, kopiert der Mix die in den Bytes 0–127 enthaltenen Nutzdaten; die Bytes 144-255 der Eingabe werden in Bytes 128–239 des `RData`-Pakets übernommen. Der Mix berechnet dann den MAC des Pakets und verschlüsselt es mit dem vom anonymen Empfänger vorgegebenen Schlüssel. Da in `RData`-Paketen die Steuerinformationen – abgesehen von den MACs, deren Verfälschung automatisch dazu führt, daß das Paket als ungültig erkannt wird – im Gegensatz zu `Data`-Paketen keine Bedeutung tragen, wird der MAC bei diesen Paketen nur über die Bytes 0–127 gebildet.

Der lokale Mix des Nutzers entschlüsselt das empfangene `RData`-Paket nacheinander mit allen für die anonyme Verbindung vereinbarten symmetrischen Schlüsseln und prüft anhand der MACs seine Integrität. Wenn er einen Fehler feststellt, darf er nicht sofort alle anonymen Verbindungen unterbrechen, da ein aktiver Angreifer dann erkennen würde, zu welchem Nutzer die von ihm gestörte anonyme Verbindung gehört. Stattdessen erzeugt er in der folgenden Runde ein `Data`-Paket, das bei dem Mix, dessen MAC fehlerhaft war, und bei allen in der Sequenz folgenden Mixen zu einem Fehler führt und damit den Abbruch der Verbindung veranlaßt, ohne daß die Identität des Nutzers erkennbar wird.

4.3.5.12 Userdata, RUserdata

In Paketen dieses Typs werden die Nutzdaten durch die anonyme Verbindung übertragen. Der Inhalt eines `Userdata`-Pakets wird an eine vorher mit `ApplConnect` aufgebaute TCP-Verbindung gesendet. Daten, die ein Mix über eine solche TCP-Verbindung empfängt, leitet er in `RUserdata`-Paketen über die anonyme Verbindung weiter.

4.3.5.13 Close, Closed

Close-Pakete weisen den empfangenden Mix an, die jeweilige Verbindung zu schließen. Der Mix sendet daraufhin ein Closed-Paket an den Mix, von dem er das Close-Paket über die anonyme Verbindung erhalten hat; in der folgenden Runde geben beide Mixe den Kanal frei. Dieser wird dann mit Padding-Paketen aufgefüllt, bis er für eine neue Verbindung benötigt wird.

4.3.6 Verbindung zum Anwender

Der Mix des Nutzers akzeptiert auf einem dafür vorgesehenen Port Verbindungen von lokalen Proxies. Die Kommunikation zwischen Proxy und Mix erfolgt – analog zu Protokollen wie HTTP – zeilenweise; jede Zeile wird mit <CR><LF> beendet.

Durch diese Aufteilung zwischen Mix und Proxies, die im Klartext miteinander kommunizieren, können mit geringem Aufwand zusätzliche Proxies für weitere Protokolle entwickelt werden. Der Proxy benötigt keine kryptographischen Operationen und hat dennoch die vollständige Kontrolle über den Aufbau der anonymen Verbindung.

Beim Aufbau einer Verbindung meldet sich der Mix zunächst mit einer Statuszeile und der Liste der benachbarten Mixe. Wenn der Mix Verbindungen zu einer Anwendung aufbauen kann, enthält die Liste zusätzlich den Text Application Proxy. Sie endet mit dem Schlüsselwort ok.

Daraufhin kann der Proxy Kommandos senden. Der Mix antwortet darauf mit dem Inhalt der von ihm empfangenen und entschlüsselten Daten. Der Mix leitet den Inhalt von RUserdata-, CreateAck- und Error-Paketen weiter. Die erste Zeile besteht aus dem Pakettyp gefolgt von der Länge der Nutzdaten in Bytes; in den folgenden Zeilen wird der Inhalt des Pakets wiedergegeben. Antworten, in denen keine Nutzdaten übermittelt werden, enthalten nur den Typ gefolgt von einem Punkt. Dies ist bei leeren RUserdata-Paketen und bei RApplClosed-Paketen der Fall. Padding-Pakete werden nicht weitergeleitet.

Folgende Kommandos sind definiert:

4.3.6.1 NODE

Dem Schlüsselwort NODE folgt ein Leerzeichen und darauf der Name des Mixes, zu dem eine anonyme Verbindung aufgebaut werden soll. Falls der Name nicht gültig ist, gibt der Mix eine Fehlermeldung zurück. Andernfalls erzeugt er ein Create-Paket, um eine anonyme Verbindung aufzubauen, und sendet sie an den Nachbarn. Bei einem erfolgreichen Verbindungsaufbau beantwortet der benachbarte Mix die Verbindungsaufbaunachricht mit einem CreateAck-Paket. Der in diesem Paket enthaltene

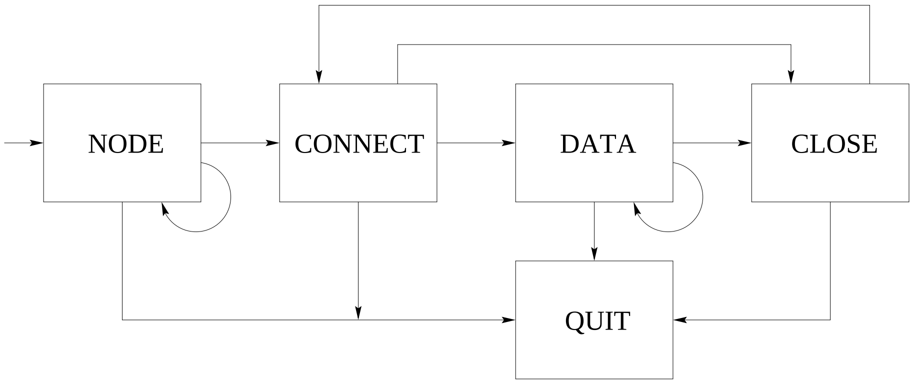

Abbildung 4.3: Abfolge der Kommandos

Text wird nach der Statuszeile an den Proxy übermittelt. Der Inhalt eines `CreateAck`-Pakets besteht entsprechend zum Verbindungsaufbau zum lokalen Mix aus einer Statuszeile und eine Liste der benachbarten Mixe. Statt der Bestätigung durch `CreateAck` kann der Proxy auch eine Fehlermeldung des Mixes in einem `Error`-Paket erhalten.

Weitere Kommandos, die der Proxy nach diesem Verbindungsaufbau sendet, werden anonymisiert an den ausgewählten Mix weitergeleitet. Sendet der Proxy ein `NODE`-Kommando, wenn schon eine anonyme Verbindung zu einem Mix besteht, dann wird diesem Mix zunächst ein `Connect`-Paket und darauf ein `Create`-Paket gesendet, so daß der Pfad der anonymen Verbindung um einen Mix verlängert wird. Das im `CreateAck`- oder `Error`-Paket enthaltene Ergebnis wird dem Proxy ausgegeben.

Bei einer bereits bestehenden längeren anonymen Verbindung werden das `Connect`- und das `Create`-Paket entsprechend der anonymen Verbindung verschlüsselt und in `Data`-Pakete eingepackt. Die in `RData`-Paketen verschlüsselt eintreffenden Antworten werden entschlüsselt und dann an den Proxy übermittelt.

4.3.6.2 CONNECT

Nachdem mit `NODE`-Kommandos eine anonyme Verbindung aufgebaut wurde, kann diese mittels `CONNECT` mit einem Empfänger verbunden werden. Der Mix erzeugt hierzu ein `ApplConnect`-Paket, das durch die anonyme Verbindung gesendet wird. Dem `CONNECT`-Kommando kann ein Argument folgen; wenn keines angegeben ist, verwendet der empfangende Mix seinen vorgegebenen HTTP-Proxy.

4.3.6.3 DATA

`DATA`-Kommandos sind nur zulässig, wenn eine Verbindung zu einem Empfänger besteht. Der dem Schlüsselwort `DATA` und einem Leerzeichen fol-

```
WMIX daphne:5551
Node: amadeus:5552
ok
NODE amadeus:5552
CreateAck: 66
WMIX amadeus:5552
Node: daphne:5551
Application Proxy
ok
CONNECT
RUserdata.
RUserdata.
RUserdata.
DATA GET /
RUserdata.
RUserdata: 26
<HTML>Beispieltext</HTML>
RApplClosed.
QUIT
```

Abbildung 4.4: Dialog zwischen Proxy und lokalem Mix (gekürzt)

gende Text wird verschlüsselt in `Userdata`-Paketen über die anonyme Verbindung gesendet und vom letzten Mix an den Empfänger weitergeleitet. Die Zeile kann beliebig lang sein; um mehrzeilige Daten zu senden, müssen jedoch mehrere `DATA`Kommandos gesendet werden.

Antworten des Empfängers treffen über die anonyme Verbindung in `RUserdata`-Paketen ein. Diese werden vom Mix entschlüsselt und an den Proxy weitergeleitet.

4.3.6.4 CLOSE

Durch ein `CLOSE`-Kommando wird der letzte Mix angewiesen, die Verbindung zur Anwendung zu schließen. Der Pfad durch das Mix-Netz bleibt jedoch bestehen und kann nach einem erneuten `CONNECT`-Kommando für weitere Datenübertragungen genutzt werden.

4.3.6.5 QUIT

Die Verbindung zwischen Mix und Proxy geschlossen. Anschließend wird die anonyme Verbindung abgebaut; zum Schutz gegen Verkehrsanalyse kann dies verzögert erfolgen.

4.4 Aufbau der Mix-Implementation WMix

Nach einer Initialisierungsphase durchläuft WMix eine Endlosschleife, in der Eingaben gelesen, verarbeitet und Ausgaben gesendet werden.

In der Initialisierungsphase liest WMix eine Konfigurationsdatei, initialisiert den Pseudozufallszahlengenerator (Abschnitt 4.5.2) und die Sockets und erzeugt bei Bedarf einen RSA-Schlüssel.

In der Eingabephase öffnet der Mix TCP-Verbindungen zu benachbarten Mixen und akzeptiert Verbindungen von Nachbarn sowie Client-Proxies. Außerdem liest er so lange Eingaben von den Sockets, bis er von allen aktiven Nachbarn alle für die jeweilige Runde vorgesehenen Pakte vollständig erhalten hat.

Daraufhin werden die empfangenen Pakete entschlüsselt und weitere durch den Pakettyp bestimmte Aktionen durchgeführt. Hierdurch werden die Ausgabepakete erzeugt; ungenutzte Pakete behalten ihren Initialisierungswert und sind daher `Padding`-Pakete. Die Pakete werden im folgenden Schritt mit den jeweiligen Verbindungsschlüsseln verschlüsselt und an die benachbarten Mixe gesendet; in dieser Phase erfolgt auch die Ausgabe an Client-Proxies.

Die benachbarten Mixe speichert jeder Mix in einer Liste von Elementen des Typs `Node`, in der Adresse, Schlüssel, Zustandsinformationen und eine Liste der dem jeweiligen Mix zugeordneten Kanäle gespeichert sind. Für Client-Proxies dient eine Liste von Elementen des Typs `Proxy`, für ausgehende Verbindungen zu Servern eine Liste von Elementen des Typs `Out`.

Zur Verwaltung der Kanäle dient der Datentyp `Channel`. Ein `Channel` enthält Zustandsinformationen, Schlüssel für symmetrische Verschlüsselung, MACs und Padding und Verweise auf den in der anonymen Verbindung vorangehenden und nachfolgenden Kanal. Zusätzlich enthält er Speicherbereiche, in denen empfangende und zu sendende Pakete gespeichert werden.

Der Datenübertragung dient die Klasse `Socket`. Sockets können mit der Member-Funktion `connect`, die einen Hostnamen oder eine IP-Nummer sowie die Portnummer als Argument erhält, mit einem entfernten System verbunden werden oder mit `listen` und `accept` Verbindungen von anderen Systemen akzeptieren. Wenn ein Socket verbunden ist, können mit den üblichen Operatoren << und >> sowie mit speziellen Funktionen Daten gelesen und geschrieben werden.

Die kryptographischen Operationen sind durch die Klassen `Cipher` (symmetrische Verschlüsselung), `PKCipher` (asymmetrische Verschlüsselung), `MAC` (Message Authentication Code) und `PAD` (Padding-Generator) implementiert. Member-Funktionen dieser Klassen rufen abhängig von der Auswahl des Algorithmus die OpenSSL-Funktionen auf, in denen die jeweiligen kryptographischen Algorithmen implementiert sind.

4.5 Die Verschlüsselungsbibliothek OpenSSL

OpenSSL [Young 1999] ist eine Bibliothek, die aus Implementationen kryptographischer Algorithmen und des SSL-Protokolls besteht. Sie war bis 1998 unter dem Namen SSLeay erhältlich.

OpenSSL ist in C programmiert und enthält Assembler-Implementationen der zeitkritischen Funktionen. Die Bibliothek kann unter Unix und Windows eingesetzt werden und darf nichtkommerziell wie auch kommerziell frei verwendet werden. Da die Bibliothek im Quellcode zur Verfügung steht, kann sie an die Bedürfnisse des Benutzers individuell angepaßt werden. Der in OpenSSL 0.9.1c noch nicht enthaltene neue OAEP-Modus für die RSA-Verschlüsselung konnte so zur Bibliothek hinzugefügt werden.

OpenSSL unterstützt die Algorithmen RSA, DSA, Diffie-Hellman, Blowfish, CAST, DES, IDEA, MDC2, RC2, RC4, RC5, MD2, MD5, RIPEMD-160, SHA-1 und HMAC [Schneier 1996]. Es stehen Funktionen zum Erzeugen von Pseudozufallszahlen und zum Umgang mit beliebig großen Zahlen zur Verfügung.

4.5.1 OAEP

In OpenSSL 0.9.1c wird RSA-Verschlüsselung nach PKCS #1 Version 1.5 sowie eine für SSL verwendete Variante unterstützt. Der in PKCS #1 Version 2.0 [Kaliski 1998] neu eingeführte OAEP-Modus wurde für die Verwendung in WMix neu implementiert.

Zum Erzeugen bzw. Prüfen der OAEP-Kodierung dienen die neuen Funktionen `RSA_padding_add_PKCS1_OAEP()` und `RSA_padding_check_PKCS1_OAEP()`. Diese Funktionen haben dieselben Parameter wie ihre PKCS1v1_5-Entsprechungen sowie zusätzlich den OAEP-Kodierungsparameter, der in [Kaliski 1998] vorgesehen ist, um für verschiedene Zwecke erzeugte Geheimtexte voneinender unterscheiden zu können, der aber im Normalfall die Länge 0 hat.

Diese Funktionen werden von den üblichen RSA-Funktionen `RSA_public_encrypt()` und `RSA_private_decrypt()` aufgerufen, wenn der Parameter `padding` den Wert `RSA_PKCS1_OAEP_PADDING` hat.

Zur Verwendung in der OAEP-Verschlüsselung definiert [Kaliski 1998] die Maskengenerierungsfunktion MGF1, die als Eingabe eine Zeichenkette und die Angabe einer gewünschten Ausgabelänge erhält und eine durch die Eingabe determinierte pseudozufällige Zeichenkette der gewünschten Länge ausgibt. MGF1 verwendet dazu eine Hashfunktion; in der vorliegenden Implementation ist hierfür SHA-1 vorgesehen.

4.5.2 Der Zufallszahlengenerator

Die Erzeugung von Zufallszahlen ist für die Sicherheit kryptographischer Systeme von entscheidender Bedeutung. Dies betrifft beispielsweise die

Erzeugung der Schlüsselpaare und der symmetrischen Sitzungsschlüssel. Wenn ein Angreifer die hierfür verwendeten Zufallszahlen errät, kann er die jeweiligen Daten entschlüsseln.

Die unmittelbare Erzeugung zufälliger Daten wird von derzeit üblicher PC-Hardware nicht unterstützt. Daher müssen per Software Pseudozufallszahlen erzeugt werden, die durch Angreifer nicht zu erraten sind. Dafür benötigt ein *Pseudozufallszahlengenerator (PRNG)* einen zufälligen Initialisierungswert, der so groß ist, daß die möglichen Initialisierungswerte nicht systematisch ausprobiert werden können. Auch muß verhindert werden, daß aus einer Folge von Pseudozufallszahlen auf vorhergehende oder nachfolgende Zahlen geschlossen werden kann.

Zur Erzeugung eines Initialisierungswertes kann auf die Hardware zurückgegriffen werden [Eastlake 1994]. Beispielsweise können physikalische Effekte in Festplatten als Quelle von Zufallszahlen genutzt werden. Auch die Bewegung des Mauszeigers oder die Eingabe zufälliger Tastendrücke durch den Nutzer ist für einen Angreifer nicht nachvollziehbar und kann daher für den Zufallszahlengenerator genutzt werden.

Viele neuere Unix-artige Betriebssysteme unterstützen die Erzeugung von Zufallszahlen, indem sie die Zeitpunkte von Hardware-Interrupts, Maus-Koordinaten und vergleichbare Angreifern nicht bekannte Daten verwenden. Wenn er vorhanden ist, nutzt OpenSSL diesen Zufallszahlengenerator; andernfalls muß die jeweilige Anwendung den PRNG von OpenSSL vor der Benutzung initialisieren. Hierfür stehen Funktionen zur Verfügung, mit denen der Zustand des PRNG in einer Datei gespeichert und wieder geladen werden kann, so daß nur vor der ersten Benutzung ein zufälliger Initialisierungswert erzeugt werden muß.

Kapitel 5

Performanz

Die Performanz ist für die Benutzerfreundlichkeit eines Anonymisierungsverfahrens von entscheidender Bedeutung, da ein Verfahren mit zu großer Verzögerung von den Anwendern nicht akzeptiert werden würde. Zudem spielt das Zeitverhalten des Netzes auch für die Sicherheit eine Rolle, da auf dem Zeitverhalten beruhende Angriffe möglich sind. Daher wurde die im vorhergehenden Kapitel beschriebene Implementation in einem praktischen Versuch erprobt.

5.1 Auswirkungen für den Anwender

Bei der Benutzung eines interaktiven Dienstes erwartet der Nutzer, daß seine Anfragen möglichst schnell beantwortet werden. Zu betrachten ist zum einen die *Verzögerung,* d.h. der Zeitraum zwischen dem Absenden der Anfrage und dem Beginn des Empfangens der Antwort und zum anderen der *Durchsatz,* also die Anzahl der übertragenen Bytes pro Sekunde.

5.1.1 Verzögerung

Eine Verzögerung ergibt sich dadurch, daß das Mix-Verfahren in Runden durchgeführt wird. Daten können also nicht sofort übertragen werden, sondern der Sender muß warten, bis ein hierfür zur Verfügung stehender Kanal an der Reihe ist. Jeder an der Übertragung beteiligte Mix empfängt sämtliche Pakete der jeweiligen Runde, bevor er sie entschlüsselt und in der folgenden Runde weiterleitet (Abschnitt 4.3.4). Die Verzögerung hängt somit von der Anzahl der verwendeten Mixe und für jeden einzelnen Mix von der Anzahl der verwendeten Kanäle ab.

Neben dieser durch das Weiterleiten der Daten durch das Mix-Netz bedingten Verzögerung verursacht die Anonymisierung auch eine Begrenzung des Datendurchsatzes.

5.1.2 Durchsatz

Da ein Mix empfangene Daten erst dann weiterleiten kann, wenn er alle zur jeweiligen Runde gehörenden Pakete empfangen hat, ist der Durchsatz des Mix-Netzes durch den Durchsatz der langsamsten Verbindung zwischen zwei Mixen begrenzt. Zudem werden zwischen je zwei Mixen in jeder Runde mehrere Pakete übertragen, so daß die zur Verfügung stehende Bandbreite auf die bestehenden Kanäle aufgeteilt werden muß. Da die Anzahl der bestehenden anonymen Verbindungen für externe Angreifer nicht erkennbar sein soll, müssen immer auch ungenutzte Kanäle bestehen, in denen nur Padding übertragen wird.

Verzögerungen, die auf zusätzlichem Verkehr im dem Anonymisierungverfahren zugrundeliegenden Datennetz oder auf anderweitiger Auslastung der Rechner beruhen, wirken sich auf alle anonymen Verbindungen aus.

Neben der Geschwindigkeit der Datenübertragung spielen auch die kryptographischen Operationen eine Rolle. Die Durchführung des RSA-Algorithmus ist aufwendig; sie tritt nur beim Verbindungsaufbau auf. Der Nutzer muß zudem alle gesendeten Pakete für alle von ihm ausgewählten Mixe nacheinander mit einem symmetrischen Algorithmus ver- und die empfangenen Pakete entschlüsseln. Jeder Mix führt für jedes weitergeleitete Paket eine symmetrische Verschlüsselungsoperation durch.

5.2 Praktische Erprobung

5.2.1 Durchführung

Zur Erprobung der Performanz wurden Mix-Netze in verschiedenen Konstellationen aufgebaut. Die Mix-Software wurde um die Möglichkeit ergänzt, die Typen der übertragenen Pakete mit Zeitstempeln versehen zu protokollieren. Zur Auswertung der Protokolldateien wurden Perl-Skripte eingesetzt.

Verwendet wurde zum einen ein Netz aus Windows 98-Systemen, zum anderen ein Unix-Netz, in beiden Fällen auf Ethernet-Basis. Die Funktion `GetSystemTime()` liefert die Zeit unter Windows 98 mit einer Auflösung von mehreren Hundertstelsekunden. Bei dieser Genauigkeit sind Auswertungen zwar möglich, eine höhere Auflösung ist jedoch wünschenswert. Im Sparc-basierten Unix-Netz können die Zeiten mit einer Genauigkeit einer Tausendstelsekunde gemessen werden.

Es wurden Mix-Netze verschiedener Größe eingesetzt und Pfade verschiedener Länge durch das Mix-Netz verwendet. Während eines Teils der Versuche wurde zusätzlicher Verkehr auf dem Ethernet erzeugt, indem parallel ein zweites Mix-Netz zum Einsatz gebracht wurde.

Um die Auswirkung der RSA-Verschlüsselung feststellen zu können, wurde eine zusätzliche Messung durchgeführt, bei der die asymmetrische

Konfiguration (Win98-Netz)	Minimum	Maximum	Durchschnitt	bei `Create`
2 Mixe	60	770	323	440
4 Mixe	0	2200	407	660
2 Mixe, 2. Mix-Netz parallel	0	1430	423	270
2 Mixe, kein RSA	60	710	387	440

Konfiguration (Unix-Netz)	Minimum	Maximum	Durchschnitt	bei `Create`
2 Mixe	94,7	191,8	101,7	112,3
4 Mixe	89,9	190,1	101,0	113,3
2 Mixe, 2. Mix-Netz parallel	87,7	180,0	99,0	101,5
2 Mixe, kein RSA	90,7	160,3	90,7	114,5
4 Mixe, kein RSA	96,2	140,2	100,7	108,6

Abbildung 5.1: Dauer einer Mix-Runde in 1/1000 Sekunden

Verschlüsselung ausgelassen wurde. Bei der Messung ergaben sich die in Tabelle 5.1 wiedergegebenen Zeiten.

Die symmetrische Verschlüsselung spielt eine geringere Rolle, da symmetrische Algorithmen schneller ablaufen. Zudem sind sie für die Verkehrsanalyse weniger wichtig, da ein Mix in jeder Runde annähernd gleichviele symmetrische Verschlüsselungsoperationen durchführt.

Im Testnetz mit zwei Mixen wurden Dateien verschiedener Größe übertragen. Die Übertragung einer 6 kB großen Datei durch das Mix-Netz dauerte 2,55 Sekunden. Für eine 50 kB große Datei waren 37 Sekunden erforderlich. Die Übertragung einer Datei, in der 10 *inline*-Grafiken angefordert wurden, dauerte 55 Sekunden. Hierbei ist durch eine andere Arbeitsweise der HTTP-Proxies eine eine deutliche Beschleunigung möglich. Die im Rahmen dieser Arbeit verwendeten Proxies leiten Nachrichten erst weiter, nachdem sie sie vollständig empfangen haben. Werden alle empfangenen Daten sofort weitergeleitet, ergibt sich insbesondere in Fällen, in denen große Dateien übertragen werden, eine geringere Verzögerung. Zu berücksichtigen ist auch, daß das Erzeugen der Protokolldatei selbst ebenfalls Zeit kostet. Im realen Einsatz sind daher höhere Geschwindigkeiten möglich, als sie bei der Messung festgestellt werden.

5.2.2 Auswertung

Bei der Dauer der Runden zeigen besonders im Windows-Netz sich deutliche Schwankungen. Im Fall von `Create`-Paketen ist die Verzögerung überdurchschnittlich groß; sie liegt jedoch in jedem Fall unterhalb der maximalen im Mix-Netz aufgetretenen aufgetretenen Verzögerungszeit. Es ergibt sich, daß die Verzögerung durch die RSA-Entschlüsselung auf den verwendeten Pentium-PCs akzeptabel ist. Im Unix-Netz zeigten sich geringere Schwankungen, aber auch in diesem Fall liegt die Verzögerung durch Verbindungsaufbaunachrichten unterhalb der maximal aufgetretenen Verzögerungszeit. Es ist demnach nicht erforderlich, die Entschlüsselung in einem separaten Thread durchzuführen.

Beim Einsatz eines Mix-Netzes im lokalen Netz war der Datendurchsatz im Vergleich zur nicht anonymisierten Form deutlich reduziert. Er lag jedoch in dem Bereich, der bei Modem-Übertragung üblich ist. Für ein verteiltes Mix ist eine vergleichbare Performanz möglich, wenn Mix-Netz-Verbindungen nur zwischen Systemen hergestellt werden, zwischen denen eine schnelle Internet-Verbindung besteht.

5.2.3 Ergebnis

Das WMix-Verfahren verursacht eine deutliche, aber in der Regel akzeptable Verzögerung beim Zugriff auf WWW-Seiten. Kryptographische Operationen sind anhand der Übertragungszeit nicht zu identifizieren, so daß zusätzliche absichtliche Verzögerungen zur Abwehr von Angriffen durch Verkehrsanalyse nicht erforderlich sind.

Kapitel 6

Sicherheit

Ein Mix-Netz ist sicher, wenn weder passive noch aktive Angreifer Informationen über die Nutzer gewinnen können. Vorausgesetzt ist in jedem Fall, daß ein Teil der Nutzer und mindestens einer der Mixe vertrauenswürdig ist (Abschnitt 3.2).

6.1 Verkehrsanalyse

Ein passiver Angreifer, der ein Mix-Netz beobachtet, kann gewisse Informationen über die Benutzung des Netzes erhalten. Zu untersuchen ist, welche Informationen der Angreifer bei der WMix-Implementation aus den übertragenen Daten gewinnen kann, und welche Informationen das Zeitverhalten des Mix-Netzes ihm gibt.

6.1.1 Externer Angreifer

Für Nutzer, die keine permanente Internet-Verbindung haben, ist in jedem Fall dadurch ein statistischer Angriff möglich, daß eine Korrelation zwischen den Zeitpunkten anonymer Verbindungen zu bestimmten Systemen und den Verbindungen eines Nutzers zum Mix-Netz festgestellt werden können (vgl. Abschnitt 3.4.3). Dieser Angriff läßt sich nur um den Preis

Angreifer	passiv	aktiv
extern	wirkungslos	denial of service
intern	Analyse des Verbindungsaufbaus	Verfälschung in der Rückrichtung

Abbildung 6.1: Mögliche Angriffe gegen das Mix-Netz

permanenter Datenübertragung verhindern; solange Internet-Zugriffe vorwiegend über Telefon-Einwahl erfolgen, ist er also nicht vermeidbar.

Wenn konkrete Datenübertragungen eines Nutzers beobachtet werden können, ist dieser Angriff nicht mit hohen Aufwand verbunden. Weniger Informationen stehen dem Angreifer zur Verfügung, wenn die Software des Nutzers während der Dauer der Verbindung in festen Abständen Daten sendet und empfängt, die tatsächlichen Sendevorgänge also durch Padding verdeckt werden. Dies ist bei WMix der Fall, da auf dem System des Nutzers ein eigener Mix mit Padding auf den Kommunikationsverbindungen läuft.

Nachdem eine Verbindung zwischen zwei Mixen aufgebaut wurde, findet dort die gesamte Datenübertragung verschlüsselt statt. Nur das `SessionKey`-Paket und das gleichzeitig vom anderen Mix gesendete `Padding`-Paket zu Beginn der Verbindung sind nicht symmetrisch verschlüsselt; für einen Angriff nützliche Daten sind ihnen jedoch nicht zu entnehmen. In Verbindung mit dem Padding auf der Verbindung führt dies dazu, daß passive externe Angriffe wirkungslos sind.

6.1.2 Interner Angreifer

Einem internen Angreifer bieten sich weitere Möglichkeiten der Verkehrsanalyse, da er den Typ und Inhalt der Pakete kennt, die die von ihm kontrollierten Mixe empfangen. Der Angreifer kann also den Verbindungsauf- und -abbau mittels `Create`- und `Close`-Paketen beobachten; wenn dieser in zeitlichem Zusammenhang mit einem anonymen Zugriff auf einen WWW-Server steht, tragen die in der Mix-Sequenz auf den Angreifer folgenden Systeme nicht zur Sicherheit bei. WMix ermöglicht es, eine Verbindung schrittweise auf- und abzubauen. Die in einer Verbindung verwendeten Mixe empfangen die jeweiligen `Create`-Pakete also nicht unmittelbar nacheinander; vielmehr der Zeitraum beliebig ausgedehnt werden, wenn der jeweils verwendete Client-Proxy diese Funktion unterstützt. Dadurch kann der Angreifer von den beobachteten Paketen nicht mehr auf tatsächliche anonyme Verbindungen schließen. Die Nutzdaten liegen bei allen Mixen mit Ausnahme des ersten und des letzten nur in verschlüsselter Form vor. Der erste Mix steht unter der Kontrolle des Nutzers. Der letzte kennt den Server, mit dem anonym kommuniziert wird und die übertragenen Daten. Wenn der Pfad einen Mix enthält, der nicht mit dem Angreifer kooperiert, ist der Nutzer jedoch auch gegenüber den nachfolgenden Mixen anonym.

6.2 Zeitverhalten

Eine Gefahr für die Vertraulichkeit besteht, wenn Angreifer aus Verzögerungen bei der Datenübertragung Rückschlüsse auf übertragene Inhalte ziehen kann. Diese Gefahr besteht besonders bei Verzögerungen, die durch die zeitaufwendigen asymmetrischen Verschlüsselungsoperationen entstehen.

6.2.1 Kryptographische Operationen

Für verschiedene Eingaben laufen kryptographische Operationen unterschiedlich schnell ab. Ein Angreifer, der den Inhaber eines geheimen Schlüssels dazu bringt, bestimmte Daten zu entschlüsseln, und die Zeit der Entschlüsselungsoperation mißt, kann dadurch auf den geheimen Schlüssel schließen [Kocher 1996]. Da Mixe automatisch Daten entschlüsseln, müssen Maßnahmen gegen diesen Angriff getroffen werden. Ein möglicher Schutz besteht bei RSA darin, den Geheimtext vor der Entschlüsselung durch Multiplikation mit einer Zufallszahl zu *blenden*. Diese Möglichkeit wird von OpenSSL unterstützt.

Einem externen Angreifer kann schon die Tatsache, daß ein Mix überhaupt eine RSA-Entschlüsselung durchführt, Informationen darüber geben, auf welchem Pfad eine anonyme Verbindung aufgebaut wird (Abschnitt 4.3.5.4). Wenn eine Entschlüsselung länger dauern würde als eine Runde des Mix-Protokolls, müßte der Mix daher ohne Unterbrechung weiter Daten übertragen, während er Verbindungsaufbaunachrichten entschlüsselt.

6.2.2 Verzögerung durch andere Prozesse

Keine Angriffspunkte entstehen, wenn die Datenübertragung eines Mixes dadurch verzögert wird, daß er auf Daten von einem benachbarten Mix warten muß, oder daß ein anderer Prozeß mit höherer Priorität als der Mix bearbeitet wird. Ausgenommen davon sind Verzögerungen auf dem Computer des Nutzers selbst, die durch die anonym abgerufenen Inhalte bedingt sind. Problematisch sind in der Hinsicht vor allem aktive Inhalte wie Applets, die aus Sicherheitsgründen deaktiviert werden sollten. Unter Umständen kann ein Angreifer jedoch schon aus der für das Darstellen einer HTML-Seite benötigten Zeit Rückschlüsse auf den Inhalt ziehen. Ein solcher Angriff läßt sich nicht vollständig verhindern.

6.3 Nicht-anonyme Datenübertragung

Eine Sicherheitslücke besteht, wenn ein WWW-Browser Daten überträgt, ohne den Proxy zu verwenden. Der Nutzer muß daher für alle Protokolle, auch beispielsweise für ftp, einen anonymisierenden Proxy eintragen.

Wichtig ist auch, daß der verwendete Browser keine DNS-Anfragen bezüglich der Adressen durchführt, auf die anonym zugegriffen werden soll. Bei Zugriffen über einen Proxy sind DNS-Anfragen nicht erforderlich; eine beliebte Funktion von Browsern besteht allerdings darin, unvollständige Hostnamen zu ergänzen, indem überprüft wird, ob der eingegebene Name mit den Ergänzungen `www` und `.com` oder `.net` existiert. *Netscape* und *Internet Explorer* führen diese Funktion bei Zugriffen über Proxy-Server nicht aus. Der Browser *Lynx* führt sie dagegen immer aus, wenn als Adresse kein vollständiger URL eingegeben wurde, auch wenn ein HTTP-Proxy eingetragen ist. Derartige Browser können nur eingesetzt werden, wenn alle DNS-Anfragen durch einen Firewall herausgefiltert werden.

6.4 Aktive Angriffe

Die im vorangehenden Abschnitt betrachteten Angriffe beruhen darauf, daß der Angreifer bestimmte Daten beobachtet und auswertet. Darüberhinaus ist zu berücksichtigen, daß ein Angreifer auch selbst in die Datenübertragung eingreifen kann, um Informationen zu erlangen.

6.4.1 Externer Angreifer

Wenn ein Angreifer eine bestehende Verbindung zwischen zwei Mixen stört, indem er Daten verfälscht, wird dies vom empfangenden Mix erkannt. Ein Mix, der ungültige Daten empfängt, oder feststellt, daß eine TCP-Verbindung geschlossen wurde, beendet sich. Daraufhin terminieren auch die benachbarten Mixe. Anonyme Kommunikation kann erst nach dem Neustart des Systems fortgesetzt werden. Dadurch sind *denial of service*-Angriffe möglich; die Anonymität der Nutzer ist aber nicht gefährdet.

Wenn der Angreifer bereits verhindert, daß zwischen zwei Mixen eine Verbindung aufgebaut wird, arbeiten die Mixe auf ihren anderen Verbindungen wie üblich. Daß Sicherheit durch ein kleineres zur Verfügung stehendes Mix-Netz reduziert ist, kann der Nutzer der Ausgabe seines Client-Proxies entnehmen und seine Nutzung daran anpassen.

Während des Verbindungsaufbaus zwischen zwei Mixen wird der symmetrische Sitzungsschlüssel mit dem öffentlichen Schlüssel des einen Mixes verschlüsselt; ein Angreifer könnte sich für den anderen Mix ausgeben, da dieser nicht kryptographisch authentifiziert wird. Der Angreifer könnte anonyme Daten jedoch nicht entschlüsseln, da sie in jedem Fall mit dem zertifizierten Schlüssel des echten Mixes verschlüsselt sind. Ein derartiger Angriff kann die Anonymität der Nutzer also nicht kompromittieren.

6.4.2 Interner Angreifer

Ein interner Angreifer kann seinen Nachbarn gültige Pakete senden. Auf bereits bestehenden anonymen Verbindungen wird der MAC jedoch vom Nutzer selbst erzeugt, so daß interne Angriffe dabei erkannt werden. In der

Rückrichtung ist die Prüfung des MAC nur durch den lokalen Mix des Nutzers möglich. Wenn dieser einen Angriff feststellt, sendet er in der folgenden Runde ein Paket, das für den Mix, bei dem der Angriff stattfindet, und die darauf folgenden Mixe ungültige Daten enthält *(jamming;* Abschnitt 4.3.5.11). Diese Mixe können den Angriff daraufhin erkennen und terminieren. Der interne Angreifer kann dann nicht feststellen, zu welchem Nutzer die von ihm gestörte Verbindung gehört. Anhand die Verzögerung bis zum Abbruch des Mix-Verfahrens kann er allerdings erkennen, durch wieviele Mixe das *jamming*-Paket weitergeleitet wurde, wie lang also der Pfad durch das Mix-Netz ist.

Gegen Angriffe auf die Kommunikation zwischen dem Nutzer und seinem lokalen Mix ist bei WMix kein Schutz vorgesehen. Hier muß der Nutzer sicherstellen, daß sein lokales System bzw. lokales Netz sicher ist.

Kapitel 7

Zusammenfassung und Ausblick

Mix-Netz-basierte Anonymisierungsverfahren bieten eine praktikable Lösung für die im Internet bestehenden Datenschutzprobleme. Bei Zugriffen auf das WWW fällt eine Vielzahl personenbezogener Daten an. Dies kann vermieden werden, indem die Zugriffe in anonymisierter Form erfolgen.

Das 1981 von D. Chaum zum Schutz von E-Mail konzipierte Mix-Netz wurde seither weiterentwickelt; es wurden verschiedene Varianten vorgeschlagen, mit denen Daten interaktiv übertragen werden können. Am Beispiel von Zero Knowledge Systems, Inc. zeigt sich, daß auch Interesse an der kommerziellen Nutzung dieser Verfahren besteht. Die bestehenden Verfahren wie Onion Routing weisen allerdings noch einige Schwächen auf; auch stehen der Nutzung von Onion Routing außerhalb der USA die Exportbestimmungen für kryptographische Software entgegen.

In dieser Arbeit wurde gezeigt, welche Anforderungen ein Anonymisierungsverfahren für WWW-Zugriffe erfüllen muß. Im Anschluß an eine Untersuchung der bestehenden Systeme in Bezug auf diese Anforderungen wurde ein Mix-Netz-basiertes Protokoll vorgeschlagen und in C++ für Windows und Unix implementiert. Eine Untersuchung der gegen dieses Verfahren möglichen Angriffe und eine praktische Erprobung zeigen, daß der Datenschutz hiermit gewährleistet werden kann. Es besteht jedoch ein Konflikt zwischen der Sicherheit und der Verfügbarkeit, da viele aktive Angriffe nur verhindert werden können, indem das gesamte Mix-Netz angehalten und neu gestartet wird. Hier muß für die einzelne Anwendung entschieden werden, ob *denial of service*-Angriffe hingenommen werden können, um einen zuverlässigen Schutz der Anonymität zu gewährleisten.

7.1 Mögliche Weiterentwicklungen

Die in dieser Arbeit entwickelte Software kann durch Anpassungen des Mix-Netzes an das Nutzerverhalten und durch Verbesserungen der Benutzerschnittstelle weiterentwickelt werden.

7.1.1 Verbesserung der Benutzerschnittstelle

WMix und AFproxy sind Textmodus-Programme, die Informationen über ihren Zustand zeilenweise ausgeben. Sie sind dadurch für Tests gut geeignet; um jedoch für die praktische Anwendung eine höhere Benutzerfreundlichkeit zu erreichen, sollte die von Anwendern installierte Software vollständig mit einer graphischen Oberfläche ausgestattet sein. Hierfür kann der WWW-Browser des Nutzers eingesetzt werden. Die Konfiguration kann dabei durch vom Proxy erzeugte HTML-Formulare vorgenommen werden. Rückmeldungen über den Zustand der anonymen Verbindung können in einem zusätzlichen Fenster des Browsers angezeigt werden.

7.1.2 Anpassungen des Mix-Protokolls

Es ist denkbar, daß Anpassungen des Protokolls an das Nutzerverhalten nötig werden. Wenn ein Mix-Netz viele Nutzer hat oder einige Nutzer die Datenübertragung absichtlich stören, sind häufige Übertragungsabbrüche zu erwarten. Die Robustheit des Mix-Netzes kann erhöht werden, wenn eine reduzierte Sicherheit akzeptabel ist. Um aktive Angriffe auf die Anonymität eines Nutzers zu verhindern, bricht WMix die gesamte Kommunikation ab, sobald die Integrität der Übertragung gestört ist (siehe Abschnitt 3.4.2). Wenn der maximal mögliche Schutz gegen aktive Angreifer nicht erforderlich ist, kann stattdessen nur die betroffene anonyme Verbindung geschlossen werden. Die Auswirkung eines *denial of service*-Angriffs ist dann auf die betroffene Verbindung beschränkt.

Auch Änderungen bei den eingesetzten Algorithmen sind möglich, wenn sich beispielsweise herausstellt, daß bestimmte in WMix verwendete kryptographische Algorithmen unsicher sind, oder wenn Algorithmen entwickelt werden, die bei vergleichbarer Sicherheit schneller sind als die in WMix eingesetzten. Dies ist problemlos möglich, da die jeweils verwendeten Algorithmen beim Schlüsselaustausch bzw. beim Verbindungsaufbau spezifiziert werden. Damit sind auch Anpassungen an Weiterentwicklungen in der Kryptologie möglich.

7.2 Vorschlag einer praktischen Mix-Architektur

Um Nutzern im Internet die Nutzung eines Mix-Netzes zu ermöglichen, muß die nötige Software und ein Verzeichnis der öffentlichen Schlüssel bereitgestellt werden. Hierfür sollte die Mix-Software WMix als Windows-Version und für Unix-artige Betriebssysteme im Quellcode auf einem ftp- oder WWW-Server zur Verfügung gestellt werden. Dieser Server sollte zusätzlich eine regelmäßig aktualisierte Liste der gegenwärtig aktiven Mixe und ihrer öffentlichen Schlüssel enthalten. Nützlich wären auch Informationen für neue Benutzer, in denen der Zweck und die Installation der Software dargestellt wird. Zudem sollte auf diesem System ein Mix betrieben werden, zu dem die Nutzer und weitere unabhängige Betreiber Verbindungen aufbauen können.

Um anonym auf das WWW zugreifen zu können, installiert der Nutzer die Software sowie die aktuelle Liste auf seinem Computer; er kann die Konfiguration an seine Sicherheitsanforderungen anpassen, indem er eine Pfadlänge für die zufällige Auswahl von Mixen wählt und Mixe aus der Liste entfernt, denen er nicht vertraut. Nachdem der Mix-Client im WWW-Browser als Proxy eingetragen wurde, erfolgen die WWW-Zugriffe dann anonym.

Wer selbst einen Mix betreiben will, benötigt eine permanente Verbindung zum Internet. Er kann einen oder mehrere der bereits vorhandenen Mixe als Nachbarn wählen, wobei es sich anbietet, Mixe auszuwählen, zu denen eine schnelle Internet-Verbindung besteht. Nach der Installation der Software trägt der Betreiber sein System in die Liste der aktiven Mixe ein. Beim Programmstart erzeugt WMix dann ein RSA-Schlüsselpaar. Der Betreiber teilt dem Verwalter der Mix-Liste die Adresse und den öffentlichen Schlüssel seines Mixes mit. Sobald der neue Mix in die Liste aufgenommen wird, kann er von allen Nutzern in anonymen Verbindungen verwendet werden.

Literaturverzeichnis

[Beaver 1992] D. Beaver: „Foundations of Secure Interactive Computing“. *Advances in Cryptology: Crypto '91*, Springer, 1992, S. 377–391.

[Bellare 1995] M. Bellare, P. Rogaway: „Optimal Asymmetric Encryption – How to Encrypt with RSA“. *Advances in Cryptology: Eurocrypt '94*, Springer, 1995, S. 92–111.

[Bellare 1997] M. Bellare, P. Rogaway: „Minimizing the Use of Random Oracles in Authenticated Encryption Schemes“. *Information and Communications Security, ICICS '97*, Springer, 1997.

[Berners-Lee 1995] T. Berners-Lee, D. Connolly: „Hypertext Markup Language – 2.0“. *Internet RFC 1866*, November 1995.

[Berners-Lee 1998] T. Berners-Lee, R. Fielding, L. Masinter: „Uniform Resource Identifiers (URI): General Syntax“. *Internet RFC 2396*, August 1998.

[Bleichenbacher 1998] D. Bleichenbacher: „Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1“. *Advances in Cryptology - Crypto '98*, Springer, 1998, S. 1–12.

[Callas 1998] J. Callas, L. Donnerhacke, H. Finney, R. Thayer: „OpenPGP Message Format“. *Internet RFC 2440*, November 1998.

[Chaum 1981] D. Chaum: „Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms“. *Communications of the ACM* 24 (1981) 2, S. 84–88.

[Chaum 1988] D. Chaum: „The Dining Cryptographers Problem. Unconditional Sender and Recipient Untraceability“. *Journal of Cryptology* 1 (1998) 1, S. 65–75.

[Cottrell 1995] L. Cottrell: „Mixmaster and Remailer Attacks“, 1995. URL: `http://www.obscura.com/~loki/remailer/remailer-essay.html`

[Dai 1998] W. Dai: „PipeNet 1.1“, 1998. URL: `http://www.eskimo.com/~weidai/pipenet.txt`

[Demuth 1998] T. Demuth, A. Rieke: „Anonym im World Wide Web?“ *Datenschutz und Datensicherheit* 22 (1998) 11, S. 623–637.

[Eastlake 1994] D. Eastlake, S. Crocker, J. Schiller: „Randomness Recommendations for Security“. *Internet RFC 1750*, Dezember 1994.

[EFF 1998] Electronic Frontier Foundation: *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*, O'Reilly & Associates, 1998.

[Fielding 1997] R. Fielding et al.: „Hypertext Transfer Protocol – HTTP/1.1“. *Internet RFC 2616*, Juni 1999.

[FIPS 1980] US Department of Commerce/National Bureau of Standards: „Data Encryption Standard“. *Federal Information Processing Standards Publication 81*, 1980.

[FIPS 1995] US Department of Commerce/N.I.S.T.: „Secure Hash Standard“. *Federal Information Processing Standards Publication 180-1*, 1995.

[Franz 1997] E. Franz, A. Jerichow, A. Pfitzmann: „Systematisierung und Modellierung von Mixen“. *Verläßliche IT-Systeme, GI-Fachtagung VIS '97*, Vieweg, 1997, S. 172–190.

[Freier 1996] A. Freier, P. Karlton, P. Kocher: „The SSL Protocol Version 3.0“, 4. März 1996. URL: `ftp://ftp.netscape.com/pub/review/ssl-spec.tar.Z`

[Goldberg 1997] I. Goldberg, D. Wagner: „TAZ Servers and the Rewebber Network – Enabling Anonymous Publishing on the World Wide Web“, CS 268 Final Project, University of California, Berkeley, 1997. URL: `http://http.cs.berkeley.edu/~daw/cs268/taz.ps`

[Goldreich 1997] O. Goldreich: „The Foundations of Modern Cryptography“, Version 2.2, 1997. URL: `http://theory.lcs.mit.edu/~oded/tfoc.html`. Vorläufige Fassung in: *Crypto '97*, S. 46–74.

[Goldwasser 1997] S. Goldwasser, M. Bellare: „Lecture Notes on Cryptography“, 19. Juni 1997. URL: `http://www-cse.ucsd.edu/~mihir/papers/gb.ps.gz`

[Gülcü 1996] C. Gülcü, G. Tsudik: „Mixing Email with BABEL". *1996 Symposium on Network and Distributed System Security*, IEEE Computer Society Press, 1996, S. 2–16.

[Johns 1993] M. Johns: „Identification Protocol". *Internet RFC 1413*, Februar 1993.

[Kaliski 1998] B. Kaliski, J. Staddon: „PKCS #1: RSA Cryptography Specifications, Version 2.0". *Internet RFC 2437*, Oktober 1998.

[Kesdogan 1998] D. Kesdogan, J. Egner, R. Büschkes: „Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System". *Information Hiding 1998*, Springer, 1998, S. 83–98.

[Kocher 1996] P. Kocher: „Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". *Advances in Cryptology: Crypto '96*, Springer, 1996, S. 104–113.

[Krawczyk 1997] H. Krawczyk, M. Bellare, R. Canetti: „HMAC: Keyed-Hashing for Message Authentication". *Internet RFC 2104*, Februar 1997.

[Kristol 1997] D. Kristol, L. Montulli: „HTTP State Management Mechanism". *Internet RFC 2109*, Februar 1997.

[Lahey 1999] A. Lahey: „What price privacy?" *Canadian Business*, 26. Februar 1999.

[Martin 1998] D. Martin: „A Framework for Local Anonymity in the Internet". Boston University Computer Science Technical Report 97-002, 1998.

[Menezes 1997] A. Menezes, P. van Oorschot, S. Vanstone: *Handbook of Applied Cryptography*, CRC Press, 1997.

[Micali 1992] S. Micali, P. Rogaway: „Secure Computation". *Advances in Cryptology: Crypto '91*, Springer, 1992, S. 392–404.

[Möller 1998] U. Möller: „Anonymisierung von Internet-Diensten". Studienarbeit, Universität Hamburg, 12. Januar 1998.

[Pfitzmann 1989] A. Pfitzmann, B. Pfitzmann, M. Waidner: „Telefon-MIXe: Schutz der Vermittlungsdaten für zwei 64-kbit/s-Duplexkanäle über den (2*64+16)-kbit/s-Teilnehmeranschluß". *Datenschutz und Datensicherung*, 13 (1989) 12, S. 605–622.

[Pfitzmann 1990] A. Pfitzmann: *Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz*, Springer, 1990.

[Pfitzmann 1990b] B. Pfitzmann, A. Pfitzmann: „How to Break the Direct RSA-Implementation of MIXes“. *Advances in Cryptology: Eurocrypt '89*, Springer, 1990, S. 373–381.

[Rackoff 1993] C. Rackoff, D. Simon: „Cryptographic Defense Against Traffic Analysis“. *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC)*, ACM Press, 1993, S. 672–681.

[Rannenberg 1995] K. Rannenberg et al.: „Mehrseitige Sicherheit als integrale Eigenschaft von Kommunikationstechnik“. H. Kubicek et al. (Hrsg.): *Jahrbuch Telekommunikation und Gesellschaft 1995*, v. Decker, 1995, S. 254–260.

[Reed 1998] M. Reed, P. Syverson, D. Goldschlag: „Anonymous Connections and Onion Routing“. *IEEE Journal on Selected Areas in Communications*, 16 (1998) 4, S. 482–494.

[Reiter 1997] M. Reiter, A. Rubin: „Crowds: Anonymity for Web Transactions“, DIMACS Technical Report 97-15, 1997.

[Schneier 1996] B. Schneier: *Applied Cryptography*, 2. Auflage, Wiley, 1996.

[Stevens 1998] W. R. Stevens: *UNIX Network Programming Volume 1*, 2. Auflage, Prentice Hall, 1998.

[Stroustrup 1997] B. Stroustrup: *The C++ Programming Language*, 3. Auflage, Addison Wesley, 1997.

[Wayner 1996] P. Wayner: *Disappearing Cryptography*, AP Professional, 1996.

[Wiener 1998] M. Wiener: „Performance Comparison of Public-Key Cryptosystems“. *CryptoBytes*, 4 (1998) 1, 1998.

[Wierda 1997] G. Wierda: „Release Notes for version 1.1 of the Squid cache“, 1997.

[Young 1999] E. Young et al.: „OpenSSL“. URL: `http://www.openssl.org`

[ZKS 1999] Zero Knowledge Systems, Inc.: „The Freedom Network Architecture (Version 1.0)“, 1999. URL: `http://www.zks.net/products/Freedom_Architecture.html`

[Zimmerman 1991] D. Zimmerman: „The Finger User Information Protocol". *Internet RFC 1288*, Dezember 1991.

Index

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe.

Hamburg, den 16. Juli 1999 ______________________