INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwurf und Implementierung einer IPv6-Umgebung für mobile Benutzer

Bernd Leinfelder

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwurf und Implementierung einer IPv6-Umgebung für mobile Benutzer

Bernd Leinfelder

Aufgabensteller:	Prof. Dr. Claudia Linnhoff-Popien
Betreuer:	Robert Hofer Dr. Helmut Reiser

Abgabetermin: 31. März 2003

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 31. März 2003

. . . .

(Unterschrift des Kandidaten)

Zusammenfassung

Die Migration vorhandener IPv4-Netze auf IPv6 wird seit langem vorhergesagt, allerdings findet der praktische Einsatz dieser neuen Technologie nur langsam Einzug in die Rechenzentren. In der Praxis wird vielen Problemen, insbesondere der Implementierung von Virtual Private Networks (VPN), vorzugsweise mit mehr oder weniger unvollständigen oder gar proprietären Lösungen begegnet. Viele Hersteller bieten Insellösungen an, die nur mit eigener Hard-/Software funktionieren und den Anforderungen an Interoperabilität nicht genügen. IPv6 bietet dagegen einen vielversprechenden Ansatz, um eine integrierte, sichere und standardisierte Vernetzung entfernter Standorte durchzuführen.

Im Rahmen dieser Diplomarbeit sollen daher aktuelle Implementierungen der IPv6-Protokolle getestet und ein Parallelbetrieb der Dienste am Institut eingerichtet werden, so dass diese Dienste sowohl via IPv4 als auch via IPv6 erreicht werden können. Desweiteren soll ein Konzept für eine mobile IPv6-Umgebung entwickelt, implementiert und bewertet werden, die es Nutzern erlaubt, unabhängig von ihrem Standort die Dienste des Instituts zu nutzen. Gleichzeitig soll es ermöglicht werden, auch vom Institut aus auf Ressourcen in einem entfernten Netz, bspw. Privatrechner zu Hause, zuzugreifen. Dazu ist eine Public Key Infrastruktur (PKI) zu entwickeln, die einerseits hohe Sicherheit als auch einfache Wartung des Systems sowie Verwaltung der Nutzungsberechtigungen gewährleistet. Besondere Aufmerksamkeit soll dabei dem CIP-Pool gelten, der die meisten Mitglieder des Instituts mit Ressourcen versorgt.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung

Bereits 1997 wurde am Institut in einer Diplomarbeit von Hoffmann [Hoff 97] das neue Internetprotokoll IPv6 untersucht. Dabei wurde die grundsätzliche Realisierbarkeit von IPv6-Netzen nachgewiesen sowie auf Änderungen im Netzmanagement durch das neue Protokoll eingegangen. Seitdem sind nun fünf Jahre vergangen, und trotzdem finden sich kaum IPv6-Netze im Produktionseinsatz.

Man stelle sich vor: Hunderte von Ingenieuren arbeiten jahrelang an einem neuen Protokoll, und niemand scheint sich dafür zu interessieren, abgesehen von einigen wenigen Diplomanden. Woran mag das liegen? Offenbar interessieren sich zwar viele Netzwerkadministratoren für IPv6, allerdings fehlt dieser neuen Technik in ihren Augen noch die 'Killer-Applikation', eine neuartige Anwendung, die alle haben wollen und die den Einsatz zwingend erfordert. Solche Anwendungen können in den Augen ihrer ursprünglichen Entwickler Beiwerk sein, während sie sich in den Händen der Anwender aber zum primären Einsatzgebiet der neuen Technologie entwickeln, was niemand hätte vorausahnen können. Ein aktuelles Beispiel für eine solche Killer-Applikation ist der SMS-Boom, der Mobiltelefonie unter Jugendlichen zum Trend des Jahrzehnts hat werden lassen (s. [BuKa 00]).

Dabei ergibt sich mit IPv6 die Möglichkeit, Netzwerkinfrastrukturen zu vereinheitlichen und zu vereinfachen. Die Administration großer Netze lässt sich durch neuartige Konfigurationsmechanismen automatisieren, und neue Anwendungsfelder können durch integrierte Features einfach erschlossen werden. Statt für jede neue Anwendung teure Add-Ons oder spezielle Tools zu installieren, bringt IPv6 viele integrierte Lösungen mit, die einfach nur aktiviert werden müssen. Dadurch vereinfacht sich die Einführung neuer Dienste und das Management wird vereinheitlicht.

Als Beispiel für solche neuen Dienste wird im Rahmen dieser Arbeit näher auf die Anbindung mobiler Nutzer eingegangen. Mitarbeiter von wissenschaftlichen Einrichtungen, Angestellte, Studenten sind viel unterwegs, wobei ihnen mittlerweile regelmäßig ein eigenes Notebook zur Verfügung steht. Auch in privaten Arbeitszimmern steht mittlerweile leistungsfähiges IT-Equipment, mit dessen Hilfe Daten zwischen dem aktuellen Standort und dem Universitäts-/Firmennetz ausgetauscht werden. Oftmals ist dieser Datenaustausch aber kompliziert, wenn nicht gar ganz unmöglich, weil er bspw. aufgrund von Sicherheitsrichtlinien unterbunden wird. Ebenso ist der Zugriff vom Arbeitsplatz auf die Daten, die auf privaten Rechnern gespeichert sind, oft gar nicht oder nur unter erschwerten Bedingungen möglich. Grund dafür sind einerseits die schon erwähnten Sicherheitsvorkehrungen, andererseits aber auch technische Unzulänglichkeiten:

- Die meisten Nutzer erhalten Zugriff zum Internet mit Hilfe einer Dial-Up-Verbindung. Bei der Einwahl wird die IP-Adresse dem Nutzer üblicherweise dynamisch zugewiesen und ist damit a priori unbekannt. Daher sind besondere Vorkehrungen nötig, um die IP-Adresse des privaten Rechners von einem entfernten Standort aus feststellen zu können, um eine Verbindung zu diesem Rechner herstellen zu können.

- Falls ein Nutzer nicht nur einen einzelnen Rechner, sondern ein ganzes Netz privat betreibt, wachsen die Probleme beim Zugriff von einem entfernen Standort aus erheblich: Für Privatleute ist es sehr teuer, mehr als eine offizielle IP-Adresse für ihre eigenen Rechner zu erhalten. In der Regel kommt nur die Anbindung über einen Dial-Up-Zugang mit einer einzigen, dynamisch zugewiesenen Adresse in Frage. Der Zugriff von außen auf ein derart angebundenes Netz muss daher über Proxy-Systeme erfolgen, die kompliziert einzurichten und zu warten sind.

Dies sind nur zwei der Probleme, die der Einsatz von IPv6 umgehen kann: Denn es wurde unter anderem wegen der Adressknappheit von IPv4 entwickelt. Mit IPv6 kann jedem Internetnutzer daher ein nahezu unerschöpflicher Adressraum zugewiesen werden, mit dessen Hilfe die obigen Probleme gar nicht erst auftreten.

Natürlich sind die Probleme von Privatanwendern kein ausreichender Grund, von einer bewährten und gut eingeführten Technologie wie IPv4 zu migrieren. Aber auch global betrachtet sind die Gründe für eine Migration bestechend: Viele Schwellenländer stecken derzeit große Ressourcen in die Verbesserung der nationalen IT-Infrastruktur. Ein Aspekt dabei ist der Anschluss breiter Bevölkerungsschichten an das Internet. So haben nach [Cent 02] derzeit 45,8 Mio. Chinesen Zugriff auf das weltweite Netzwerk.

Eine Fortsetzung des rasanten Wachstums des Internets in Asien ist abzusehen. Für alle diese Benutzer werden natürlich IP-Adresse benötigt. Ebenso wird in den hoch entwickelten Industriestaaten durch die Entwicklung neuer Technologien wie UMTS der Bedarf an Adressen in den nächsten Jahre enorm wachsen. Es ist daher sehr wahrscheinlich, dass für diese Zwecke der Adressraum von IPv4 definitiv nicht mehr ausreichen und damit der Druck, IPv6 einzuführen, steigen wird.

Diese globalen Entwicklungen können gleichzeitig von einzelnen Organisationen wie dem Institut, die selbst nicht direkt von dem Problem der Adressknappheit betroffen sind, genutzt werden, um ihren Anwendern neue Dienste zur Verfügung zu stellen. Einer dieser Dienste könnte die Delegation gerouteter Adressräume an Endanwender wie Mitarbeiter oder Studenten sein, denen dadurch die Möglichkeit gegeben wird, ihre private IT-Infrastruktur besser an die Dienste des Instituts anzubinden und so produktiver zu arbeiten.

1.1 Die Gründe für IPv6

IPv4 spezifiziert einen Adressraum der Größe 32 Bit, was $2^{32} = 4,3 * 10^9$ verschiedenen Adressen entspricht. Vor mehr als 20 Jahren, als IPv4 eingeführt wurde, war das mehr als ausreichend. Adressen wurden im folgenden großzügig an Organisationen und Firmen vergeben. Aus diesem Grund ist nach wie vor ein Teil der vergebenen Adressen ungenutzt und damit für den Netzaufbau vergeudet.

Im Gegensatz zum streng hierarchischen Telefonnetz wurden außerdem Adressen ohne Rücksicht auf die geographische Lage oder sonstige Gruppenmerkmale der Organisationen vergeben (flaches Adressierungsschema), mit der Folge einer Zersplitterung des Adressbereichs und Anwachsen der Routingtabellen [Ditt 02]. Durch die Einführung des World-Wide-Webs kam es in den 90er Jahren des letzten Jahrhunderts zu einer explosionsartigen Vergrößerung des Internet. Schon Ende 1990 kam es daher zu ersten Überlegungen, eine neue Protokollgeneration im Internet einzusetzen.

Nach [Ditt 02] wurden dabei folgende Entwicklungsziele festgelegt:

- neuer und damit großer Adressraum
- einfacher und damit schnell auswertbarer Paketaufbau
- Gruppierung von Adressen und damit effizienteres Routing aufgrund kompakterer Routingtabellen

1.1.1 Neuer Adressraum

Das neue IPv6 [1] bekam daher einen Adressraum von 128 Bit, was theoretisch $2^{128} = 3,4 * 10^{38}$ Adressen ermöglicht. Laut Huitema [Huit 00] ging man dabei von einer geschätzten Bevölkerung der Erde im Jahr 2020 von 10 Milliarden Menschen aus, von denen jeder im Durchschnitt 100 Computer besitzt. Geht man davon aus, das in der Zukunft von Haushaltsgeräten bis Servomotoren in Autos praktisch alle elektronischen Geräte an ein Netzwerk konnektiert sein werden, scheint diese Zahl nicht übertrieben. Die Zielwerte für das neue Protokoll wurden daher auf eine Billiarde (10^{15}) Rechner festgelegt, die über eine Billion (10^{12}) Netze miteinander verbunden sind.

Huitema schlug weiterhin vor, in die Betrachtungen die Effizienz der Adressvergabe zu berücksichtigen, da eine 100prozentige Ausnutzung des Adressraumes aufgrund der Struktur von Netzwerken nicht möglich ist. Als Maßzahl für diese Effizienz bestimmte er die H-Verhältniszahl:

$$H = \frac{log_{10}(AnzahlAdressen)}{AnzahlAdressbits} \tag{1.1}$$

Die logarithmische Natur dieses Quotienten versucht dabei, die inhärente Ineffizienz hierarchischer Netzwerkstrukturen zu berücksichtigen. In einem Netzwerk mit perfekt ausgenutztem Adressraum von n Bit Größe ergibt sich damit ein H-Wert von

$$H_{ideal} = \frac{log_{10}(2^n)}{n} = \frac{n * log_{10}(2)}{n} = log_{10}(2) = 0,30103 \tag{1.2}$$

Praktische Beobachtungen in realen Netzen zeigen, dass in der Praxis derzeit ein H-Wert von 0,22 bis 0,26 erreicht werden kann. Damit reichen die 32-Bit-Adressen von IPv4 für 11 bis 200 Millionen Rechner aus, was derzeit genügen mag, aber für die zukünftige Entwicklung kaum Spielraum lässt.

Mit [DuHu 01] wurde eine alternative Berechnungsweise vorgestellt, die einen intuitiveren Zugang zum ermittelten Wert bieten soll, indem der H-Wert auf den Bereich $[0, 1]$ normiert wird:

$$HD = \frac{log(AnzahlzugewiesenerObjekte)}{log(MaximaleAnzahlzugewiesenerObjekte)} \tag{1.3}$$

In den meisten Fällen kann HD aus H berechnet werden:

$$HD = \frac{H}{log_{10}(2)} \tag{1.4}$$

Gruppierung von Adressen

Um das Problem der wuchernden Routingtabellen in den Griff zu bekommen, war außerdem ein Mechanismus gefordert, um diese zu vereinfachen. Nach welchen Gesichtspunkten diese Gruppierung stattfinden soll, wird derzeit noch diskutiert [Ditt 02]. Grundsätzlich sollen Adressen hierarchisch vergeben werden, wobei noch unklar ist, wie die einzelnen Ebenen der Hierarchie gegeneinander abgegrenzt werden sollen. Der ursprüngliche Vorschlag einer Gliederung nach geographischen Prinzipien wurde inzwischen aufgegeben, da hinsichtlich vieler supernational arbeitender Organisationen dies nicht sinnvoll erschien.

Vereinfachung der Adressverwaltung

Während die Netzwerkkonfiguration von Endgeräten bei IPv4 überwiegend manuell erfolgt, wird bei Verwendung von IPv6 ein großer Teil dieser Tätigkeit automatisiert. Dadurch entfällt der aufwendigste und fehleranfälligste Teil der Konfigurationsarbeiten, der Administrator kann sich mehr auf die Planung und Wartung des Netzwerks konzentrieren. Die automatischen Mechanismen von IPv6 sind in das Protokoll integriert, sie sind daher selbst wartungsarm und können in vielen Fällen einen dedizierten Konfigurationsdienst, wie den heute eingesetzten DHCP-Service,

[1] Zu Beginn der Standardisierung trug das Protokoll noch den Namen IPng für Next Generation IP.

ersetzen, da die am häufigsten genutzten Funktionalitäten von DHCP von den Mechanismen zur Autokonfiguration wahrgenommen werden.

Diese Mechanismen unterstützen außerdem automatisches netzweites Renumbering, so dass ein Netzwerk problemlos und schnell in ein anderes Netzwerksegment verlagert werden kann.

1.1.2 Vereinfachung des Headers

Im Laufe der Zeit hat sich gezeigt, dass das Headerformat von IPv4 Informationen enthält, die für viele Pakete nicht benötigt werden. Als Beispiel seien die Fragmentierungsfelder genannt, die naturgemäß nur dann genutzt werden, wenn ein Paket fragmentiert werden muss. In allen anderen Paketen sind die verwendeten vier Byte überflüssig, und ihre Auswertung in den Routern verschwendet Rechenleistung. Daher wurde in IPv6 der Headeraufbau vereinfacht, alle nicht zwingend erforderlichen Informationen sind in Erweiterungsheader ausgelagert worden, um so die Ressourcennutzung zu minimieren.

1.1.3 Integration bestehender Erweiterungen für IPv4

Im Laufe der letzten 20 Jahre wurden neue Anforderungen an das Internetprotokoll gestellt, die mit Hilfe von Erweiterungen realisiert wurden. Viele davon sind bei IPv6 bereits integriert, so dass sich eine Reduzierung des Administrationsaufwandes ergibt. Wenn in einem Netzwerk eine dieser Erweiterungen benötigt wird, muss die nötige Software anders als bei IPv4 nicht separat installiert und konfiguriert werden, sondern kann einfach verwendet werden. Zusätzliche Konfigurationseinstellungen können mit Hilfe vorhandener Managementeinrichtungen vorgenommen werden.

Protokollintegrierte Sicherheit

Ein großes Plus bei IPv6 sind die direkt ins Protokoll integrierten Sicherheitsmechanismen. Bei IPv4, wo zum Zeitpunkt der Standardisierung Sicherheitsprobleme in keinster Weise beachtet wurden, mussten diese Features im nachhinein durch das Zusatzprotokoll IPSec ([KeAt 98c]) nachgerüstet werden.

Vor Verwendung müssen sie daher üblicherweise erst separat installiert und unabhängig vom IP-Stack konfiguriert werden. Durch die Integration in das Protokoll selbst bei IPv6 vereinfacht sich die Implementierung von Sicherheit, die Standardisierung führt zu erhöhter Interoperabilität zwischen heterogenen IP-Implementierungen, da die Nutzung proprietärer Sicherheitssoftware an Attraktivität verliert, wenn eine solche Software bereits im IP-Stack integriert ist.

Die Sicherheitsmechanismen in IPv6 bieten einerseits die Möglichkeit zur Verschlüsselung, andererseits zur Authentifizierung von Datenverkehr. Gerade bei der Bereitstellung von Diensten zum Remote-Zugriff auf das Netzwerk des Instituts sind diese beiden Features unverzichtbar. Verschlüsselung dient hier zur Herstellung der Vertraulichkeit der übertragenen Daten, während Authentifizierung den nötigen Zugriffsschutz sowie die Möglichkeit zur Identifikation von Remote-Nutzern bietet.

Multicasting

Ähnlich der Sicherheitsproblematik war bei der Spezifikation von IPv4 vor 20 Jahren kein Bedarf nach Multicastingfähigkeiten. Gerade mit dem Aufkommen von Multimedia-Streaming wurde ein Weg gesucht, um das übertragene Datenvolumen zu reduzieren. 1989 wurden dazu Multicasting-Erweiterungen für IPv4 definiert ([Deer 89]), die aber insbesondere in der Backbone-Infrastruktur überwiegend nur auf experimenteller Ebene realisiert wurden (MBone). Praktische Bedeutung konnte diese Erweiterung nie erlangen, was zur massiven Verschwendung von Netzwerkressourcen bei Streaming-Anwendungen führt.

Angesichts der offensichtlichen Nützlichkeit von Multicasting wurden auch diese Features in die Spezifikation von IPv6 integriert. Damit werden Backbone-Router bei Implementierung von IPv6 automatisch Multicast-fähig, ohne dass deren Administratoren besonderen Installations- und Konfigurationsaufwand betreiben müssen.

Da auch im universitären Umfeld Streaming-Anwendungen, bspw. zur Übertragung von Lehrveranstaltungen, verwendet werden, kann Multicasting hier zu einer effizienteren Ressourcennutzung beitragen.

Definierte Quality-of-Service

Quality-of-Service (QoS) bezeichnet die Fähigkeit eines Netzwerks, die übertragenen Daten in verschiedene Klassen einzuteilen und sie unterschiedlich zu behandeln ([Hust 00]). Dies kann dazu benutzt werden, um Datenverkehr unterschiedlich zu priorisieren, um dadurch schnelleren Transport oder geringeren Paketverlust zu realisieren oder bestimmten Klassen eine garantierte Bandbreite zur Verfügung zu stellen. Dies führt zu einer höheren Übertragungsqualität und besserer Ressourcennutzung, insbesondere für zeitkritischen Datenverkehr. So können die Daten einer interaktiven Anwendung mit geringer Latenz, Daten einer Streaming-Anwendung jedoch mit maximalem Durchsatz transportiert werden.

In IPv4 war eine solche Priorisierung zwar vorgesehen, jedoch nicht genau spezifiziert und somit nicht verwendbar, da jeder Routerhersteller eine andere Vorstellung von der Semantik der für QoS vorgesehenen Parameter hatte. Diese Schwäche wurde bei der Spezifikation von IPv6 beseitigt, so dass die Implementierung von IPv6 gleichzeitig die Verwendbarkeit von QoS-Features impliziert.

1.2 Aufgabenstellung

Einrichtung eines Parallelbetriebs

Bewusst wurde in den Standardisierungsgremien wert darauf gelegt, dass der Übergang von IPv4 zu IPv6 fließend erfolgen kann, da die globale Umstellung zu einem festgelegten Zeitpunkt aufgrund der Größe des Internets, der Vielfalt der eingesetzten Endgeräte und der Unübersichtlichkeit der Verantwortlichkeiten völlig utopisch ist. Für lange Zeit werden also IPv4- und IPv6-basierte Netze nebeneinander existieren.

Um diesen Parallelbetrieb von IPv4 und IPv6 zu testen, ist für das Rechnernetz des Institut für Informatik ist ein Konzept zu dessen Realisierung zu entwickeln. Dabei sind die unterschiedlichen Mechanismen, die dafür in Frage kommen, zu beschreiben und basierend auf den Anforderungen des Instituts zu bewerten. Es ist sicherzustellen, dass dieser Parallelbetrieb die vorhandene IPv4-Infrastruktur nicht beeinträchtigt. Dazu sind die betriebsnotwendigen Dienste zu identifizieren und auf ihre IPv6- Fähigkeit zu untersuchen.

Mobilität

Desweiteren ist die Möglichkeit zu schaffen, von außen als mobiler Benutzer auf die Ressourcen des Instituts zuzugreifen. Mobil im Sinne dieser Arbeit sind alle Benutzer, an deren physischen Aufenthaltsort keine weiteren Bedingungen geknüpft sind als die Zugriffsmöglichkeit auf ein IPv4-fähiges Netzwerk. Darunter fallen folgende Personengruppen:

- Benutzer, die unter Verwendung ihres privaten Rechners von zuhause aus die Ressourcen des Instituts nutzen wollen.
- Benutzer, die ihren heimischen Rechner an das Netz des Instituts anschließen wollen.
- Benutzer, die von ihrem Arbeitsplatz im Institut Daten mit ihrem heimischen Rechner austauschen wollen.
- Benutzer auf Reisen, die unter Verwendung ihres Notebooks auf das Institutsnetz zugreifen wollen.

Sicherheit

Ein zentrales Thema dieser Arbeit ist die Gewährleistung der notwendigen Sicherheit beim Zugriff vom/zum Institutsnetz. Die integrierten Sicherheitsmechanismen heben IPv6 deutlich von seinem Vorgängerprotokoll ab. Diese Mechanismen sind zu beschreiben und im Rahmen der Anforderungen zu bewerten. Dabei sollen Angriffsszenarien entwickelt und die Robustheit des Protokolls gegenüber diesen Szenarien diskutiert werden.

Administration

Beim Betrieb großer Netze spielt jedoch nicht nur die Sicherheit der eingesetzten Produkte, sondern auch der Aufwand zu deren Verwaltung eine wichtige Rolle. Da die Sicherheitsmechanismen allesamt auf kryptographischen Verfahren basieren, kann die Erzeugung und Verwaltung der Schlüssel einen nicht zu unterschätzenden Aufwand verursachen. Deshalb ist ein Konzept für eine Schlüsselverwaltung zu entwickeln, die einerseits den hohen Anforderungen an die Sicherheit genügt, andererseits aber auch im Rahmen der dem Institut zur Verfügung stehenden Mittel bleibt. Die zentralen Anforderungen an eine solche Schlüsselverwaltung sind also Sicherheit und Wartungsarmut. Sicherheit bedeutet, dass eine Kompromittierung der hinterlegten Schlüssel durch unbefugte Personen oder Angreifer soweit möglich ausgeschlossen sein muss. Wartungsarmut meint, unter Zuhilfenahme bereits vorhandener Sicherheitsmechanismen den Aufwand manueller Administrationsvorgänge zu minimieren.

Evaluation neuerer Implementierungen für relevante Plattformen

Gerade durch den Boom des Internet in Asien entstehen derzeit vor allem in Japan neuere Implementierungen des IPv6-Protokolls. Während die in aktuellen Linux-Standardkerneln vorhandene Implementierung zum einen nicht vollständig, zum anderen gegenüber den aktuellen IETF-Spezifikation veraltet ist, bieten neuere Implementierungen eine mittlerweile nahezu vollständige Implementierung an.

Auch die Hersteller der anderen großen Unix-Systeme wie HP und Sun verbessern laufend die Implementierung von IPv6 in ihren Betriebssystemen. Microsoft hat bereits angekündigt, eine IPv6-Implementierung in das kommende Windows aufzunehmen. Eine ähnliche Entwicklung zeigt sich bei den Routerherstellern wie Cisco. Diese Arbeit soll sich daher auch mit Implementierungen des neuen Protokollstacks beschäftigen und deren Produktionsreife beurteilen.

1.3 Aktuelle Zugriffsmöglichkeiten zum IFI-Netz

Abhängig von Standort des Benutzers gibt es derzeit unterschiedliche Einschränkungen, was den Zugriff auf das Netz des Instituts angeht. Befindet sich der Nutzer innerhalb des Münchner Wissenschaftsnetzes (MWN),so erhält er Zugriff auf fast alle angebotenen Dienste. Befindet er sich jedoch außerhalb des MWN, so verhindern Firewall-Regeln den Zugriff auf die meisten Dienste. Der Nutzer kann sich dann nur per SSH an einen dedizierten Zugangscluster anmelden, und von dort aus bei Bedarf weitere Dienste ansteuern. Diese Vorgehensweise wurde notwendig, da mehrmals von außerhalb des MWN Angriffe auf das Institutsnetz durchgeführt wurden, die in einigen Fällen auch erfolgreich waren. Wegen der Vielzahl der sicherheitsrelevanten Softwarefehler, die in einem derart komplexen Netzwerk zwangsläufig auftreten, schien diese sehr restriktive Vorgehensweise geboten, da selbst bei genauer und sorgfältiger Durchführung aller Softwareupdates nicht ausgeschlossen werden kann, dass ein noch unbekannter Fehler zu einem Sicherheitsleck führt.

1.4 Aufbau der Arbeit

Kapitel 2 beschäftigt sich mit den Anforderungen bezüglich einer Protokollmigration im Institut. Dabei werden technische Anforderungen, Managementanforderungen, organisatorische Anforderungen sowie Sicherheitsanforderungen dargestellt und der Aufwand zu deren Realisierung abgeschätzt.

Kapitel 3 definiert grundlegende Begriffe und führt in das neue Protokoll IPv6 ein. Insbesondere wird hier auf Unterschiede zum aktuellen Protokoll IPv4 eingegangen. Desweiteren werden Konfigurations- und Migrationsmechanismen vorgestellt, wie sie vom IETF vorgeschlagen wurden. Ebenso wird auf die besonderen Features von Mobile-IPv6 eingegangen, ein Protokoll, dass speziell für ortsunabhängige Internetnutzung entwickelt wurde.

Kapitel 4 zeigt, wie Sicherheitsanforderungen durch die Mechanismen von IPv6 erfüllt werden können und führt in die grundlegenden Konzepte ein.

Kapitel 5 konzipiert eine Lösung, die den Anforderungen aus Kapitel 2 entspricht. Daneben wird eine Public Key Infrastructure entsprechend den Anforderungen des Institut entwickelt.

Kapitel 6 zeigt, wie die in Kapitel 5 entwickelte Lösung in der Praxis umgesetzt werden kann.

Kapitel 7 enthält eine Zusammenfassung der Arbeit und gibt einen Ausblick auf zukünftige Entwicklungen.

Kapitel 2

Anforderungsanalyse

Das folgende Kapitel beschäftigt sich mit den konkreten Anforderungen des Instituts für Informatik an eine IPv6-Infrastruktur. Basis für jede Einführung neuer Technologien in Netzwerken muss daher immer die Analyse im konkreten Fall sein, die sich mit folgenden Aspekten befasst:

- Technische Anforderungen: Welche der gewünschten Funktionalitäten ist mit der vorhandenen technische Ausstattung und der zur Verfügung stehenden Hard- und Software überhaupt leistbar?
- Anforderungen der Benutzer: Welche Funktionalitäten sind bereits implementiert und sollen auch nach der Einführung noch vorhanden sein? Welche Funktionalitäten sollen zusätzlich implementiert werden?
- Managementanforderungen: Hier ist zu fragen, welchen zusätzlichen Aufwand die Implementierung für die Administratoren mit sich bringt, ob dieser Aufwand mit der personellen Ausstattung überhaupt geleistet werden kann, ob das zuständige Personal das nötige Know-How hat und inwieweit bereits vorhandene administrative Prozesse auch nach der Einführung noch anwendbar sind. Ebenfalls abzuschätzen sind die Kosten, die dadurch während des Betriebs entstehen.
- Organisatorische Anforderungen: Oft werden von der Leitung von Organisation Vorgaben gemacht, die sich durch keine der anderen Anforderungen begründen lassen. Hier wäre bspw. die Präferierung eines bestimmten Herstellers aus vertraglichen Gründen einzuordnen oder die Beschneidung von Funktionalitäten wegen rechtlicher oder finanzieller Probleme.
- Sicherheitsanforderungen: Wie ist sicherzustellen, dass die Migration die Sicherheit des Netzwerkes des Instituts als auch der übertragenen Daten gewährleistet?
- Aufwand der Implementierung: Die Einführung neuer Technologien bedingt Kosten, die sich aus den Kosten für das Personal sowie Kosten für technische Geräte zusammensetzen. Es ist daher zu klären, wie hoch diese Aufwände sind.

Im folgenden sollen diese Anforderungen dargestellt und analysiert werden. Dies wird die Basis für alle weiteren Abschnitte dieser Arbeit sein, in denen dann versucht werden soll, diesen Punkten gerecht zu werden.

2.1 Technische Anforderungen

Bei den Endgeräten ist zu ermitteln, welche davon IPv6-fähig sind und damit migriert werden können. Neuanschaffungen sollten aus Kostengründen vermieden werden. Bei vorhandenen IPv6-Implementierungen ist ihre Kompatibilität zum Standard und der Stand und Status dieser Implementierungen zu klären.

Zur Minimierung des Aufwands soll die Einführung von IPv6 so weit wie möglich auf der bereits vorhandenen Hard- und Software stattfinden. Als Vorgaben sind hier folgende Eigenschaften des Institutsnetzes einzuhalten:

- Betriebssysteme: Der größte Teil der im Institut eingesetzten Server und Workstations wird mit einen Linux-Kernel der Version 2.4 betrieben. Außerdem ist an den einzelnen Lehrstühlen sowie im studentischen CIP-Pool eine Vielzahl weiterer Betriebssysteme im Einsatz, deren IPv6-Tauglichkeit zu prüfen ist. Ein Austausch der Betriebssysteme ist nicht möglich, um die Fülle der darauf installierten Anwendungen weiterhin bereitstellen zu können. Jedoch können die Systeme um Patches erweitert werden, um sie IPv6-tauglich zu machen.

  Um Betriebsunterbrechungen zu vermeiden, sind Änderungen an den eingesetzten hardwarebasierten Routern nicht möglich.

- Es steht derzeit kein IPv6-fähiger Uplink im Institut zur Verfügung. Hier ist eine alternative Anbindung erforderlich.

2.2 Anforderung der Benutzer

Aus Benutzersicht ist der beste Administrator derjenige, vom dem man gar nicht weiß, dass es ihn gibt. Denn Benutzer wollen ihre Arbeit erledigen und keinen Fehlern nachspüren, um sie von einem Administrator beheben zu lassen. Daher sollten bestehende Funktionalitäten transparent erhalten werden, Benutzer ohne Interesse an Netzwerktechnik im allgemeinen und IPv6 im besonderen sollen gar nicht merken, dass an ihrem Rechner sich irgendetwas verändert hat.

Daher gilt es, zuerst die wichtigsten Benutzerdienste zu identifizieren; bei der Migration muss sichergestellt werden, dass sie sowohl während der Migrationsphase als auch nach vollendeter Migration ohne Einschränkung erhalten bleiben.

- Zuallererst sind alle Dienste zu nennen, die keine Netzwerkfunktionalität bieten: Das sind bspw. Editoren, Programme zur Bildbearbeitung und ähnliches. Da ihre Funktion in keiner Weise vom Netzwerkparametern abhängt, müssen sie im weiteren nicht mehr betrachtet werden. Die Migration hat keinerlei Auswirkungen auf sie.

- Eine besondere Klasse von Anwendungen sind Tools zur Softwareentwicklung: Sie selbst nutzen keine Netzwerkfunktionen, womit sie eigentlich zu vorhergehenden Klasse gehören. Allerdings können mit ihnen Programme erstellt werden, die das tun. Es ist daher sicherzustellen, dass alle Bestandteile der installierten Compiler in der Lage sind, Programme mit IPv6-Funktionalität zu erstellen.

- Eine weitere Klasse bilden alle Netzwerkanwendungen, die als Teil der Benutzeroberfläche direkt mit dem Benutzer interagieren. Zu den am häufigsten genutzten Anwendungen dieser Art zählen:

  - HTTP-Clients: Die Webbrowser mausern sich immer mehr zum zentralen Interface der Informationsbeschaffung. Eine Migration, die den verbreiteten HTTP-Clients den Zugang zum Internet verwehrt, ist damit völlig undenkbar.
  - Email-Clients: Natürlich muss auch der Zugang zu einem der wichtigsten Kommunikationsmittel erhalten bleiben.
  - Multimedia-Anwendungen: Ein immer wichtigeres Thema sind Anwendungen, die Audio- und Videodaten aus dem Internet laden und darstellen.
  - NNTP: Es wird zwar immer weniger benutzt, aber gerade im universitären Umfeld ist der Zugang zu Newssservern unverzichtbar.
  - ...

- Zentral für jede Nutzerinteraktion sind die Daten des Users. Diese werden durch Fileserver bereitgestellt, im Institut für Informatik vor allem mit Hilfe Suns Network Filesystem (NFS), aber auch durch das SMB-Protokoll vom Microsoft. Es ist zu klären, wie weit die IPv6-Unterstützung in dieser Software bereits gediehen ist.
- Vor allem für diverse Lehrveranstaltungen, wie z. B. das Datenbankpraktikum, wird Zugriff auf diverse Datenbanken benötigt. Es ist daher zu überprüfen, ob die genutzten Datenbanken und die dazugehörigen Treiber in den verschiedenen Programmiersprachen auch über IPv6 genutzt werden können.
- Der Zugriff auf Remote-Systemen soll sowohl wie bisher auf Rechner in IPv4-Netzen als für Rechner in IPv6-Infrastrukturen möglich sein. Es ist also festzustellen, ob die installierte SSH-Version Zugriff via IPv6 auf entfernte Systeme erlaubt.

Zusammenfassend lautet die primäre Anforderung der Benutzer, dass alle genutzten Dienste auch nach Einrichtung des Parallelbetriebs uneingeschränkt und in gewohnter Weise funktionieren müssen.

Mobile Nutzung

Ein wichtiges Ziel dieser Arbeit ist die Möglichkeit zur Nutzung von entfernten Ressourcen durch mobile Benutzer. Ressourcen können dabei sowohl Dienste des Instituts sein, als auch Dienste, die von den Benutzern selbst oder Dritten bereitgestellt werden. Benutzer sind alle autorisierten Personen, also im wesentlichen Mitarbeiter und Studenten. Die Nutzung der Ressourcen soll dabei unabhängig vom physischen Standort der Ressourcen ebenso wie dem der Benutzer stattfinden können. Dabei sind bspw. folgende Szenarien denkbar:

- Ein Mitarbeiter möchte zu Hause an seiner Promotion weiterarbeiten. Dazu benötigt er Zugriff auf die Daten, die sich auf einem Dateiserver seines Lehrstuhls befinden. Der Internetzugang seines Privatrechner wird über einen DSL-Anschluss mit dynamischer Adressvergabe vermittelt.
- Derselbe Mitarbeiter fährt einige Tage später zu einem Kongress. Dort wird er spontan gebeten, einen kurzen Vortrag über sein Forschungsgebiet zu halten. Um die aktuellsten Daten zu verwenden, muss er sich mit einem Analog-Modem von seinem Hotel aus ins Internet einwählen, um Zugriff zum Datenbankserver seines Lehrstuhls zu erhalten.
- Zurück in seinem Büro muss er eine Spesenabrechnung erstellen. Die dazu nötigen Belege hat er bereits zu Hause sortiert und in eine Tabellenkalkulation eingegeben. Diese Datei möchte er sich nun von seinem Rechner zu Hause auf seinen Bürorechner kopieren.
- Um sich über die Ergebnisse seiner Forschung einfacher mit Kollegen außerhalb des Instituts austauschen zu können, möchte der Mitarbeiter bei sich zu Hause einen Webserver betreiben. Die Konnektivität soll über eine feste IP-Adresse hergestellt werden.
- Ein halbes Jahr danach fährt der Mitarbeiter zu einem Forschungsaufenthalt in die USA. Um dort nicht mehrere Tage Arbeit damit zuzubringen, die ihm vertraute Infrastruktur seines Lehrstuhls nachzubauen, damit er weiterforschen kann, sollen die Dienste transparent in das Netz an seinem neuen Standort eingebunden werden.

Diese Beispiele sollen verdeutlichen, wie flexibel das zu planende System in unterschiedlichen Situationen eingesetzt werden soll (s. a. Abb. 2.1).

Ein weiteres Szenario ergibt aus dem Betrieb der X-Terminals, die im Gebäude der Mathematischen Fakultät in der Theresienstraße betrieben werden. Diese Terminals bieten Mitarbeitern und Studenten des Instituts die Möglichkeit, Dienste des Instituts auch dann zu nutzen, wenn sie sich in der Theresienstraße, also außerhalb des Institutsnetzes aufhalten. Diese Terminals sind derzeit mit Hilfe von IPSec-Tunnels an das Institutsnetz angebunden.

Aus diesen Szenarien ergeben sich folgende Anforderungen:

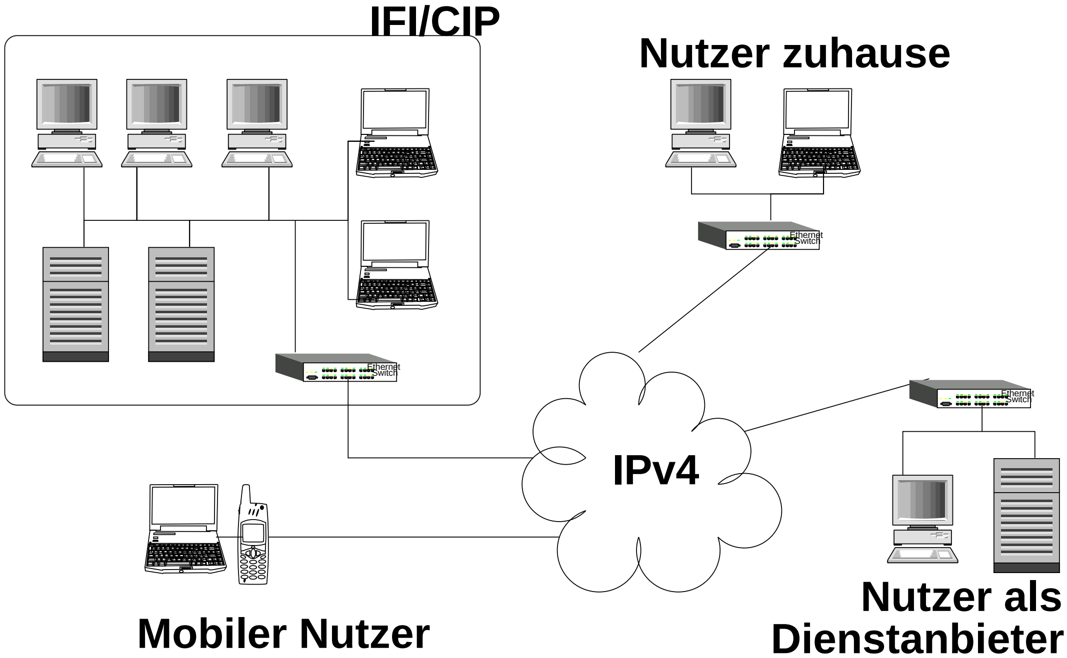

Abbildung 2.1: Mobile Nutzung von Ressourcen des Instituts

- Zugang zum Institutsnetz: Der Benutzer muss eine Möglichkeit erhalten, von einem entfernten Standort aus auf Dienste im Institut zuzugreifen.
- Sicherheit: Um Missbrauch zu verhindern, muss dieser Zugriff sicher authentifiziert und autorisiert werden.
- Vertraulichkeit: Um Vertraulichkeit zu gewährleisten, muss der Datenverkehr zwischen Nutzer und Institut verschlüsselt werden.
- Nutzer als Dienstanbieter: Um selbst mit seinem privaten Rechner Dienste anbieten zu können, muss der Nutzer mit mindestens einer, bei Bedarf auch mehreren festen IP-Adressen ausgestattet werden.

2.3 Managementanforderungen

Grundlegend verschieden von den Anforderungen der Benutzer sind die Anforderungen der Administratoren an die Einführung neuer Dienste in komplexen vernetzten Systeme.

Solche Systeme wie die IT-Infrastruktur des Instituts müssen einem möglichst vollständigen Management unterworfen werden. Nach [HAN 99] versteht man darunter "alle Maßnahmen für einen unternehmenszielorientierten effektiven und effizienten Betrieb eines verteilten Systems mit seinen Ressourcen". Ziel des Management ist es also, Dienste bereitzustellen und ihre Verfügbarkeit zu sichern. Grundlegende Konzepte und Begriffe des Managements sind im ISO-OSI-Referenzmodell beschrieben, das auf der Kooperation offener Systeme basiert. Aufgrund der Heterogenität des Institutsnetzes ist dieser Ansatz proprietären Architekturen vorzuziehen. Ausgehend vom ISO-OSI-Referenzmodell wurden verschiedene Modelle entwickelt, wobei das sog. Funktionsmodell hier näher ausgeführt werden soll.

2.3.1 Funktionsmodell

Der beim Funktionsmodell benutzte Ansatz ist, die Gesamtheit der Managementfunktionalität in einzelne Funktionsbereiche zu gliedern und generische Managementfunktionen für jeden dieser Bereiche zu festzulegen. Dazu werden die erwartete Funktionalität, die Dienste selbst sowie geeignete Managementobjekte spezifiziert. Dadurch kann die Funktionalität als Basis für eine Managementbibliothek verwendet werden, einen Baukasten alle Aufgaben des Managements. Grundlage der Managementfunktionen ist dabei das dazugehörige Informationsmodell. Das Design der einzelnen Funktionen ist dabei essentiell für die Effizienz des Managements großer Netze. Nur wenn die Funktionalitäten ausreichend umfangreich und flexibel sind, ist die benötigte Skalierbarkeit der Lösung zu erreichen.

Die Entwicklung einzelner Funktionsbausteine folgt dabei dem Prinzip "stepwise refinement". Zu konkreten Managementaspekten wird versucht, möglichst umfassende Modelle, zum Beispiel eine Zustandsmodell, zu entwickeln, das auf alle Managementobjekte angewendet werden kann.

2.3.2 Managementbereiche

Um die einzelnen Funktionen des Funktionsmodells bereichsspezifisch zu klassifizieren, hat es sich durchgesetzt, es in fünf Funktionsbereiche aufzuteilen: Fehlermanagement, Konfigurationsmanagement, Leistungsmanagement, Abrechnungsmanagement und Sicherheitsmanagement. Diese Bereiche werden nach den Anfangsbuchstaben der englischen Begriffe oft auch als FCAPS bezeichnet.

Um in heterogenen Systemen Bestand zu haben, muss die Beschreibung der einzelnen Funktionsbereiche in offener Weise geschehen. Dabei gelten diese Funktionsbereiche für alle Objekttypen. Im folgenden sollen nun typische Managementaufgaben der einzelnen Bereiche dargestellt werden.

Grundsätzlich ist darauf zu achten, dass die Managementressourcen im Institut stark beschränkt sind und daher der zusätzliche Administrationsaufwand, der durch die Einführung des neuen Protokolls und neuer Dienste zu leisten ist, minimiert werden soll.

Fehlermanagement

Dieser Bereich beschäftigt sich mit dem Erkennen, Eingrenzen und Beheben von fehlerhaftem Systemverhalten. Dies sind wesentliche Problemkreise beim Management von Netzen. Bei verteilten Systemen wird diese Aufgabe erschwert durch die große Anzahl an Ressourcen, die möglicherweise weite räumliche Verteilung der einzelnen Komponenten, die Heterogenität des Systems sowie die ebenfalls verteilte Verantwortlichkeit einzelner Bereiche durch verschiedenes Personal.

Ein Fehler ist definiert als Abweichung vom spezifizierten Soll-Zustand der Ressourcen. Das Erkennen von Fehlern erfolgt dabei entweder durch die Komponente selbst oder durch die Meldung unüblichen Verhaltens durch Benutzer. Fehlerquellen sind üblicherweise Datenübertragungssysteme (zum Beispiel gebrochene Kabel), Netzkomponenten (Ausfall eines Switches), Endsysteme (Defekt einer Festplatte im Rechner des Benutzers), Software oder Fehlbedienung.

Ziel des Fehlermanagements ist es, Fehler schnell zu entdecken und durch schnelle Beseitigung die Verfügbarkeit möglichst hoch zu halten. Daraus ergeben sich u. a. folgende Aufgaben:

- Überwachen des Systemzustands
- Entgegennehmen und Verarbeiten von Fehlermitteilungen
- Diagnose der Fehlerursache
- Erkennen von Fehlerauswirkungen auf andere Systeme
- Einleiten der Fehlerbehebung
- Führen eines Trouble-Ticket-Systems
- Benutzersupport

Nützliche Voraussetzungen zur Unterstützung der Diagnose im Rahmen des Fehlermanagements sind dabei:

- ◇ Selbstidentifikation von Systemkomponenten
- ◇ Protokoll der Fehlermeldungen (Error Log)
- ◇ Fähigkeit zum Test isolierter Komponenten
- ◇ Vorkehrungen zum Tracen der Komponenten, das heißt, Protokoll des Datenverkehrs zum späteren Nachvollziehen der Fehlerursache
- ◇ Abrufmöglichkeiten von Speicherabzügen (Memory-Dumps)
- ◇ Möglichkeiten zur Fernwartung und -diagnose
- ◇ Vorhalten definierter Testsysteme zur gezielten Fehlererzeugung
- ◇ Vorhalten dedizierter Hard und -software zur Diagnose

Konfigurationsmanagement

Ein weiterer Aspekt des Managements ist die Behandlung der Konfiguration, worunter man das Setzen und Ändern von Einstellungen, die den normalen Betrieb regeln, versteht. Zu diesen Einstellungen gehören u. a. folgende Punkte:

- ◇ Installation der nötigen Software
- ◇ Netzwerkeinstellungen, wie IP-Adresse und Routen
- ◇ Konfiguration des Nameservers

Zu klären ist dabei, wie die relevanten Einstellungen erfolgen müssen. Zum einen können sie direkt auf den gemanagten Endgeräten erfolgen, zum anderen können Einstellungen aber auch von einem zentralen Management-Dienst an die Endgeräte propagiert werden. Weiterhin muss beachtet werden, ob die Engeräte das dynamische Konfigurieren erlauben, d. h. das Ändern von Einstellungen ohne Unterbrechung des Betriebs, oder ob bei Änderungen ein Neustart nötig ist (statische Konfiguration).

Desweiteren ist zu klären, welche Benutzerschnittstellen und Werkzeuge existieren, mit deren Hilfe der Administrator Änderungen vornehmen kann. Die Einstellungen selbst sowie der Zugang zu den Konfigurationswerkzeugen sind gegen unberechtigten Zugriff zu schützen. Von der Gestaltung der Konfigurationswerkzeuge hängt auch die Akzeptanz ab, die eine neue Technologie bei den Verwaltern eines Netzes erfährt.

Abrechnungs- und Benutzermanagement

Zu diesem Bereich zählt die Verwaltung der Betriebsmittelberechtigungen sowie die Abrechnung der tatsächlichen Nutzung. Um Inkonsistenzen zwischen verschiedenen Benutzerverzeichnissen zu vermeiden, sollten die Berechtigungen für den Remote-Zugang in der zentralen Benutzerdatenbank des Instituts bzw. CIP-Pools integriert werden. Bereits vorhandene Informationen über Benutzer können somit soweit möglich wiederverwendet werden. Zur Einrichtungen zusätzlicher Nutzerparameter sollten weitgehend automatisierte Werkzeuge bereitgestellt werden, um die Administration der neuen Dienste zu vereinfachen.

Da den Nutzern des Instituts alle zur wissenschaftlichen Arbeit notwendigen Netzwerkdienste kostenlos zur Verfügung gestellt werden, ist ein Abrechnungsmanagement nur sehr eingeschränkt nötig. Allerdings sollten Mechanismen vorgesehen werden, die es erlauben, Missbrauch der Ressourcen des Instituts zu erkennen und einem konkreten Nutzer zuzuordnen.

Leistungsmanagement

Als Fortsetzung des Fehlermanagements kann das Leistungsmanagement betrachtet werden. Während sich das Fehlermanagement damit beschäftigt, dass die betrachteten Dienste überhaupt funktionieren, beschäftigt sich das Leistungsmanagement mit der Qualität der Dienste. Die gewünschte Qualität wird mit Hilfe einer definierten Dienstgüte festgelegt. Dazu gehört bei Netzwerkdiensten die Spezifikation eines gewünschten Datendurchsatzes mit Mittel- und Grenzwerten sowie das Festlegen von Überwachungsaktivitäten, um Abweichungen davon sicher feststellen zu können. Neben dieser statischen Vorgehensweise muss während des Produktionsbetriebs regelmäßig kontrolliert werden, ob die ursprünglich vorgesehene Dienstgüte zur Aufrechterhaltung eines qualitativ akzeptablen Betriebs weiterhin optimal ist oder ob durch eine Änderung des Nutzerverhalten eine Anpassung der gewünschten Parameter notwendig geworden ist.

Sicherheitsmanagement

Die Anforderungen, die sich aus dem Management der Sicherheit ergeben, sind in Abschnitt 2.5 zusammengefasst.

2.4 Organisatorische Anforderungen

Da das Netzwerk des Instituts keine isolierte Einrichtung, sondern im Münchner Hochschulnetz integriert ist, fallen in diesen Bereich alle Anforderungen des Betreibers, des Leibniz Rechenzentrums (LRZ). Es ist zu klären, welche IPv6-Infrastruktur bereits im LRZ zur Verfügung steht und inwieweit sie für diese Arbeit genutzt werden kann. Obwohl es sinnvoll wäre, bereits für den Produktionsbetrieb vergebene IPv6-Adressen zu verwenden (statt 6bone-Testadressen), muss diese Entscheidung davon abhängen, ob solche Adressen im LRZ zur Verfügung stehen und ins Institut geroutet werden können.

Weiterhin sind bestehende Sicherheitspolicies zu bei dieser Arbeit zu berücksichtigen. So darf unter gar keinen Umständen der Zugang mit IPv6 die Sicherheit des Netzwerkes stärker gefährden oder mehr bzw. andere Dienste exponieren als dies bei Verwendung von IPv4 geschieht.

2.5 Sicherheitsanforderungen

Spätestens seit den spektakulären Einbrüchen in fremde Netzwerke in den letzten Jahren ist Sicherheit ein vorrangiges Ziel bei der Planung von Netzwerken. Als Beispiel sei hier der Einbruch in die Kreditkartendatenbank von CD Universe im Januar 2000 ([Nash 02]) bei der angeblich 300.000 Kreditkartennummern gestohlen wurden und der Dieb versucht hat, sein Opfer mit der Drohung der Veröffentlichung zu erpressen. In diesem Abschnitt sollen daher die möglichen Angreifer sowie die von ihnen ausgehenden Sicherheitsbedrohungen analysiert werden, denen durch geeignete Maßnahmen im Rahmen dieser Arbeit entgegengewirkt werden soll.

2.5.1 Angreifer

Nach [aCPo 01] lassen sich Angreifer wie folgt in verschieden Gruppen einteilen:

- Hacker: Sie brechen aus Spieltrieb und Ehrgeiz in fremde Rechensysteme ein, um ihren Status innerhalb einer Community zu verbessern. Oft handelt es sich dabei um Jugendliche, sie arbeiten wenig zielgerichtet, sind aber unberechenbar und können großen Schaden anrichten.
- IT-Spione: Bezahlte Angreifer, die versuchen, konkrete Informationen für ihre Auftraggeber auszuspähen. Ihre Motivation ist politisch oder wirtschaftlich.
- IT-Terroristen: Ihr Ziel ist es, Rechnernetze anzugreifen und möglichst hohen Schaden anzurichten. Aus politischen Gründen soll Angst und Chaos erzeugt werden.

- Kriminelle: Ihr Ziel ist die persönliche Bereicherung durch Einbrüche in fremde Netzwerke.
- Vandalen zerstören um der Zerstörung willen. Diese Gruppe überschneidet sich oft mit der der Hacker.

2.5.2 Angriffszenarien

Seit der Verbreitung des Internet zielen die meisten der Angriffe auf Schwachstellen in der Kommunikation. Grundsätzlich werden zwei Arten von Angriffen unterschieden: passive Angriffe und aktive Angriffe.

Passive Angriffe

Passive Angriffe sind dadurch gekennzeichnet, dass der Betrieb des Kommunikationssystem und die übertragenen Daten nicht verändert werden. Angriffe dieser Art sind bspw. die Verwendung von Netzwerksniffern oder das Abhören von Richtfunk- der Satellitenverbindungen. Passive Angriffe lassen sich in drei Arten einteilen:

- Ausspähen von Daten: Der Angreifer gelangt unmittelbar in den Besitz von Daten und kann sie entsprechend seinem Ziel verwenden. Ein Beispiel ist das Abhören einer Verbindung zwischen POP3-Server und -Client, um Login und Passwort von Nutzern herauszufinden. Eine weitere Möglichkeit ist das Herausfiltern von SMTP-Datenströme, um so den gesamten Email-Verkehr auswerten zu können. Solche Angriffe sind problemlos durchführbar, wenn der Angreifer Zugang zum lokalen Netzwerk hat und der Verkehr nicht verschlüsselt ist.
- Ausspähen von Teilnehmeridentitäten auf Anwendungsebene: Der Angreifer möchte dadurch in Erfahrung bringen, welche Kommunikationsteilnehmer untereinander Daten austauschen und kann dadurch Rückschlüsse auf die Art der Kommunikation ziehen. Dazu wird versucht, aus Anwendungsdaten, z. B. dem Email-Verkehr, die Identitäten der Teilnehmer zu ermitteln. Dies ist auch der Wunsch von Strafverfolgungsbehörden, wenn sie die Speicherung aller Verbindungsdaten bei den Diensteanbietern fordern.
- Verkehrsflussanalyse: Selbst wenn die Datenübertragung verschlüsselt ist, kann eine Analyse des Verkehrsflusses auf Netzwerkebene Informationen liefern. Dabei werden Art, Richtung, Zeitpunkt und Größe der Übertragung ausgewertet, um daraus wieder auf die Teilnehmer oder Teilnehmergruppen schließen zu können.

Aktive Angriffe

Im Gegensatz zu passiven Angriffen wird bei aktiven Angriffen die Kommunikationsinfrastruktur gezielt sabotiert. Dies kann bspw. durch das Durchtrennen von Übertragungsleitungen oder die Emulation von Protokollen geschehen. Man unterscheidet folgende Arten:

- Wiederholen oder verzögern von Informationen: Dadurch können die Teilnehmer irritiert oder zu einer falschen Aktion veranlasst werden. Beispiele hierfür sind das Wiederholen einer Übertragung von Login-Daten (Replay-Attacke) oder die mehrfache Ausführung einer Überweisung im Online-Banking.
- Einfügen, Löschen oder Verändern bestimmter Daten: Der Angreifer verändert dabei die übertragenen Daten, indem er Teile ergänzt oder löscht. Dadurch soll der Empfänger manipuliert und zu bestimmten Aktionen verleitet werden. Darunter fallen bspw. Angriffe, die beim Online-Banking einen Überweisungsauftrag abfangen, dem Sender aber die korrekte Übertragung und Ausführung seines Auftrags vorspiegeln. Der Angreifer kann dann in der ursprünglichen Nachricht das Ziel-Konto oder die Summe verändern und diesen veränderten Auftrag an den Rechner der Bank weiterleiten (Man-in-the-Middle-Angriff).

- Dienstverweigerung (Denial-of-Service): Darunter versteht man das systematische Versenden nutzloser Nachrichten, die die Ressourcen des Empfängers so belasten sollen, dass "echte" Anfragen nicht mehr bearbeitet werden können. Neuere Angriffstechniken wurden so weiterentwickelt, dass der Angriff nicht nur von einem Rechner, sondern einer Vielzahl zuvor für diesen Zweck manipulierten Maschinen ausgeht (Distributed Denial-of-Service). Die Abwehr solcher Angriffe ist derzeit noch ungelöst und Gegenstand der Forschung.

- Vortäuschung einer falschen Identität: Gelingt es einem Kommunikationspartner sich für anderen auszugeben, so kann er Zugriff auf Informationen erhalten, die nur für den bestimmt sind, dessen Identität angenommen wurde. Solchen Angriffen geht oft erst ein passives Ausspähen von Authentifizierungsdaten o. ä. voraus. Wird beiden Teilnehmern einer bidirektionalen Kommunikation eine falsche Identität vorgetäuscht, so handelt es sich um einen Man-in-the-Middle-Angriff.

- Leugnen einer Kommunikationsbeziehung: Aufgrund der zunehmenden Verwendung elektronischer Kommunikation zur Abwicklung von Rechtsgeschäften kann ein Angreifer das Senden oder den Empfang von Daten abstreiten und damit wirtschaftlichen Schaden beim Kommunikationspartner anrichten. Als Beispiel sei das Abstreiten von Bestellungen bei einem Internetversandhaus genannt.

2.5.3 Unbeabsichtigte Fehler

Neben dem absichtlichen Verändern von Nachrichten gibt es auch die Möglichkeit von zufälligen Verfälschungen. Dieses Problem kann nach [aCPo 01] aus folgenden Gründen auftreten:

- Fehlrouting von Informationen: In den Routern an den Verbindungsstellen von Netzwerken kann es zu Fehlern kommen, die einzelne Datenpakete an die falsche Adressen weiterleiten. Dieser Fehler kann bereits während der Etablierung einer Verbindung auftreten, so dass der gesamte Datenverkehr dieser Verbindung an das falsche Ziel geleitet wird.

- Übertragungsfehler: Durch Einwirkungen auf das Übertragungsmedium (z. B. Übersprechen von benachbarten Leitungen oder Fremdstörungen) kann es zu Bitfehlern in der Übertragung kommen. Die Wahrscheinlichkeit dafür liegt zwischen 10^{-4} bis 10^{-7}, das heißt bereits bei der Übertragung von 1 Kb kann ein einzelnes Bit umfallen und so möglicherweise sicherheitskritische Fehler verursachen.

- Software-Fehler: Der größte Teil der heute eingesetzten Software hat Fehler, die zu Problemen führen können. So könnte ein Implementierungsfehler im DNS das Versenden von Nachrichten an das falsche Ziel verursachen, indem bspw. bei Anfragen nach einer IP-Adresse nicht die Adresse geliefert wird, die in der DNS-Datenbank hinterlegt wurde.

- Hardwarefehler durch elektromagnetische Einflüsse: Ebenso wie auf dem Übertragungsmedium kann ein Bitfehler auch innerhalb von Rechnersystemen, z. B. im Speicher eines Routers, auftreten und so ein fehlerhaftes Verhalten verursachen.

- Fehlbedienung: Natürlich kann auch ein Benutzer unbeabsichtigt sicherheitskritische Daten bspw. an den falschen Benutzer senden, indem er in seinem Adressbuch den falschen Eintrag auswählt.

Wie man sieht, kann auch durch unbeabsichtigte Fehler ein Sicherheitsproblem ausgelöst werden. Sie lassen sich nur schwer oder gar nicht verhindern, allerdings kann durch eine sorgfältige Planung der Infrastruktur der mögliche Schaden begrenzt werden.

2.5.4 Folgerungen

Die Infrastruktur des Instituts ist vielfältigen Bedrohungen ausgesetzt. Wegen der Offenheit von Rechnerräumen und dem einfachen Zugang zum Netzwerk ist der physische Zugriff auf das Netzwerk einfach zu realisieren. Die passenden Angriffstools sowohl für passive als auch aktive Angriffe sind im Internet frei erhältlich und auch für Laien einfach zu benutzen. Die Zunahme der spektakulären Angriffe in den letzten Jahren deutet auch darauf hin, dass die Anzahl der Attacken in Zukunft eher zu- als abnehmen wird.

Im Rahmen dieser Arbeit ist daher großer Wert auf die Sicherheit der übertragenen Daten zu legen, um diesen Bedrohungen zu begegnen. Insbesondere für die Kommunikation mit Netzen außerhalb des Instituts sind Vorkehrungen zum Schutz der Daten zu treffen. Diese müssen nach [Aust 01] folgende Bedingungen erfüllen:

- Vertraulichkeit: Unter Vertraulichkeit versteht man die Sicherung von Daten gegen den Zugriff nicht dazu berechtigter Dritter. Dies wird durch Verschlüsselung von Daten auf dem Übertragungsweg realisiert.
- Authentifizierung: Darunter versteht man die Verifizierung der Identität einer Person und/oder die Verifizierung des Ursprungs oder Absenders der Daten. Dadurch soll sichergestellt sein, dass in einer Kommunikationsverbindung die beiden Teilnehmer tatsächlich diejenigen sind, für die sie sich ausgeben. Authentifizierung kann über mehrere Verfahren realisiert werden:
    - Etwas, das nur die zu authentifizierende Instanz weiß, also zum Beispiel ein Passwort oder eine PIN.
    - Etwas, das nur die zu authentifizierende Instanz besitzt, zum Beispiel ein Türschlüssel oder eine Chipkarte.
    - Etwas, das nur der zu authentifizierenden Instanz zu eigen ist, das im Gegensatz zum vorherigen Punkt also nicht von der Person getrennt werden kann, wie zum Beispiel ein Fingerabdruck oder ein Netzhautmuster.
- Autorisierung: Darunter versteht man die Gewährung von Zugriffsprivilegien für authentifizierte Benutzer. Nutzer, die sich remote per IPv6 am Netzwerk anmelden, sollen nur die Berechtigungen haben, die sie auch dann bekommen, wenn sie sich lokal oder mit Hilfe anderer Remote-Zugriffsmöglichkeiten anmelden.
- Integrität: Unter Integrität versteht man den Schutz der übertragenen Daten gegen Veränderung durch unberechtigte Dritte.
- Nicht-Abstreitbarkeit: Darunter versteht man die Kombination von Integrität und Authentifizierung, die gegenüber Dritten beweisbar ist. Der Urheber einer Nachricht soll also seine Urheberschaft nicht abstreiten können, um damit Angriffe nach 2.5.2 zu unterbinden.

Kapitel 3

Möglichkeiten zur Realisierung eines Parallelbetriebs

Entsprechend den Anforderungen soll nun ein Parallelbetrieb von IPv4 und IPv6 konzipiert werden. Die Implementierung eines solchen Parallelbetriebs wird am Beispiel Linux vorgestellt.

3.1 Das Internetprotokoll IPv6

Entsprechend [DeHi 98] ist IP Version 6 (IPv6) die neue Version des Internet Protokolls als Nachfolger von IP Version 4 (IPv4) [Post 81] entwickelt worden. Die Änderungen betreffen hauptsächlich folgende Felder:

- Erweiterte Adressierungsmöglichkeiten: IPv6 erweitert den Adressraum von 32 Bit auf 128 Bit, um mehr Ebenen in der Adressierungshierachie, eine weit größere Anzahl von Netzwerkknoten und eine einfache automatische Konfiguration von Adressen zu ermöglichen. Desweiteren wurden neue Möglichkeiten des Multicastings (d. h. Versenden von Paketen an eine Gruppe) erweitert und mit Anycasting eine neue Adressierungsmöglichkeit geschaffen, um Pakete an ein beliebiges einzelnes Mitglied einer Gruppe zu versenden.
- Vereinfachung des Headerformats: Einige der IPv4-Headerfelder wurden entfernt oder als optional spezifiziert, um den Verarbeitungsaufwand in Routern zu senken und den Protokolloverhead zu reduzieren.
- Erweiterung der Möglichkeiten für Optionen: Änderungen in der Codierung des Headers erlauben ein effizienteres Routing und den flexibleren Einsatz von Optionen.
- Möglichkeit des "Flow Labeling": Ein Flow ist ein durch ein spezielles Label gekennzeichneter Datenstrom, für den der Sender besondere Verarbeitungsrichtlinien vorgesehen hat, wie bspw. die Übertragung von Daten in Echtzeit. Diese Möglichkeit wird erstmals in IPv6 eingeführt.
- Möglichkeit zur Authentifizierung und Verschlüsselung: Für IPv6 wurden Erweiterungen spezifiziert, die Authentifizierung, Datenintegrität und (optional) Vertraulichkeit der übertragenen Daten sicherstellen.

Im folgenden werden nun die einzelnen Features näher besprochen.

3.1.1 Adressierung

Adresslänge

Die augenfälligste Änderung, die das neue Protokoll mit sich bringt, ist sicherlich die neue Adresslänge von 128 Bit, die $3,4 * 10^{38}$ verschiedene Adressen ermöglicht.

In der Literatur, zum Beispiel bei Dittler [Ditt 02] finden sich hierzu interessante Vergleiche: Mit dem neuen Adressraum stehen für jeden Menschen $6 * 10^{28}$ Adressen zur Verfügung. Oder anders berechnet: Für jeden Quadratmillimeter der Erdoberfläche stehen 667 Billiarden Adressen bereit.

Selbst bei weiterhin explosionsartigem Wachstum des Internets und bei Fortführung der bisher verschwenderischen Vergabepraxis sollte dieser Vorrat für einige Zeit ausreichen.

Das neue Adressformat

Bisher (das heißt in Zeiten von IPv4) wurden Adressen wie folgt dargestellt: Um dem Anwender, falls er doch mal direkt mit Adressen in Berührung kam, den Umgang zu erleichtern, wurden von den 32-Bit Adressen je 8 Bit (1 Byte) als Dezimalzahl dargestellt und die vier Zahlen durch Punkte getrennt. Das Ergebnis ist die heute übliche Schreibweise, z. B. 217.76.89.156. Dieses Verfahren würde allerdings bei Adressen mit 128 Bit zu äußerst unhandlichen Gebilden führen.

Daher wurde für IPv6 die Darstellung im hexadezimalen Zahlensystem gewählt, um die Darstellung kompakter als bisher zu gestalten. Zur besseren Lesbarkeit werden Gruppen von 2 Bytes bzw. 4 Halbbytes gebildet und diese jeweils durch Doppelpunkt voneinander getrennt:

5712:0:0:0:0:5:EEC1:6008

Innerhalb einer Gruppe wird auf führende Nullen verzichtet. Als weitere Abkürzung ist es gestattet, einmal innerhalb einer Adresse eine Gruppe von beliebig vielen, aufeinander folgenden Nullen durch einen doppelten Doppelpunkt zu ersetzen: 5712::7:EEC1:8 ist das gleiche wie 5712:0000:0000:0000:0000:0007:EEC1:0008. Die beiden Doppelpunkte können auch am Beginn oder am Ende der Adresse stehen, allerdings nur ein einziges Mal innerhalb einer Adresse.

Präfix-Notation

Die Darstellung eines Adresspräfixes ist ähnlich der Darstellung, die für IPv4 in [FLYV 93] eingeführt wurde. Dabei werden, getrennt durch einen Schrägstrich, IPv6-Adresse und Länge des Präfixes nebeneinander geschrieben. So bezeichnet der Ausdruck 5712::/35 ein Präfix der Länge 35 Bit.

3.1.2 Aufteilung der Adressen

Wie bei IPv4 bezeichnet eine IP-Adresse keine Station, sondern ein Interface in einem Subnetz. Ein Multi-homed-Host hat also auch in IPv6 mindestens so viele Adressen wie Interfaces. Tatsächlich ist in IPv6 die Zuweisung mehrerer Adressen an eine Schnittstelle eher die Regel als die Ausnahme, um Routing- und Management-Aufgaben zu erleichtern.

Nach [Huit 00] gehört eine IPv6-Adresse zu einer der folgenden Kategorien:

Unicast Unicast ist die neue Bezeichnung für die bisher bereits üblichen Punkt-zu-Punkt-Adressen. Diese Adressen geben genau eine Schnittstelle innerhalb ihres Gültigkeitsbereiches an. Ein Paket, das an diese Adresse geschickt wird, wird üblicherweise an genau diese Schnittstelle geschickt.

Multicast Eine Multicast-Adresse gibt eine Gruppe von Schnittstellen an. Ein Paket, das an eine Multicast-Adresse geschickt wird, wird an alle Mitglieder dieser Gruppe weitergeleitet. Für Multicast-Anwendungen ist der Adressraum ff00::/8 reserviert. In [HiDe 02] sind verschiedene Adressen für Multicastgruppen vorbelegt, die nicht für eigene Gruppen verwendet werden dürfen. Dazu gehören u. a. Adressen für lokale Router oder alle lokalen Endgeräte.

Anycast Eine Anycast-Adresse gibt ebenfalls eine Gruppe von Schnittstellen an. Im Unterschied zu Multicast-Adressen wird ein Paket, das an eine Anycast- Adresse verschickt wird, normalerweise aber nur an ein einziges, d. h. das nächste erreichbare Gruppenmitglied geschickt.

Die Unterschiede der einzelnen Adresstypen sind in Abb. 3.1 dargestellt.

Abhängig von ihrem Gültigkeitsbereich werden Adressen wie folgt bezeichnet:

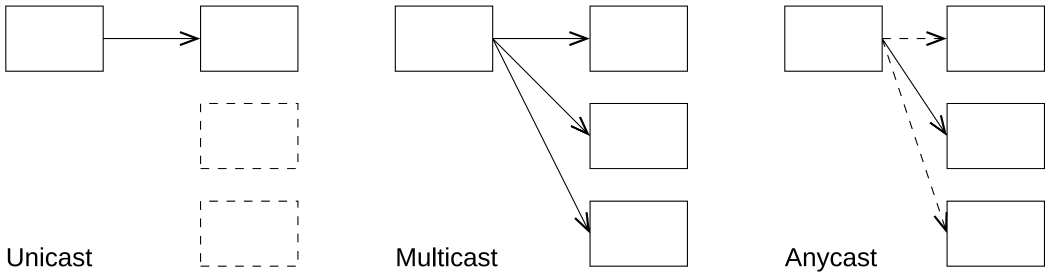

Abbildung 3.1: Adresstypen

Unspezifiziert Die unspezifizierte Adresse ist ::/128 und besteht nur aus Nullen. Sie bezeichnet die Abwesenheit einer Adresse und wird bspw. als Absenderadresse eines Endgeräts verwendet, das noch keine Adresse hat. Sie darf niemals einem Interface zugewiesen oder als Zieladresse genutzt werden.

Loopback Die Adresse ::1/128 wird als Loopback-Adresse bezeichnet und entspricht der IPv4-Adresse 127.0.0.1. Sie wird von Endgeräten verwendet, um Pakete an sich selbst zu senden. Sie darf niemals einem physischen Interface zugewiesen oder als Absender- oder Empfängeradresse eines Pakets eingetragen werden, das ein Endgerät verlässt.

Link local Adressen dieses Typs sind auf eine physikalische Verbindung beschränkt und nur für die lokale Nutzung vorgesehen. Sie können selbständig von den Endgeräten generiert werden und werden genutzt, um beim Bootvorgang Kontakt mit Routern oder Servern herzustellen. Sie werden niemals von Routern nach außen weitergeleitet. Das verwendete Präfix ist fe80::/10.

Site local Mit diesen Adressen kann IPv6 in einem Netz eingeführt werden, das noch keine Verbindung zum Internet mit IPv6 hat. Site local-Adressen unterscheiden sich von Link Local-Adressen durch eine Subnet-ID, so dass diese Adresse innerhalb einer Organisation routbar sind. Wird das Netz später mit dem Internet verbunden, so muss nur das Site local-Präfix ausgetauscht werden. Diese Art von Adressen entsprechen etwa den zum gleichen Zweck benutzen privaten Adressen von IPv4 nach [RMKdG 94]. Zugleich kann damit auch eine Adressierung eingeführt werden, die nur Endgeräten im gleichen Netzwerk zugänglich ist. Das verwendete Präfix ist fec0::/10.

Global Darunter fallen alle Adressen, die weltweit geroutet werden.

Der konkrete Typ einer Adresse wird durch ihre führenden Bits angegeben. Diese definierten führenden Bits haben unterschiedliche Längen, deren derzeitige Bedeutung in Tabelle 3.1 angegeben ist.

Alle Unicast-Adressen, außer denen, die mit dem Präfix (binär) von 000 beginnen, müssen nach [HiDe 02] mit einem 64 Bit langen Interface Identifier gebildet werden. Interface Identifier werden in IPv6 Unicast-Adressen benutzt, um ein Interface an einem Link zu bezeichnen. An diesem Link müssen sie eindeutig sein.

6bone-Adressen

Mit [HFd 98] wurde ein Adressbereich zu Testzwecken reserviert. Dieser Bereich hat das Präfix 3ffe::/16. Das auf diesem Präfix aufgebaute experimentelle Netzwerk trägt den Namen 6bone.

Nutzung	Präfix (binär)	Anteil am Adressraum
Reserviert	0000 0000	1/256
Nicht zugewiesen	0000 0001	1/256
Reserviert für NSAP	0000 001	1/128
Nicht zugewiesen	0000 01	1/64
Nicht zugewiesen	0000 1	1/32
Nicht zugewiesen	0001	1/16
Global Unicast	001	1/8
Nicht zugewiesen	010	1/8
Nicht zugewiesen	011	1/8
Nicht zugewiesen	100	1/8
Nicht zugewiesen	101	1/8
Nicht zugewiesen	110	1/8
Nicht zugewiesen	1110	1/16
Nicht zugewiesen	1111 0	1/32
Nicht zugewiesen	1111 10	1/64
Nicht zugewiesen	1111 110	1/128
Nicht zugewiesen	1111 1110 0	1/512
Link-Local Unicast	1111 1110 10	1/1024
Site-Local Unicast	1111 1110 11	1/1024
Multicast	1111 1111	1/256

Tabelle 3.1: IPv6-Adressraum

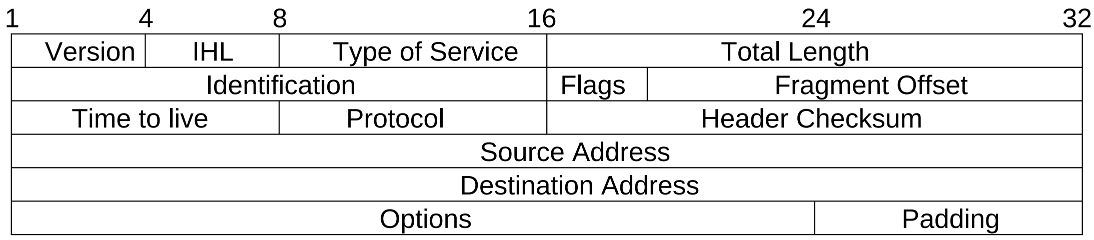

Abbildung 3.2: Format eines IPv4-Headers

3.1.3 Vergleich der Headerformate

Im Vergleich zu IPv4 ist der IPv6-Header auf das unbedingt notwendige Minimum gekürzt worden. Dadurch wird eine effizientere Bearbeitung in Routern ermöglicht sowie der Protokolloverhead reduziert. Um die im Basisheader entfallenen Funktionen und alle neuen Anforderungen im Header abzubilden, ist ein Erweiterungsmechanismus spezifiziert worden. Die geschieht im Vergleich zu IPv4 nicht mit einem Optionsheader variabler Länge, sondern durch die Verkettung beliebig vieler zusätzlicher Header. Jeder dieser Header hat nur eine bestimmte Funktion und wird nur dann verwendet, wenn die Funktion auch tatsächlich benötigt wird. Ein weiterer Vorteil ist, dass Erweiterungsheader in Routern natürlich auch nur dann ausgewertet werden müssen, wenn sie an der betreffenden Station von Bedeutung sind.

Der Aufbau von IPv4-Datagrammen

Abbildung 3.2 zeigt den Aufbau eines IPv4-Headers nach [Post 81].

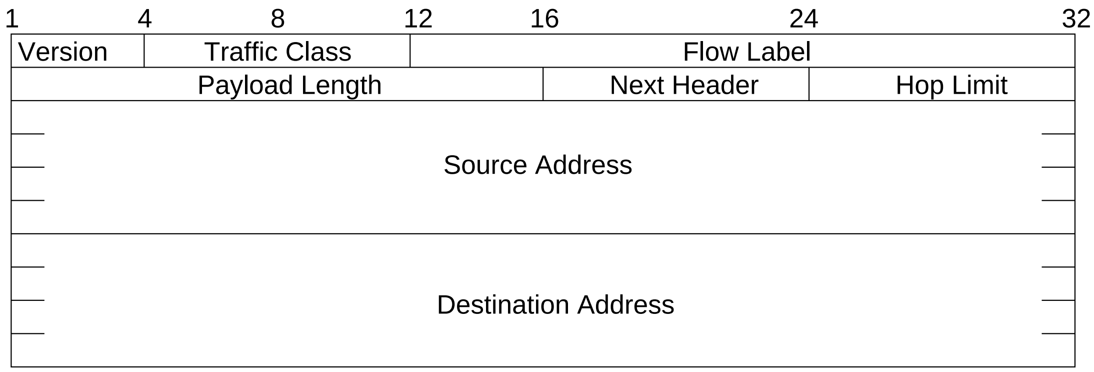

Abbildung 3.3: Format eines IPv6-Headers

Der Aufbau von IPv6-Datagrammen

Abbildung 3.3 zeigt den Aufbau des IPv6-Basis-Headers.

Vergleich

Die auffälligsten Änderungen des neuen Headers sind einerseits die langen Adressen und andererseits die Erweiterungsheader als Ersatz für die Optionsfelder vom IPv4. Bei der alten Version von IP wurden laut [Ditt 02] einige Felder im Header sowie einige Optionen immer seltener benutzt oder erhielten eine andere Bedeutung. Diese Felder sind im neuen Header nicht mehr enthalten. Andere der nur selten genutzten Optionen wurden in Erweiterungsheader ausgelagert.

Die Ausrichtung und die Längen der Headerfelder wurden auf eine Speicher- und Prozessor-Architektur mit einer Zugriffsbreite von 64 Bit ausgerichtet, während IPv4 vor zwanzig Jahren noch eine Ausrichtung auf die damals fortschrittliche 32-Bit-Architektur hatte.

Das Feld zur Angabe der Header-Länge konnte entfallen, da bei IPv6 der Header immer eine feste Länge hat. Zwar ist der Header von IPv6-Paketen größer als der von IPv4-Datagrammen, aber trotz der viermal längeren Adressen nur doppelt so lang. Gleichwohl ist nach ersten Tests davon auszugehen, dass die Gesamtperformance etwa der vom IPv4 entspricht, da die Vereinfachung des Headers bei der Bearbeitungszeit die größere Länge aufwiegt.

Die als Ersatz für das bei IPv4 kaum genutzte Precedence-Feld eingeführten Felder Priorität und Flow-Label ergeben erhebliche Vorteile beim Einsatz im Bereich Echtzeitanwendungen. Durch diese Parameter können einzelnen Datenströmen beim Transport Informationen zur gewünschten Übertragungsqualität zugeordnet werden, so dass Router die Möglichkeit haben, die verfügbare Bandbreite optimal einzusetzen.

Sicherheitsfeatures sind in IPv6 von Anfang an integraler Bestandteil des Protokolls, während sie in IPv4 erst sehr spät als Add-On eingeführt wurden.

Da von IPv6 eine Fragmentierung von Paketen auf der Übertragungstrecken von vorherein ausgeschlossen wurde, konnten die entsprechenden Felder im Header entfallen. Ein optionaler Fragmentierungsheader kann aber weiterhin durch den Sender einer Nachricht eingesetzt werden.

Da das Feld “Time-To-Live” von IPv4 schon lange nicht mehr als Zeitangabe interpretiert wird, wurde es durch das Feld “Hop-Limit” ersetzt, dessen Name jetzt auch wieder zum Inhalt passt.

Eine besondere Headerprüfsumme ist in IPv6 verzichtbar, da eine Prüfsumme über das gesamte Paket als ausreichend angesehen wird.

Das einzige neue Feld im Header ist der Zeiger auf den nächsten Erweiterungsheader, die im nächsten Abschnitt näher betrachtet werden sollen.

Nummer	Typ des Headers bei IPv6
0	Hop-By-Hop-Optionen
60	Ziel-Optionen
43	Routingheader
44	Header für Fragmentierung
51	Header für Authentifizierung
50	Header für Verschlüsselung
59	kein weiterer Header

Tabelle 3.2: Protokoll-Identifier für Erweiterungsheader

Nummer	Typ des Headers bei IPv6
1	ICMP (Internet Control Message Protocol)
2	IGMP (Internet Group Management Protocol)
4	IPv4-Daten im IPv6-Tunnel
6	TCP (Transmission Control Protocol)
8	EGP (Exterior Gateway Protocol)
9	IGP (Interior Gateway Protocol)
17	UDP (User Datagram Protocol)
41	IPv6-Daten in einem Tunnel)
58	ICMP für IPv6
59	No next Header

Tabelle 3.3: Protokoll-Identifier für Payload

3.1.4 Erweiterungsheader

Eine wichtige Neuerung in der Spezifikation von IPv6 ist die Einführung von Erweiterungsheadern zur Darstellung vom Optionen. Dadurch wird es möglich, das Protokoll zu einem späteren Zeitpunkt jederzeit um neue Features zu erweitern. Gleichzeitig wird auch dem Wunsch nach hoher Performance Rechnung getragen, da nicht jede Option bei jedem Schritt der Übertragung ausgewertet werden muss. Oft genügt es, wenn ein Router den ersten Erweiterungsheader der Liste betrachtet, um danach eine korrekte Routingentscheidung zu treffen.

Durch die Verwendung von Zeigern wird es möglich, beliebig viele Erweiterungsheader hintereinander zu verketten. Dazu wurde das Protokollfeld des IPv4-Headers so zum Feld Next erweitert, dass es den Typ des nächsten Erweiterungsheaders angibt. Folgt kein weiterer Header, so enthält das Feld den Typ der Payload (z. B. TCP). Einige mögliche Belegungen des Feldes Next sind in 3.2 und 3.3 aufgeführt.

Um eine effiziente Behandlung von Paketen durch Router zu gewährleisten, legt [DeHi 98] auch eine sinnvolle Reihenfolge der Erweiterungsheader fest (3.4).

Jeder Header, außer den Ziel-Optionen, darf nur ein einziges Mal vorkommen. Eine Ausnahme ergibt sich natürlich bei der Verschachtelung von Protokollen in Tunneln. Der Hop-By-Hop-Options-Header ist der einzige Header, der bei jeder Routingentscheidung berücksichtigt werden muss, er steht daher gleich nach dem Basis-Header.

Der Header Ziel-Option wird abhängig von seiner Position in der Headerkette unterschiedlich interpretiert, wenn er mehr als einmal angegeben wird. Steht er vor dem Routing-Header, so muss sein Inhalt von jedem Router beachtet werden. Steht er dagegen am Ende der Kette direkt vor der Payload, so muss er ausschließlich vom Zielgerät ausgewertet werden.

Nummer	Typ
	Basis-Header
0	Hop-By-Hop-Optionen
60	Ziel-Optionen für jeden Router
43	Routingheader
44	Header für Fragmentierung
51	Header für Authentifizierung
50	Header für Verschlüsselung
60	Ziel-Optionen für Endgerät
	Nutzdaten

Tabelle 3.4: Reihenfolge der Header

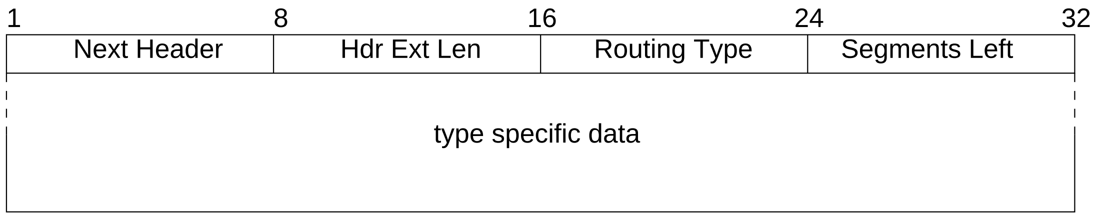

Abbildung 3.4: Routing-Header

Routing-Header

Als Ersatz für die Source-Routing-Option von IPv4 wurde im neuen Protokoll der Routing-Header eingeführt. Mit diesem Header kann der Absender festlegen, welchen Weg sein Datenpaket durch das Internet nehmen soll. Sein Aufbau ist in Abb. 3.4 dargestellt.

Verpflichtend für diesen Erweiterungsheader sind nur die ersten vier Byte, alle weiteren Felder hängen vom Typ des Routing-Headers ab. Das Feld “Next Header” enthält den Typ des nächsten IPv6-Erweiterungsheaders. das Feld “Hdr Ext Len” enthält die Länge des Routingheaders in Einheiten zu 8 Byte, wobei die ersten 8 Byte nicht gezählt werden. Das Feld “Routing Type” bezeichnet die Variante des des Routing-Headers. “Segments Left” bezeichnet die Anzahl der verbleibenden Routing-Segmente, also die Anzahl der explizit aufgeführten Zwischenstationen, die noch auf dem Weg zum Ziel besucht werden müssen. Da die Liste der Adressen von hinten nach vorne abgearbeitet wird, bezeichnet dieser Wert die Anzahl der noch gültigen Adressfelder in diesem Header. Das Feld “type-specific data” ist von variabler Länge, sein Format wird durch den Routing-Type bestimmt. Die Länge muss so gewählt sein, dass die Länge des gesamten Routing-Headers ein Vielfaches von acht Bytes ist. Der Typ-0 Routing-Headers hat das in Abb. 3.5 dargestellte Format.

Das Feld “Reserved” wird mit Nullen belegt. Das Feld “Address[1..n]” ist eine Liste von IPv6-Adressen, die die Stationen bezeichnet, die das Paket auf dem Weg zum Ziel zu besuchen hat.

Fragmentierungsheader

Der Fragmentierungsheader (s. Abb 3.6) wird von einer IPv6-Quelle verwendet, um Pakete zu versenden, die größer sind als die MTU zu seinem Ziel. Im Gegensatz zu IPv4 kann im neuen Protokoll also nur der Sender Pakete in mehrere Fragmente unterteilen. Dadurch soll eine Verschlechterung der Performance durch Fragmentierung auf dem Weg zum Ziel vermieden werden. Falls ein Router feststellt, dass ein Paket zu groß für die Übertragung auf der nächsten Teilstrecke ist, so wird das Paket verworfen und der Sender mit einer ICMP-Nachricht darüber informiert. Dies wird solange wiederholt, bis die größte für eine Strecke anwendbare MTU gefunden wurde. Wie in IPv4 können einzelne Fragmente unabhängig voneinander transportiert werden und müssen vom Empfänger der

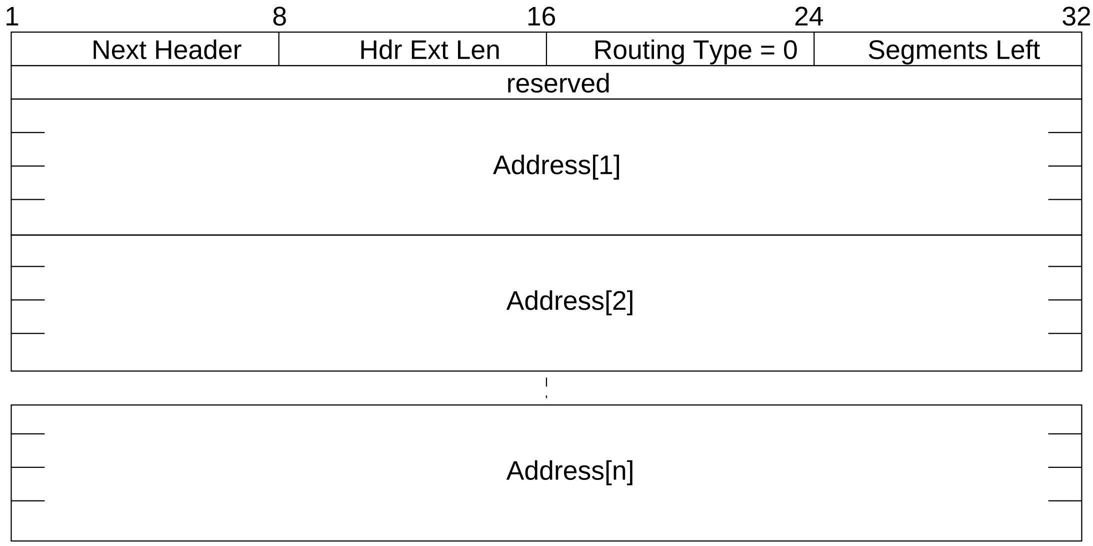

Abbildung 3.5: Typ0-Header

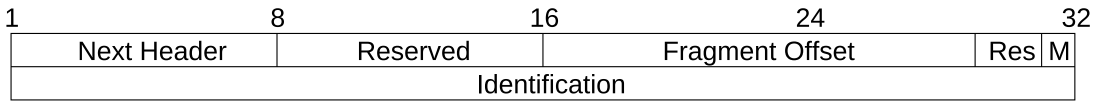

Abbildung 3.6: Fragmentierungsheader

Nachricht zusammengesetzt werden.

Durch das Feld “Next” wird auf den nächsten Erweiterungsheader verwiesen. Das Feld “Offset” enthält die Angabe darüber, an welcher Stelle dieses Fragment in die Gesamtnachricht einzubauen ist. Der Offset wird dabei in 8-Byte-Einheiten angegeben. Das Flag “M” sagt aus, ob weitere Fragmente folgen. Dabei bedeutet das gesetzte Flag, dass weitere Teilblöcke vorhanden sind, bei nicht gesetztem Flag ist dies nicht der Fall.

Anders als bei IPv4 wird im Feld “Identifikation” ein 32-Bit Identifier eingetragen. Jedes Fragment einer Nachricht enthält hier den gleichen Wert. Zur Unterscheidung müssen aber verschiedene Nachrichten, die sich gleichzeitig im Transportkanal befinden können, auch unterschiedliche Werte in diesem Feld haben. [DeHi 98] empfiehlt daher, das Feld als Zähler zu verwenden, der bei jeder Nachricht, die fragmentiert werden soll, um eins erhöht werden soll. Bei Überlauf soll der Zähler einfach zurückgesetzt werden. Es ist darauf zu achten, dass der Header mit seinen Erweiterungsheadern (genauer: den Hop-by-Hop- und Routing-Headern) selbst nicht fragmentiert werden darf.

Wenn nicht alle Fragmente einer Nachricht innerhalb 60 Sekunden beim Empfänger eintreffen, so sind vom Empfänger alle bisher empfangenen Fragmente zu verwerfen. Ist beim Empfänger bereits das erste Fragment angekommen (also das Fragment mit dem Offset 0), so soll eine ICMP-Meldung (Time exceeded) an den Sender verschickt werden.

Hop-by-Hop-Optionsheader

Dieser Header wird verwendet, um Optionen zu spezifizieren, die jeder Router auf dem Transportweg auswerten muss. der Aufbau ist in Abb. 3.7 dargestellt.

Durch das Feld “Next” wird auf den nächsten Erweiterungsheader verwiesen. Das Feld “Hdr Ext

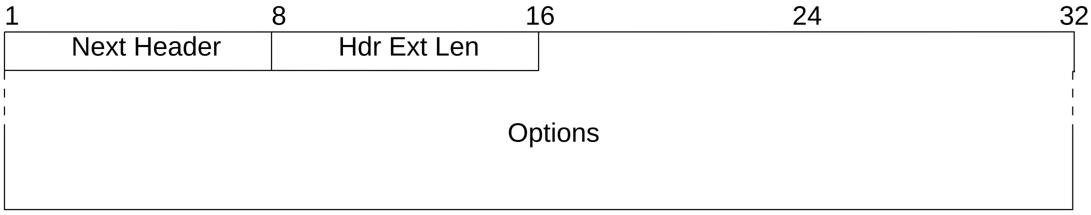

Abbildung 3.7: Hop-by-Hop-Optionsheader

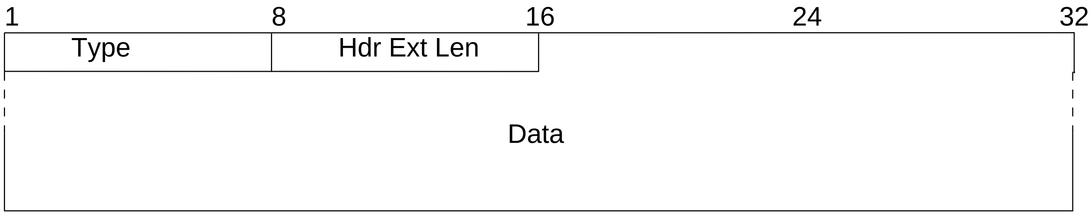

Abbildung 3.8: Aufbau einer Hop-by-Hop-Option

Len” enthält die Länge des Headers in 8-Byte-Einheiten abzüglich der ersten 8 Byte (die immer da sind). Die Minimalgröße des Headers ist damit also auf 8 Byte festgelegt; aus Performancegründen ist der Header immer auf die nächste 8-Byte-Grenze aufzufüllen.

Jede Option, die spezifiziert werden kann, hat den in Abb. 3.8 dargestellten Aufbau.

Der Wert im Feld “Hdr Ext Len” bezeichnet die Länge der Parameter der Option ohne Typ- und Längenfeld; damit wird es möglich, eine parameterlose Option durch den Eintrag von 0 in diesem Feld zu kennzeichnen.

In [DeHi 98] selbst werden nur zwei Optionen definiert: Pad1 und Padn. Diese werden benutzt, um den Hop-by-Hop-Header auf die geforderte 8-Byte-Grenze aufzufüllen. Pad0 ist ist genau ein Byte lang und hat als Typ den Wert 0. Es ist zu beachten, dass in diesem speziellen Fall keine Angaben für Länge und Parameter nötig sind. Padn hat als Typ den Wert 1 und eine variablen Länge. Wird als Länge der Wert 0 angegeben, so ist Padn 2 Bytes lang. Wird ein größerer Wert eingetragen, so ist die Option entsprechend länger, als Parameter werden Nullbytes verwendet.

In [BDH 99] wird als weitere Option die “Jumbo-Payload” definiert. Mit dieser Option werden Pakete ermöglicht, deren Länge die vom Standard zugelassenen 65535 Byte übersteigt. Als Typ wird der Wert 194 eingetragen, die Länge des Headers ist 4 Bytes. Der 4 Byte lange Parameterstring enthält die Länge des Jumbograms in Bytes abzüglich IPv6-Header. Damit ergibt sich eine maximale Länge der Payload von $2^{32} - 1 = 4.294.967.295$ Byte (4 GB).

Die Definition weiterer Parameter ist zu erwarten, so zum Beispiel für das Bandbreitenreservierungsprotokoll RSVP ([Ditt 02]).

Destination-Optionsheader

Dieser Header wird verwendet, um Optionen zu spezifizieren, die nur der Empfänger auswerten soll. Der Aufbau dieses Headers ist in Abb. 3.9 beschrieben.

Durch das Feld “Next” wird auf den nächsten Erweiterungsheader verwiesen. Das Feld “Hdr Ext Len” enthält die Länge des Headers in 8-Byte-Einheiten abzüglich der ersten 8 Byte (die immer da sind). Die Minimalgröße des Headers ist damit also auf 8 Byte festgelegt; aus Performancegründen ist der Header immer auf die nächste 8-Byte-Grenze aufzufüllen.

Die einzigen Optionen, die derzeit definiert sind, sind Pad1 und Padn, wie sie im vorhergehenden Abschnitt beschrieben sind.

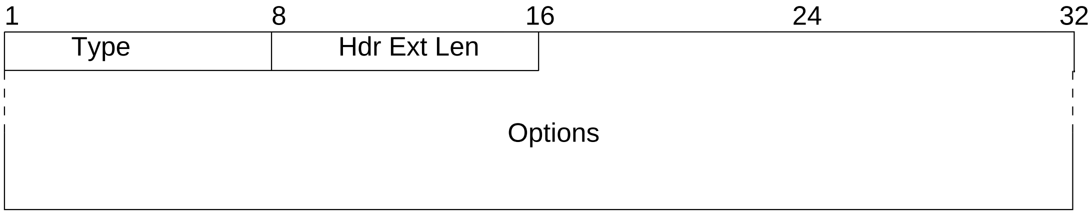

Abbildung 3.9: Destination-Optionsheader

No next Header-Header

Steht im Next-Header-Feld der IPv6-Headers oder eines Erweiterungsheaders der Wert 59, so wird damit gekennzeichnet, dass kein weiterer Header folgt.

3.1.5 Autokonfigurationsmechanismen in IPv6

Um Problemen wie bei IPv4 aus dem Weg zu gehen, wo die Host-Konfiguration in den meisten Fällen manuell erfolgen musste, wurde bei der Spezifikation von IPv6 von Anfang an Wert gelegt auf möglichst automatisierte Selbstkonfiguration von Endgeräten. Neben der weiterhin möglichen manuellen Konfiguration bietet IPv6 zwei verschiedene Verfahren zur automatischen der Netzwerkkonfiguration eines Knotens an. Mit DHCPv6 ([RD 02]) wird das bereits von IPv4 bekannte Verfahren auf die Anforderungen von IPv6 erweitert. Da dieser Server Adressen aus einem vorgegeben Pool vergibt, benötigt er Zustandsinformationen über die bereits vergebenen Adressen. Man spricht daher von einem statusgesteuerten Verfahren.

Beim alternativen Verfahren werden Adressen von den Knoten lokal erzeugt. Mit geeigneten ICMP-Anfragen überprüft dieser Knoten dann die Eindeutigkeit der generierten Adresse. Weitere allgemeine Netzwerkparameter werden dann von Nachbarn oder einem Router bezogen. Da dieses Verfahren ohne zentrale Instanz auskommt, spricht man hier von einem statusfreien Verfahren ([ThNa 98]).

Ebenso möglich sind gemischte Verfahren, bei denen die lokalen Adressen zwar vom Endgerät selbst erzeugt wird, weitere Parameter aber von einem DHCP-Server bezogen werden.

Statusfreie Adressgenerierung

Grundsätzlich ist jedes Endgerät in der Lage, Link-Local-Adressen selbständig und automatisch zu generieren. Da diese Adressen nur in dem Teilnetz eindeutig sein müssen, in dem sich das Endgerät befindet, wird diese Adresse mit Hilfe eines nach EUI-64 gebildeten Interface Identifier generiert.

Für Endgeräte, die mit einer 48 Bit langen MAC-Adresse ausgestattet sind, erzeugt EUI-64 eine 64 Bit lange Hostadresse, indem die MAC-Adresse des Netzwerkadapters auf 64 Bit erweitert wird. Dazu nimmt man die ersten drei Byte der MAC-Adresse, gefolgt von den zwei festen Byte FF FE (hexadezimal), gefolgt von den letzten drei Byte der MAC-Adresse. Zusätzlich muss das zweitniederwertigste Bit des ersten Bytes (universal/local Bit) der so gewonnenen Adresse auf 1 gesetzt werden, wenn diese Adresse global eindeutig ist, wie es bei MAC-Adressen der Fall ist. Ist die Adresse nicht global eindeutig, weil sie bspw. aus einer Telefonnummer gewonnen wurde, so muss dieses Bit auf 0 gesetzt werden. Weitere Regeln zur Generierung von Interface Identifiern nach EUI-64 für unterschiedliche Typen von Endgeräten sind in [HiDe 02] zusammengefasst. Die Bildung einer Hostadresse nach EUI-64 aus einer MAC-Adresse wird in Abb. 3.10 veranschaulicht.

Aus dieser Adresse nach EUI-64 entsteht eine Link-Local-Adresse, indem sie mit dem Präfix FE80::0 zu einer vollständigen IPv6-Adresse erweitert wird.

Immer, wenn eine Adresse lokal erzeugt wurde, muss das Endgerät noch prüfen, ob diese Adresse im Netzwerk bereits verwendet wird. Da die Möglichkeit besteht, Adressen auch manuell zu vergeben, ist nicht ausgeschlossen, dass eine erzeugte Adresse bereits existiert. Diese Überprüfung

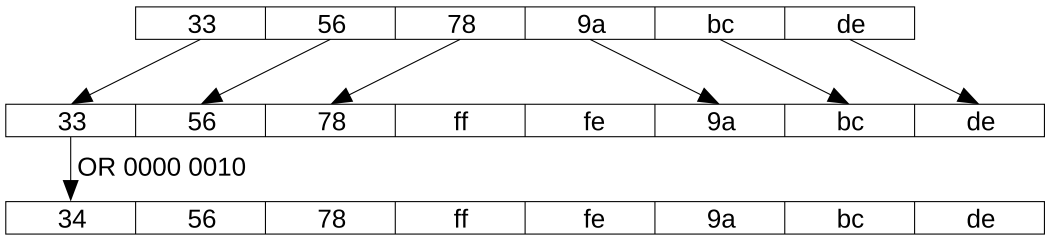

Abbildung 3.10: Bildung einer Hostadresse aus einer MAC-Adresse

auf Dubletten erfolgt mit einer "Nachbar-Anfrage" (Neighbor-Discovery). Wird die Eindeutigkeit der generierten Adresse bestätigt, so kann sie verwendet werden. Zu diesem Zeitpunkt kann das Endgerät mit anderen Geräten am gleichen Link kommunizieren. Die weitere Vorgehensweise der Autokonfiguration findet nur bei Endgeräten Anwendung, für die Autokonfiguration von Routern existieren andere Mechanismen, die im Rahmen dieser Arbeit nicht behandelt werden.

Im nächsten Schritt wird versucht, ein Router Advertisement zu empfangen oder festzustellen, dass kein Router an den Link angeschlossen ist. Wenn ein Router angeschlossen und konfiguriert ist, so senden sie regelmäßig Router Advertisements, die bestimmen, wie das Endgerät weiter verfahren soll. Wenn keine Router gefunden werden können, so soll das Endgerät versuchen, eine statusgebundene Autokonfiguration durchzuführen.

Die Router Advertisements werden periodisch ausgesandt, allerdings sind die Pausen zwischen zwei aufeinander folgenden Nachrichten zu groß, um eine schnelle Autokonfiguration beim Bootvorgang zu gewährleisten. Daher kann ein Endgerät Router Solicitations aussenden, um einen Router zum unverzüglichen Senden eines Router Advertisements zu veranlassen. Bietet der Router statusfreie Autokonfiguration an, so enthält das Router Advertisement Präfix-Informationen, die vom Endgerät zur Bildung von Site-Local- oder Global-Adressen verwendet werden. Gleichzeitig enthält das Router Advertisement Informationen über die zeitliche Gültigkeit sowie die Länge der Präfixe. Zusammen mit dem schon zuvor vom Endgerät gebildeten Interface Identifier können mit den Präfixen gültige IPv6-Adressen gebildet werden. Dabei werden Präfixe ignoriert, wenn bereits eine Adresse mit dem selben Präfix existiert, die nicht autokonfiguriert wurde. Somit wird Autokonfiguration für Endgeräte verhindert, die manuell mit einer Adresse ausgestattet wurden. Durch die periodische Aussendung von Router Advertisements wird sichergestellt, dass die Gültigkeit von Präfixen regelmäßig verlängert wird, außer bei Änderung oder Ungültigwerden eines Präfix.

Typischerweise beträgt die Länge eines Präfix 64 Bit, da ja auch die nach EUI-64 gebildeten Interface Identifier 64 Bit lang sind. Tatsächlich sind die bspw. in Linux implementierten Verfahren zur automatischen Adresskonfiguration nicht in der Lange, Präfixe mit einer Länge ungleich 64 Bit zu verarbeiten. Die Mechanismen in den verschiedenen BSD-Varianten unterstützen nur Präfixlängen, die kleiner oder gleich 64 Bit sind.

Statusgebundene Konfiguration mit DHCPv6

Im Vordergrund bei der Einführung von DHCP für IPv4 stand die möglichst effiziente Verwaltung knapper Ressourcen, besonders von Adressen. Bei Verwendung von IPv6 stehen dagegen allgemeine Konfigurationsprobleme sowie die Durchführung von dynamischen DNS-Updates im Vordergrund der DHCP-Entwicklung.

Die Funktionalitäten von DHCPv4 waren von Anfang an gut skalierbar. Im Laufe der Zeit haben daher über 100 Hersteller eigene Erweiterungen von DHCPv4 eingeführt, die allerdings oft nur von Geräten eines einzigen Herstellers genutzt und verstanden wurden. Anstatt nun DHCPv4 um die IPv6-relevanten Optionen zu erweitern, hat man sich entschlossen, mit DHCPv6 eine eigene

Lösung zu spezifizieren ([RD 02]).

Da DHCP die Möglichkeit bietet, Adressen und andere Information aus einem zentralen Pool zu verwalten und nach Ablauf einer definierten Zeitperiode wiederzuverwenden, spricht man hier von einem statusgebundenen Verfahren.

Bei der Spezifikation von DHCPv6 galten folgende Grundsätze ([Ditt 02]), deren Umsetzung zum Teil auch schon bei DHCPv4 verfügbar war:

- ◇ DHCP ist ein Verfahren und kein Regelwerk, das den Administrator in seiner Freiheit beschneidet.
- ◇ DHCP befähigt den Administrator, alle auf den Endgeräten erforderlichen Einstellungen zentral zu verwalten.
- ◇ Die Verwendung von Relay-Servern zum Ansprechen von DHCP-Servern in anderen Subnetzen ist möglich.
- ◇ DHCP funktioniert auch dann, wenn nur ein Teil der Endgeräte DHCP verwendet.
- ◇ DHCPv6 arbeitet ohne Einschränkung mit statusfreien Mechanismen zusammen.
- ◇ Anders als statusfreie Konfiguration benötigt DHCPv6 keinen Router.
- ◇ DHCP kann vergebene Adressen automatisch an einen DNS-Server weitergeben.

Will ein Endgerät von einem DHCP-Server IPv6-Adressen erhalten, so sendet es eine DHCP-Anfrage (Solicit Message) an eine dafür festegelegte Multicast-Gruppe, der alle DHCP-Server angehören müssen. Jeder Server, der die Anfrage des Endgeräts beantworten kann, sendet daraufhin eine Antwort (Advertise Message) mit seiner Adresse. Das Endgerät wählt einen dieser Server aus und sendet an dessen Unicast-Adresse die Adressanfrage (Request). Wird diese Anfrage vom Server beantwortet (Response), so ist die Transaktion abgeschlossen. Zur Verlängerung der Gültigkeit der angeforderten Adressen werden bei Bedarf Renew Messages zwischen Endgerät und Server ausgetauscht.

Werden von einem Endgerät nur Informationen angefordert, die keine statusgebundene Verwaltung auf dem Server erfordern, also bspw. die Adressen von DNS- oder Zeit-Servern, so wird ein vereinfachtes Verfahren angewandt, bei dem das Endgerät statt der Solicit Message einen Request an die Multicast-Gruppe sendet und diese direkt die angeforderte Information als Response liefert.

Ein weiteres Verfahren, das ebenfalls mit nur zwei Messages funktioniert, allerdings auch für adressbezogene Anfragen verwendet werden kann, setzt voraus, dass das Endgerät bereits Adress- und andere Informationen von einem DHCP-Server erhalten hat. In diesem Fall sendet das Endgerät eine Solicit Message an die Multicast-Gruppe, wobei angegeben wird, dass das Endgerät eine direkte Response von dem bereits vorher verwendeten Server akzeptieren will. Dieser Server sendet die geforderte Information statt der Advertise Message als Response an das Endgerät.

3.1.6 DNS

Um den Benutzern zu ersparen, sich unhandliche IP-Adressen merken zu müssen, wurden bereits bei IPv4 symbolische Namen eingeführt. Diese Namen in Adressen umzusetzen ist Aufgabe des Domain Name Service (DNS). Um solche Namen auch für IPv6-Adressen einsetzen zu können, müssen für DNS-Abfragen daher neue Datentypen spezifiziert werden. Desweiteren muss für die Reverse-Auflösung, die Adressen in Namen umsetzt, eine neue Domain festgelegt sowie die existierenden Anfrage-Typen erweitert werden, um die neuen Datentypen zu transportieren. Diese Erweiterungen mussten so vorgenommen werden, dass sie kompatibel sind mit bisherigen Anwendungen ebenso wie mit vorhandenen DNS-Implementierungen.

Um Namen auf Adressen abzubilden, wurde in [ThHu 95] der Datentyp (Ressource Record, RR) AAAA eingeführt. Als Datum enthält dieser Typ die IPv6-Adresse eines Endgerätes. Besitzt ein Endgerät mehr als eine IPv6-Adresse, so können mehrere AAAA-Records definiert werden.

Während für Reverse-Lookups bei IPv4 die Domain IN-ADDR.ARPA verwendet wurde, ist für IPv6-Adressen die Domain IP6.INT definiert worden. IPv6-Adressen werden dabei als Folge von 4 Bit-Einheiten in hexadezimaler Darstellung (Nibbles), getrennt durch Punkte ausgedrückt. Wie bei IPv4 auch werden diese Nibbles in umgekehrter Reihenfolge aufgeführt. So wird bspw. die Adresse 4321:0:1:2:3:4:567:89ab in einer etwas unübersichtlichen Form dargestellt als b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.INT.

Existierende Anfragen, die zusätzliche A-Anfragen verursachen, so wie Anfragen nach Nameservern (NS) und Mail-Exchangern (MX), müssen so verändert werden, dass sie neben der A-Anfrage auch AAAA-Anfragen vornehmen.

3.1.7 Transitionsmechanismen

Voraussetzung für einen reibungslosen Übergang zwischen IPv4 und IPv6 ist Kompatibilität mit der großen Menge an bereits installierten Routern und Endgeräten, die mit IPv4 arbeiten. In [GiNo 96] wurden daher Mechanismen definiert, die die Interoperabilität zwischen beiden Welten ermöglichen. Es wird erwartet, dass diese Kompatibilität für eine lange Zeit notwendig sein wird.

Grundsätzlich wurden zwei Strategien spezifiziert, die die Interoperabilität ermöglichen sollen:

- Dual IP Layer: Es ist vorgesehen, dass Router und Endgeräte eine vollständige Implementierung beider Protokollstacks besitzen und so problemlos mit IPv4- und IPv6-Geräten kommunizieren können.
- Tunneling: Für die Verbindung von IPv6-Inseln über IPv4-Netzwerke wird die Etablierung von Tunneln vorgeschlagen, so dass bereits IPv6-fähige Netzwerke transparent miteinander kommunizieren können. Dabei werden automatische und konfigurierte Tunnel vorgeschlagen.

Dual IP Layer

Der einfachste Weg, um auf Endgeräten eine Kompatibilität zwischen IPv4 und IPv6 herzustellen, besteht in der Bereitstellung einer kompletten Implementierung beider Protokollstacks auf den einzelnen Knoten. Solche Knoten bezeichnet man als IPv4/IPv6-Knoten. Sie können sowohl IPv4- als auch IPv6-Pakete senden und empfangen und damit direkt sowohl mit IPv4- als auch IPv6-Knoten kommunizieren.

Bei der Konfiguration von IPv4/IPv6-Knoten ist es nicht erforderlich, dass die verwendeten IPv4- und IPv6-Adressen in Beziehung zueinander stehen. Beide Adressen können unabhängig voneinander gewählt werden.

Die DNS-Resolver-Libraries von IPv4/IPv6-Knoten müssen sowohl mit A- wie auch AAAA-Records umgehen können, um eine Namensauflösung für beide Protokolle anbieten zu können.

Tunnelingmechanismen

Bei den meisten Implementierungsszenarien wird eine IPv6-Routing-Infrastruktur langsam aufgebaut. Während die existierende IPv4-Infrastruktur weiterhin bestehen bleibt, werden einzelne Subnetze mit Hilfe von Dual IP Layer IPv6-Fähigkeiten erlangen. Um einzelne IPv6-Inseln über eine IPv4-Infrastruktur miteinander zu verbinden, wurden Tunneling-Mechanismen spezifiziert, die IPv6-Pakete über IPv4-Netze transportieren können.

Tunneling-Techniken werden klassifiziert, indem man betrachtet, wie der Router, der den Eingang zum Tunnel darstellt, die Adresse des Tunnelendpunkts bestimmt. Man unterscheidet dabei automatisches und konfiguriertes Tunneling.

Automatisches Tunneling

Beim automatischen Tunneling wird die Adresse des Tunnelendpunkts durch das Paket bestimmt, das getunnelt werden soll. Die IPv6-Zieladresse des Pakets muss eine IPv4-kompatible IPv6-Adresse sein, deren niederwertige 32 Bit die IPv4-Adresse enthalten und deren höherwertigen 96

Bit mit Nullen aufgefüllt werden. Dadurch kann die IPv4-Adresse des Tunnelendpunkts extrahiert und als Tunnelendpunkt im IPv4-Netz verwendet werden.

Konfiguriertes Tunneling

Bei dieser Variante wird der Tunnelendpunkt mit Hilfe von Konfigurationsinformationen, die Tunneleingang besitzt, bestimmt. Mit einer Routingtabelle kann festgelegt werden, welcher Tunnel bei welcher Zieladresse verwendet werden soll.

6to4

Die Konnektierung einzelner isolierter IPv6-Endgeräte untereinander über ein IPv4-Netz erlaubt 6to4 ([CaMo 01]). Dabei wird als IPv6-Adresse des zu konnektierenden Endgeräts eine spezielle 6to4-Adresse aus der IPv4-Adresse berechnet und verwendet.

Weitere Tunnelmechanismen

Für verschiedene Anforderungen wurden weitere Tunnelmechanismen definiert. Um innerhalb einer Site, die aus verschiedene Subnetzen besteht, einzelne IPv6-Endgeräte in unterschiedlichen Subnetzen miteinander zu verbinden, wurde automatisches Intra-Site-Tunneling-Protokoll (ISATAP) definiert ([FT 02]). Außerdem gibt es diverse Mechanismen, um auch umgekehrt isolierte IPV4-Netze und -Endgeräte über eine IPv6-Infrastruktur zu verbinden, deren Anwendung aber wohl erst in ferner Zukunft notwendig werden wird.

Zusammenfassung

Zusammen mit den im nächsten Kapitel vorgestellten Sicherheitslösungen bieten diese Tunnellingverfahren eine Möglichkeit, Remote-Nutzer an das Institutsnetz anzubinden. Der vielversprechendste Ansatz scheint die Nutzung konfigurierter Tunnel zu sein. Zu klären ist jedoch noch die konkrete Ausgestaltung der Implementierung, die im Kapitel 5 dargestellt wird.

3.2 Mobility-Support in IPv6

Mit der Einführung tragbarer Geräte, die Internetfunktionialität unterstützen, entstand auch der Wunsch der Nutzer, immer und überall erreichbar zu sein. Erreichbarkeit bedeutet in diesem Kontext, unabhängig vom Ort der Inbetriebnahme des Geräts die immer gleiche IP-Adresse nutzen zu können, ohne erst Änderungen an der Konfiguration durchführen zu müssen. Denn in klassischen IP-Netzen ist die Adresse eines Endgerätes immer abhängig von dem Netz, an den das Gerät angeschlossen ist. Die Adresse ist also nur am jeweiligen Standort gültig. Um in anderen Netzen IP-Dienste nutzen zu können, ist es daher immer auch nötig, eine in diesen Netzen gültige Adresse zu konfigurieren.

Werden auf einem solchen mobilen Gerät selbst Dienste angeboten, so muss die neue Adresse allen Kommunikationspartnern mitgeteilt werden, bevor sie die Dienste auf dem mobilen Gerät nutzen können. Diese Vorgehensweise mag bei gelegentlichen Standortwechseln nur lästig sein, bei häufiger Änderung der Adresse ist der Aufwand jedoch unpraktikabel.

Ziel muss es also sein, dass mobile Geräte unabhängig von ihrem Standort immer die gleiche Adresse nutzen können, um so Änderungen in der Konfiguration des Geräts selbst sowie bei Kommunikationspartnern zu vermeiden.

3.2.1 Features von Mobile-IPv6

Eine Methode, um diese Anforderung zu realisieren, wurde bereits 1996 in [PeEd 96] als Erweiterung zu IPv4 entwickelt. In [JPA] wurde diese Spezifikation erweitert, um mit IPv6 benutzt werden zu können.

Grundsätzlich sind Endgeräte nur dann im Internet zu erreichen, wenn sie mit einer Adresse aus dem Netz konfiguriert sind, in dem sie sich gerade aufhalten. Routing in IP-Netzen erfolgt mit Hilfe von Netzadressen, so dass Pakete nicht an Geräte ausgeliefert werden können, wenn sie sich außerhalb ihres Heimatnetzes befinden. Diese Regel findet auch bei IPv6 Anwendung. Allerdings wurde bei der Standardisierung von IPv6 auf eine bessere Unterstützung mobiler Endgeräte geachtet, was in der Spezifierung des Protokolls "Mobile-IPv6" ([JPA]) seinen Niederschlag fand.

Mobile-IPv6 ist eine Erweiterung des IPv6-Protokolls, die es ermöglicht, dass sich ein Endgerät in beliebigen Netzen aufhalten kann, ohne seine Adresse ändern zu müssen. Dadurch wird erreicht, dass es unabhängig von seinem tatsächlichen Standort immer über dieselbe erreichbar ist. Dabei wird es sogar ermöglicht, dass bereits existierende Verbindungen weiterhin bestehen, wenn ein Gerät seine Position, sein Heimatnetz oder das Transportmedium wechselt. So kann ein Endgerät von einem Netzwerksegment in ein anderes wechseln, von einem Ethernetstrang an einen anderen angeschlossen werden oder sich von einer drahtgebundenen Ethernetverbindung in ein WLAN-Segment bewegen. Diese Änderungen am Standort sind dabei transparent für höhere Netzwerkschichten.

Mobile-IPv6 benötigt dabei keine Unterstützung von lokalen Routern. In Netzen, die von mobilen Geräten besucht werden, muss also keine spezielle Routersoftware installiert werden. Die Unterstützung von Mechanismen zur Optimierung des Routings ist integraler Bestandteil des Protokoll, so dass keine nichtstandardisierten Erweiterungen des Routings eingesetzt werden müssen. Das Protokoll arbeitet außerdem unabhängig vom Link-Layer, so dass es auf allen Übertragungsmedien eingesetzt werden kann. So ist zum Beispiel keine Unterstützung von ARP notwendig, da Neighbor Discovery Mechanismen von IPv6 zum Einsatz kommen können. Gleichzeitig wurde versucht, den Overhead des Protokolls sowie die Last des besuchten Netzes zu minimieren, indem statt aufwendiger Tunnels und Broadcasts, wie in anderen Ansätzen, einfache Optionsheader sowie Unicasts verwendet wurden.

Folgende Begriffe wurden im Kontext von Mobile-IPv6 eingeführt:

Home Link Das Heimatnetz des Knotens, definiert durch ein Netzwerk-Präfix.

Home Address Eine routbare Unicast-Adresse, die einem Knoten zugeordnet ist und als permanente Adresse dieses Knotens gilt. Diese Adresse entstammt dem Home Link des Knotens. Alle Pakete, die an diese Adresse geschickt werden, werden an den Home Link geliefert.

Mobiler Knoten Ein Knoten, der seinen Standort ändern kann und trotzdem über seine Home Address erreichbar bleibt.

Korrespondierender Knoten Ein Knoten, der mit dem mobilen Knoten kommuniziert. Dieser Knoten kann mobil oder stationär sein.

Movement Wird ein Knoten an ein anderes Netz konnektiert als er vorher war, so spricht man von einem Movement. Wenn der Knoten nicht an seinem Home Link angeschlossen ist, so sagt man, er ist "nicht zu Hause".

Foreign Link Jedes Netz, das nicht der Home Link des Knotens ist.

Care-of Adresse Eine routbare Unicast-Adresse, die einem Knoten zugeordnet ist, während er mit einem Foreign Link verbunden ist. Die primäre Care-of Adresse ist diejenige Adresse, unter der der Knoten zum aktuellen Zeitpunkt erreichbar ist.

Home Agent Ein Router im Home Link des Knotens, bei dem der Knoten seine jeweils aktuelle Care-of Adresse registriert. Während der mobile Knoten nicht zu Hause ist, übernimmt der Home Agent Pakete, die an die Home Address des Knotens adressiert sind und tunnelt sie an die aktuelle Care-of Adresse weiter.

Binding Die Zuordnung der Home Address eines mobile Knotens zu einer Care-of Adresse, zusammen mit der zeitlichen Gültigkeit dieser Zuordnung.

Registrierung Der Vorgang, während dessen das Binding eines mobilen Knotens erneuert oder eingerichtet wird (Binding Update). Diese Registrierung kann beim Home Agent oder dem Korrespondierenden Knoten geschehen.

3.2.2 Funktionsweise

Ein mobiler Knoten soll immer über die gleiche Adresse erreicht werden können, egal ob er an seinen Home Link angeschlossen ist oder nicht. Wenn der Knoten sich an seinem Home Link befindet, werden Pakete, die an diese Adresse gerichtet sind, mit konventionellen Routingmechanismen an ihr Ziel geleitet.

Wenn der Knoten aber an einen Foreign Link angeschlossen ist, dann ist er zusätzlich über eine oder mehrere Care-of Adressen erreichbar. Diese Care-of Adresse kann der Knoten über konventionelle Autokonfigurationsmechanismen erwerben, sobald er an das fremde Netz angeschlossen wird. Über diese Care-of Adresse kann der Knoten dann Pakete empfangen.

Die Zuordnung einer Care-of Adresse und seiner Home Address wird Bindung genannt. Wenn ein mobiler Knoten außerhalb seines Home Links über seine Home Address erreichbar sein will, so registriert er seine primäre Care-of Adresse an einem Router, der sich an seinem Home Link befindet. Dieser Router agiert dann als Home Agent des mobilen Knotens. Diese Registrierung erfolgt durch Senden eines Binding Updates an den Home Agent, der ein erfolgreiches Update dem mobilen Knoten bestätigt.

Jeder Knoten, der mit einem mobilen Knoten Kommunikationsbeziehungen unterhält, wird als korrespondierender Knoten bezeichnet. Dieser korrespondierender Knoten kann selbst entweder mobil oder stationär sein. Mobile Knoten können ihre korrespondierenden Knoten mit einer Correspondent Binding Procedure über ihren aktuellen Standort informieren.

Es sind zwei Möglichkeiten für die Kommunikation zwischen einem mobilen Knoten und einem korrespondierenden Knoten vorgesehen: Die erste Variante, bidirektionales Tunneling, erfordert keine Unterstützung von Mobile IPv6 durch den korrespondierenden Knoten und ist sogar dann anwendbar, wenn der mobile Knoten sein aktuelles Binding (also seine aktuelle Care-of Adresse) nicht dem korrespondierenden Knoten mitgeteilt hat. Bei diesem Verfahren werden Pakete an den korrespondierenden Knoten vom mobilen Knoten an dessen Home Agent getunnelt und dann von dort aus normal an den korrespondierenden Knoten weitergeroutet. Der Home Agent benutzt dabei einen IPv6-Mechanismus (Neighbour Discovery als Proxy), um alle Pakete, die an die Home Address des mobilen Knotens geschickt werden, abzufangen und an die primäre Care-of Adresse des mobilen Knotens weiterzuleiten. Diese Weiterleitung erfolgt mittels IPv6-Tunneling.

Das zweite Verfahren, genannt "Route optimization", setzt voraus, dass der mobile Knoten dem korrespondierenden Knoten seine aktuelle Care-of Adresse mitteilt. Der korrespondierende Knoten kann dann Pakete an den mobilen Knoten direkt an die Care-of Adresse des mobilen Knoten verschicken. Bevor der korrespondierende Knoten ein IPv6-Paket verschickt, prüft er eine Tabelle von Bindings, und wenn dort die Zieladresse des Pakets enthalten ist, so wird ein spezieller IPv6-Routingheader verwendet, um das Paket an die Care-of Adresse des mobilen Knotens zu verschicken.

Diese zweite Variante erlaubt die Verwendung des kürzesten Pfades zwischen mobilem und korrespondierendem Knoten. Sie vermindert außerdem die Gefahr einer Überlastung des Home Agents und des Home Links. Außerdem werden dabei mögliche Fehler durch Ausfall des Home Agents reduziert.

Wenn ein korrespondierender Knoten Pakete direkt an den mobilen Knoten sendet, so setzt er als Zieladresse für dieses Paket die Care-of Adresse ein. Mit einem speziellen Header, dem Type 2 Routing Header, wird die Home Address des mobilen Knotens in das Paket eingefügt. Vor der Weiterverarbeitung in den höheren Netzwerkschichten wird die in diesem Header enthaltenen Home Address als Zieladresse in das Paket kopiert, so dass die Verwendung einer Care-of Adresse für höhere Netzwerkschichten transparent erfolgt.

Analog setzt der mobile Knoten als Absenderadresse eines an den korrespondierenden Knoten gerichteten Pakets seine Care-of Adresse ein. Seine Home Address teil der dem Partner mit

Hilfe einer Home Address Option mit. Dadurch kann der korrespondierende Knoten vor der Weiterverarbeitung des Pakets in höheren Netzwerkschichten die Absenderadresse des Pakets durch die Home Address des mobilen Knotens ersetzen, so dass auch hier die Verwendung der Care-of Adresse transparent bleibt.

3.2.3 Betrachtungen zur Sicherheit

Zur Sicherung der Kommunikation bietet Mobile IPv6 Mechanismen an, die das Protokoll gegen Angreifer schützen sollen. Dabei handelt es sich insbesondere um folgende Features:

Verschlüsseltes Reverse Tunneling Jeder Mobile IPv6-Knoten muss diesen Mechanismus unterstützen. Dadurch können Angriffe auf die Kommunikation zwischen Knoten, die mittels des Home Agent abgewickelt werden, unterbunden werden.

Sicherung der Binding Updates Ein besonders attraktives Ziel für Angreifer sind die Binding Updates, die an den Home Agent sowie die korrespondierenden Knoten gesendet werden.

Sicherung der Home Address Destination Option Dadurch werden Angriffe verhindert, die versuchen, einem korrespondierenden Knoten ein falsche Home Address vorspiegeln.

Die Sicherung dieser Punkte erfolgt mittels der im nächsten Kapitel vorgestellten Standardmechanismen, die IPv6 zur Verfügung stellt, also der für Authentifizierung und Verschlüsselung von Paketen zur Verfügung gestellten Verfahren.

3.3 IPv6-Adressarchitektur

Bevor IPv6-Adressen an einzelne Teilnehmer des IPv6-Betriebs im Institut vergeben werden können, müssen Regelungen getroffen werden, in welcher Weise dies geschehen soll. In einem Top-down-Ansatz sollen die vom IETF u. a. entwickelten Regeln ([IAIE 01], [Hind 98], [HDFH 00], [add 02], [HiDe 02]) für den globalen Adressraum dargestellt werden und konsistent erweitert werden, um innerhalb des Instituts Anwendung zu finden.

Dabei sind die Vergaberichtlinien so zu gestalten, dass einerseits effizient mit den vorhandenen Adressen umgegangen wird, andererseits ist der zur Verfügung stehende Adressraum so umfangreich, dass allen Nutzern einfacher Zugang gewährt werden kann und soll. Dabei sind folgende Kriterien zu erfüllen:

Eindeutigkeit Es muss garantiert sein, dass jede vergebene globale Adresse weltweit eindeutig ist. Das bedeutet, dass einerseits alle im Institut verwendeten Adressen aus dem zugeteilten Adressraum stammen müssen, und andererseits, dass die innerhalb dieses Adressraums gebildeten Adressen innerhalb des Instituts nur ein einziges Mal vergeben werden dürfen.

Aggregation Ein wesentliches Kriterium bei der Vergabe von IPv6-Adressen ist das Prinzip der hierarchischen Vergabe, die der Netzwerktopologie folgen soll. Dadurch soll erreicht werden, dass Routingtabellen so effizient wie möglich aufgebaut werden können. Dies ist insofern wesentlich, als der Umfang des globalen Adressraums bei übermäßiger und unkontrollierter Segmentierung zu riesigen, kaum wartbaren Routingtabellen führen würde. Dieses Prinzip sollte auch im kleinen Rahmen des Institutsnetzes verfolgt werden.

Effizienz Obwohl der Adressraum außerordentlich umfangreich ist, sollte unnötige Verschwendung vermieden werden. Gerade hinsichtlich neuer, noch nicht bekannter Entwicklungen ist der zukünftige Verbrauch von Adressen noch nicht vorhersehbar.

[add 02] zählt weitere Kriterien auf, die allerdings nur für die globale Adressvergabe relevant sind. Darunter zählen Fairness sowie Minimierung des bürokratischen Overheads bei der Beantragung von Adressraum.

Problematisch sind bei diesen Kriterien die teils konkurrierenden Erfordernisse der Aggregation und der Effizienz. Einerseits sollen Adressen großzügig vergeben werden, um hierarchisches Routing sowie für alle Nutzer zusammenhängende Adressräume zu ermöglichen. Andererseits sollen aber keine Adressen verschwendet werden, um weiteres Wachstum des Netzes auf allen Ebenen zu ermöglichen. Hier ist bei der Definition von Vergaberichtlinien innerhalb des Instituts darauf zu achten, einen möglichst sinnvollen Kompromiss zwischen beiden Anforderungen herzustellen.

Zuteilung von Adressen Das Standardisierungsgremium IETF hat festgelegt [Ditt 02], dass IPv6-Adressen zentral vergeben werden sollen. Diese Aufgabe wird von der IANA (Internet Assigned Numbers Authority) unter der Aufsicht von ICANN (Internet Corporation for Assigned Names and Numbers) wahrgenommen. Nach den Vorgaben der IANA wird der Adressraum in Abstimmung mit den regionalen Registraturen verwaltet.

Im Gegensatz zu IPv4 ist die Vergabe von IPv6-Adressen nicht endgültig, vergebene Adressblöcke können wieder zurückgerufen werden, falls dies aus technischen Gründen oder wegen Missbrauchs notwendig sein sollte.

Von IANA werden die Adressen nicht direkt an Endkunden, sondern an regional oder anders aufgeteilte Registraturen vergeben, die Blöcke von Adressen wiederum an Provider oder an Endkunden weitergibt. Als Registraturen arbeiten derzeit ARIN für die USA, RIPE für Europa und AP-NIC für den asiatischen und pazifischen Raum. Weitere Registraturen z. B. AfriNic für Afrika sind geplant.

Grundsätzlich sind die linken 64 Bit der Adressen für die Netzadressierung vorgesehen, die rechten 64 Bit für die Hostadressierung. Die Netzadresse wird entsprechend Policies vergeben, die von der IANA in Zusammenarbeit mit den Registraturen festgelegt werden.

In früheren Entwürfen der Policies wurden zur Vergabe der Netzadressen drei Hierarchieebenen vorgesehen, für deren jede eine feste Anzahl Bits in der Adresse vorgesehen ist. Die TLA-Ebene (Top-Level-Aggregation) stellt dabei die oberste Ebene des Internets dar. Es handelt sich dabei um Superprovider, die zentrale Austauschpunkte zur Verfügung stellen und internationale Backbones betreiben. Für diese Hierarchieebene sind 13 Bit zur Bildung von Präfixen vorgesehen. Die NLA-Ebene (Next-Level-Aggregation) besteht aus den jeweiligen Internetprovidern. Jedem NLA-Provider stehen 24 Bit zur Verfügung, um NLA-Präfixe zu bilden. Davon wird er einen Teil zur Strukturierung seines eigenen Netzes verwenden und einen anderen zur Bildung von SLA-Präfixen (Site-Level-Aggregation), die an seine Endkunden (Firmen oder Organisationen) vergeben werden. Damit stehen jeder Organisation 16 Bit zur Segmentierung zur Verfügung.

Nach weiteren Diskussionen wurden Ende 2001 das Konzept der TLA, NLA und SLA mit ihren fixen Präfixlängen aufgegeben und durch ein weiterhin hierarchisches System ersetzt, dass aus der IANA selbst, regionalen (RIR), nationalen (NIR) und lokalen (LIR) Internetregistraturen besteht. Die nationale Ebene kann entfallen und tritt nur dann auf, wenn sich die LIRs auf nationaler Ebene organisiert haben. Dies ist hauptsächlich in der asiatisch-pazifischen Region der Fall. In anderen Teilen der Welt arbeiten Provider oft supranational, so dass eine Adressvergabe entlang politischer Grenzen wenig Sinn machen würde. LIRs sind in der Regel ISPs, die Endnutzer oder andere ISPs mit Internetkonnektivität versorgen. [add 02] legt nur eine Policy für die Vergabe von Adressraum an LIRs fest, die Vergabe an RIRs und eventuell NIRs wird offenbar informell gehandhabt. Nach dieser Policy werden an LIRs mindestens Adressbereiche der Größe /32 delegiert. Die Zuteilung an Enduser erfolgt dann nach folgenden Regeln:

- Im allgemeinen erhalten Enduser ein Netz der Größe /48[1], es sei denn, es handelt sich um sehr große Organisationen.
- Wenn feststeht, dass der Enduser nur ein einziges Netz betreiben will, so erfolgt die Zuteilung eines Netzes der Größe /64.
- Wenn absolut sicher ist, dass nur ein einziges Gerät konnektiert werden soll, so erfolgt die Zuweisung eines Netzes/einer Adresse mit der Präfixlänge /128.

[1] Die Notation /48 bezeichnet dabei die Länge des Netzwerkpräfixes von 48 Bit. Die rechten 80 Bit können dann vom Nutzer frei verwendet werden.

Die Folgerung aus diesen Regeln ist, dass Endusern entweder eine einzelne Adresse (/128) oder, falls mehrere Endgeräte angeschlossen werden sollen, ein Netz der Mindestgröße /64 zugewiesen werden muss. Eine Segmentierung in Netze kleiner als /64, also die Verwendung von Präfixlängen größer als 64 und kleiner als 128, ist nicht zulässig!

Kapitel 4

Konzeption geeigneter Sicherheitsmechanismen

Ein großer Vorteil von IPv6 gegenüber seinem Vorgängerprotokoll IPv4 sind die eingebauten Sicherheitsmechanismen. Anstatt Sicherheit mit kompliziert zu installierenden Zusätzen zu realisieren, haben die Designer des Protokolls alle notwendigen Vorkehrungen getroffen, um Verschlüsselung und Authentifizierung bereits im Kern von IPv6 zu verankern, so dass diese Mechanismen einfach zu bedienen und zu administrieren sind. Das folgende Kapitel soll daher einen Überblich über die vorhandenen Features geben.

4.1 Grundlagen der Kryptographie

Kryptographische Algorithmen verwenden mathematische Verfahren, um durch deren (wiederholte) Anwendung Daten vor dem unberechtigten Zugriff Dritter zu sichern. Dabei nutzt man die Tatsache, dass bestimmte Berechnungen einfach durchzuführen sind, ihre Umkehrfunktion aber nur unter größten Schwierigkeiten. Das bekannteste Beispiel für eine solche Berechnung ist wohl das Faktorisierungsproblem für Primzahlen. Kryptographische Algorithmen verwenden Schlüssel, um aus einem Klartext ein Chiffrat zu berechnen. Mit dem selben oder einem anderen Schlüssel kann die Operation wieder rückgängig gemacht werden. Ein kryptographischer Schlüssel ist dabei ein Bitmuster beliebiger Länge.

Abhängig von der Art des verwendeten Schlüssels lassen sich kryptographische Verfahren in verschiedene Gruppen einteilen, deren wichtigste folgen:

4.1.1 Symmetrische Algorithmen

Bereits im Altertum (siehe zum Beispiel [Caes 95]) wurden Verfahren aus dieser Gruppe eingesetzt. Das Grundprinzip besagt, dass aus dem Klartext mit einem symmetrischen Schlüssel eine verschlüsselte Version des Klartexts (Chiffrat) erzeugt wird. Mit Hilfe des selben Schlüssels kann aus dem Chiffrat der Klartext berechnet werden. Die Symmetrie liegt nun in der Tatsache begründet, dass sowohl beim Ver- als auch beim Entschlüsseln der selbe Schlüssel verwendet werden muss.

Als Schlüssel werden Zufallszahlen einer vorher festgelegten Länge verwendet. Dabei ist es von besonderer Bedeutung, dass der diese Zufallszahl durch einen Generator hoher Qualität erzeugt wird, um Angriffe durch "Erraten" des Schlüssels zu vermeiden. Die Güte eines Generators hängt dabei wesentlich von der Verteilung der erzeugten Zufallszahlen im Schlüsselraum ab. Ausführliche Darstellungen verschiedener Generatoren für Zufallszahlen finden sich in der Literatur (bspw. in [Gent 98]).

Die Qualität der Verschlüsselung hängt also von folgenden Bedingungen ab:

DES-Schlüssel	ECC-Schlüssel	RSA-Schlüssel	Zeit bis Kompromittierung	Maschinen	Speicher
56 Bit	112 Bit	430 Bit	$<$ 5 Minuten	105	trivial
80 Bit	160 Bit	760 Bit	50 Jahre	4300	4 Gb
96 Bit	192 Bit	1020 Bit	3 Mio. Jahre	114	170 Gb
128 Bit	256 Bit	1620 Bit	10^{16} Jahre	0,16	120 Tb

Tabelle 4.1: Relative Stärke von Algorithmen

- Qualität des Algorithmus
- Qualität des Zufallszahlengenerators
- Länge des Schlüssels

Werden alle Bedingungen erfüllt, so lässt sich das Chiffrat ohne Kenntnis des Schlüssels nur dann entschlüsseln, wenn man alle möglichen Schlüssel des Schlüsselraums ausprobiert (Brute-Force-Methode). Statistisch gesehen führt das Durchprobieren von der Hälfte der möglichen Schlüssel zum Erfolg. Daher wird die Länge des Schlüssels so festgelegt, dass dieses Ausprobieren selbst dann aussichtslos ist, wenn man außerordentlich Ressourcen an Zeit und Rechenleistung dafür einsetzt. Typische heute eingesetzte Schlüssellängen reichen von 40 Bit bis 128 Bit, was einem Schlüsselraum der Größe $1,1 * 10^{12}$ bis $3,4 * 10^{38}$ entspricht.

Man unterscheidet zwei Arten von symmetrischen Algorithmen: Block- und Datenstrom-Algorithmen. Block-Algorithmen unterteilen den Klartext in gleich große Teile, die einzeln verschlüsselt werden. Datenstrom-Algorithmen können auf Datenblöcke beliebiger Länge angewendet werden.

Symmetrische Algorithmen sind im Allgemeinen einfach zu implementieren und einfach anzuwenden. Allerdings haben sie zwei bedeutende Nachteile, die ihre Einsatz bei manchen Problemen ausschließen: Da sowohl Sender als auch Empfänger über den selben Schlüssel verfügen müssen, müssen beide Partner diesen Schlüssel besitzen. Da er aber geheim bleiben muss, kann er nicht über den selben Kommunikationskanal ausgetauscht werden wie das Chiffrat. Der Schlüsselaustauch stellt damit ein erhebliches Problem beim Einsatz symmetrischer Algorithmen dar.

Ein zweites Problem ergibt sich, sobald mehrere Kommunikationspartner sichere Daten austauschen wollen. Für jedes Paar von Sender/Empfänger ist ein eigener Schlüssel notwendig, so dass jeder von n Teilnehmern $n - 1$ Schlüssel benötigt. Es ergeben sich damit in einem solchen Verbund insgesamt $n * (n - 1)$ verschiedene Schlüssel, die alle erzeugt und sicher verwahrt und verwaltet werden müssen. Dies macht den Einsatz in größeren Gruppen nahezu unmöglich.

4.1.2 Asymmetrische Algorithmen

Um den Problemen symmetrischer Algorithmen zu begegnen, wurde in den 70er Jahren von Whitfield Diffie und Martin Hellman das Konzept asymmetrischer Verschlüsselung eingeführt. Etwa zur gleichen Zeit wurde am MIT von Rivest, Shamir und Adelman der RSA-Algorithmus eingeführt. RSA wird noch heute sehr erfolgreich für die Verschlüsselung von Nachrichten eingesetzt, Diffie-Hellman hat sich zum bevorzugten Algorithmus für den Schlüsselaustausch entwickelt. Dabei werden typischerweise Schlüssellängen von 1024- Bit und darüber verwendet, da prinzipbedingt im Vergleich zu den symmetrischer Verfahren längere Schlüssel nötig sind, um das gleiche Maß an Sicherheit zu erreichen.

Eine jüngere Klasse asymmetrischer Algorithmen verwenden im Raum rotierende elliptische Kurven, weshalb das Verfahren Elliptic Curve Cryptography (ECC) genannt wird. Diese Algorithmen sind einfacher als RSA und Diffie-Hellman mit Integer-Einheiten aktueller Prozessoren zu berechnen und werden daher gelegentlich den anderen beiden vorgezogen.

In [Silv 01] wird die relative Stärke von verschiedenen Algorithmen verglichen (Abbildung 4.1).

Silvermann geht bei seiner Analyse von einem Budget von 10 Millionen US-Dollar aus, um einen Rechner zu bauen, der die Verschlüsselung brechen soll. Der Preis für Speicher wird mit 0,5

US-Dollar pro Mb geschätzt. Aus dem benötigten Speicher für die Berechnungen ergibt sich die Anzahl von Maschinen, die das Problem parallel bearbeiten können. Allerdings reichen die zur Verfügung stehenden 10 Mio. Dollar nicht aus, um 120 Tb Speicher zu kaufen, die aber benötigt würden, um ein RSA-Chiffrat mit einem Schlüssel von 1620 Bit zu brechen, woraus sich die Zahl von 0,16 Maschinen in dieser Zeile errechnet.

Der wichtigste Unterschied zwischen symmetrischer und asymmetrischen Verfahren besteht in der Art der verwendeten Schlüssel. Bei asymmetrischen Algorithmen kommen für Ver- und Entschlüsselung verschiedene, aber mathematisch miteinander verwandte Schlüssel zum Einsatz. Dieses Paar nennt man Public- und Private-Key. Der Public-Key oder öffentliche Schlüssel kann beliebig verbreitet werden und für jeden zugänglich sein. Mit ihm ist das Verschlüsseln von Nachrichten möglich, aber nicht das Entschlüsseln. Dazu wird der Private-Key oder private Schlüssel benötigt, der infolgedessen auch keinesfalls jemand anderem als dem Eigentümer zugänglich gemacht werden darf.

Da der private Schlüssel geheim ist und davon ausgegangen werden kann, dass nur der Eigentümer die Möglichkeit zu seiner Verwendung hat, kann allein dieser damit auch mathematische Operationen ausführen. Dadurch wird es möglich, eine Nachricht mit einem Echtheitszertifikat seines Autors auszustatten. Dieses Zertifikat nennt man digitale Signatur, sie kann nur mit dem privaten Schlüssel erzeugt, aber mit dem öffentlichen Schlüssel verifiziert werden. Eine wichtige Anwendung für digitale Signaturen ist die Authentifizierung von Benutzern.

Das Konzept eines Schlüsselpaares ermöglicht ein wesentlich handlicheres und besser skalierbares Management von Schlüsseln. Wenn n Teilnehmer untereinander kommunizieren wollen, so genügen dafür n verschiedene Schlüsselpaare. Auch das Problem des Schlüsselaustauches lässt sich durch die Verwendung öffentlicher Schlüssel elegant lösen.

Allerdings birgt asymmetrische Kryptographie auch Nachteile:

- Durch die komplexeren Algorithmen und die längeren Schlüssel ist die Durchführung der Operationen wesentlich langsamer als bei symmetrischen Verfahren. Abhängig vom eingesetzten Algorithmus kann es sich dabei um eine Verlangsamung um den Faktor 10 bis 100 handeln.

- Desweiteren sind die bei asymmetrischer Verschlüsselung entstehenden Chiffrate größer als bei symmetrischen Verfahren. Gerade bei der mehrfachen Verschlüsselung großer Datenmengen, wie sie bspw. beim Transport über Tunnel auftreten kann, führt dies zu einer deutlichen Reduzierung des Datendurchsatzes.

Im folgenden soll beispielhaft für asymmetrische Verfahren, der Algorithmus nach Diffie-Hellman vorgestellt werden.

Der Algorithmus von Diffie und Hellmann

Der nach seinen beiden Entdeckern Whitfield Diffie und Martin Hellmann benannte Algorithmus war 1976 der erste Algorithmus, der das Prinzip der asymmetrischen Verschlüsselung realisierte. Er beruht auf der Tatsache, dass die Potenzierung großer Zahlen wesentlich einfacher ist als deren Logarithmisierung.

Beschreibung Das ursprüngliche Ziel von Diffie und Hellmann war dabei das Ermöglichen eines sicheren Schlüsselaustausches für symmetrischen Verschlüsselung über ein unsichere Transportmedium. Zwei Kommunikationspartner Bob und Alice einigen sich auf eine große Primzahl n und eine ganze Zahl g, die ohne Gefährdung der Sicherheit über ein unsicheres Medium ausgetauscht werden können. Beide Partner besitzen jetzt n und g und wählen unabhängig je eine geheime Zahl x (Bob) und y (Alice).

Nun berechnet Bob

$$X = g^x \bmod n \tag{4.1}$$

und sendet X an Alice. Alice berechnet

$$Y = g^y \bmod n \tag{4.2}$$

und schickt das Ergebnis Y an Bob. Jetzt kann Bob

$$k = Y^x \bmod n \tag{4.3}$$

und Alice

$$k' = X^y \bmod n \tag{4.4}$$

berechnen, wobei

$$k = k' = g^{xy} \bmod n \tag{4.5}$$

gilt. Damit ist $k = k'$ der gesuchte geheime Schlüssel.

Mit einer kleinen Erweiterung kann dieses Verfahren auch genutzt werden, um einen Schlüsselaustausch durchzuführen. Dazu generiert jeder Teilnehmer per Zufallsgenerator einen geheimen Schlüssel x und berechnet mit Hilfe der allen Teilnehmern bekannten Primzahl n und der Ganzzahl g den dazugehörigen öffentlichen Schlüssel X als

$$X = g^x \bmod n \tag{4.6}$$

Die öffentlichen Schlüssel können über unsichere Medien ausgetauscht werden. Will nun Bob an Alice eine Nachricht schicken, so ermittelt er aus seinem geheimen Schlüssel x_B und Alice' öffentlichem Schlüssel X_A den zur symmetrischen Verschlüsselung benutzten Schlüssel k wie folgt:

$$k = X_A^{x_B} \bmod n \tag{4.7}$$

Analog kann Alice bei Empfang der mit k verschlüsselten Nachricht den Schlüssel k berechnen.

Sicherheit des Verfahrens Zur Generierung der Schlüssel werden die Primzahl n und die Ganzzahl g an alle Teilnehmer übermittelt. Desweiteren werden zur Verteilung die öffentlichen Schlüssel X_A und X_B versendet. Diese Übertragungen können durch einen Angreifer belauscht werden, allerdings ist die Berechnung der geheimen Schlüssel x_A und x_B aus g, n, X_A und X_B wegen der Schwierigkeit, die Umkehrfunktion zu g^x und g^y für große Primzahlen g zu berechnen, nicht einfach. Die Sicherheit des Verfahrens hängt also wesentlich von der Größe der Primzahl n ab, sie sollte eine Größe von mindestens 1024 oder 2048 Bit haben, um ausreichende Sicherheit zu gewährleisten.

Die Größe von g trägt nichts zur Sicherheit bei, sie muss nur primitiv modulo n sein. Diese Zahl kann daher im ein- bis zweistelligen Bereich ungleich 1 gewählt werden.

Anfällig ist das Verfahren dagegen gegenüber einer Man-in-the-middle-Attacke, wenn ein Angreifer den Schlüsselaustausch nicht nur passiv belauscht, sondern in aktiv manipuliert, und Bobs oder Alice' Schlüssel durch seinen eigenen ersetzt. Wird auch in den Austausch der Nachricht aktiv eingegriffen, so bleibt dieser Angriff ohne weitere Sicherheitsmaßnahmen unentdeckt.

4.1.3 Zusammenfassung

Die wesentlichen Eigenschaften symmetrischen und asymmetrischer Verschlüsselung sind in der folgenden Übersicht zusammengefasst.

Symmetrische Verfahren

- ◇ Gleicher Schlüssel für Ver- und Entschlüsselung
- ◇ Hohe Performance
- ◇ Hohe Sicherheit bei ausreichender Schlüssellänge
- ◇ Nachricht und Chiffrat haben gleiche Länge
- ◇ Schlüsselaustauch problematisch
- ◇ Die Anzahl der nötigen Schlüssel wächst quadratisch zur Anzahl der Teilnehmer
- ◇ Für digitale Signaturen unbrauchbar

Asymmetrische Verfahren

- ◇ Unterschiedliche Schlüssel für Ver- und Entschlüsselung
- ◇ Schlechte Performance
- ◇ Hohe Sicherheit bei ausreichender Schlüssellänge
- ◇ Nachricht wird durch Verschlüsselung vergrößert
- ◇ Schlüsselaustausch einfach
- ◇ Einfaches Schlüsselmanagement
- ◇ Für digitale Signaturen geeignet

4.1.4 Folgerungen

Wie man sieht, haben beide Verfahren ihre Vor- und Nachteile. Besonders kritisch bei symmetrischen Algorithmus ist das Problem des Schlüsselaustausches, die schlechte Skalierbarkeit und die fehlende Brauchbarkeit für digitale Signaturen, während bei asymmetrische Verfahren der hohe rechnerische Aufwand problematisch ist. Glücklicherweise sind aber die Stärken des einen Verfahrens die Schwächen des anderen und umgekehrt. Ideal ist daher der Einsatz einer kombinierten Lösung, die die Performance der symmetrischen Verschlüsselung mit dem einfachen Schlüsselmanagement und der Möglichkeit digitaler Signaturen asymmetrischer Algorithmen vereinigt.

Key-Wrapping

Diese Kombination wird durch Key-Wrapping ([Nash 02]) ermöglicht: Eine Nachricht soll zwischen Sender und Empfänger ausgetauscht werden. Dazu erzeugt der Sender der Nachricht einen zufälligen Schlüssel ausreichender Größe für eine symmetrische Verschlüsselung. Das Problem der Übertragung dieses Schlüssels an den Empfänger wird durch asymmetrische Verschlüsselung gelöst. Es sei vorausgesetzt, dass der Sender bereits im Besitz des öffentlichen Schlüssels des Empfängers ist. Mit diesem öffentlichen Schlüssel wird nun der Schlüssel, der bei der symmetrischen Verschlüsselung verwendet wurde und an den Empfängers übertragen werden soll, verschlüsselt. Das Ergebnis dieser Operation, ein Umschlag für den symmetrischen Schlüssel, wird an die symmetrisch verschlüsselte Nachricht angehängt. Beides, Nachricht und Umschlag, wird nun an den Empfänger übermittelt.

Dank seines privaten Schlüssels kann der Empfänger den Umschlag öffnen und daraus des symmetrischen Schlüssel entnehmen. Mit Hilfe dieses Schlüssels kann er nun die Nachricht selbst entschlüsseln und erhält dadurch den Klartext.

Diese Lösung ist sicher: Ein Angreifer, der auf dem Transportmedium Nachricht und/oder Umschlag abfängt, kann mit beidem nichts anfangen. Die Nachricht ist mit dem symmetrischen Verfahren sicher vor jedem unberechtigten Zugriff. Der Umschlag ist sicher, da der Angreifer nicht im Besitz des für die Entschlüsselung nötigen privaten Schlüssels des Empfängers ist.

Diese Lösung ist performant: Die Verschlüsselung der Nachricht erfolgte symmetrisch, so dass Nachricht und Chiffrat die gleiche Größe haben und die Ver- und Entschlüsselung schnell berechnet werden können. Der Umschlag enthält nur einen Schlüssel von ca. 128 Byte Länge, daher kann der Aufwand für Ver- und Entschlüsselung vernachlässigt werden.

Dennoch, ein letztes Problem bleibt: Wie kann der Sender sicher sein, dass der Public-Key des Empfängers, den er aus einer öffentlich zugänglichen Quelle wie einen Keyserver oder per unverschlüsselte Email erhalten hat, tatsächlich vom Empfänger stammt und keine Fälschung ist?

4.2 Konzept einer wartungsarmen PKI

Das vorgestellte Konzept des Key-Wrapping ist zwar sicher gegen passives Lauschen, alledings nicht gegen aktive Angriffe, bei denen ein Dritter sich als Sender oder Empfänger ausgibt (Man-in-the-middle-Attack). Dabei wird vom Angreifer die Übertragung der öffentlichen Schlüssel abgefangen und er gibt seinen eigenen Public-Key an Sender und/oder Empfänger weiter. Diese benutzen die gefälschten Schlüssel, was dem Angreifer bei einem aktiven Eingreifen in die Kommunikationsverbindung die Möglichkeit gibt, den Umschlag zu öffnen und den symmetrischen Schlüssel zu entnehmen. Wenn er den Umschlag mit Hilfe der echten Public-Keys der Kommunikationspartner neu erstellt und weiterleitet, bleibt dieser Angriff unbemerkt.

Vertrauensstellungen

Dieses Problem kann mit einer Public Key Infrastructure (PKI) angegangen werden. Die wichtigste Aufgabe einer PKI ist das Verwalten von vertrauenswürdigen Identitäten, die zusammen mit kryptographischen Mechanismen zur Authentifizierung und Autorisierung eines Nutzers verwendet werden können. Das Herstellen solcher vertrauenswürdigen Identitäten erfolgt mit Hilfe von Vertrauensstellungen. Dabei wird die aus dem Leben bekannte Transitivität von Vertrauen zwischen Menschen formalisiert und dazu genutzt, um die Vertrauenswürdigkeit eines unbekannten Schlüssels zu berechnen. Ein Beispiel dafür ist die Erstellung von beglaubigten Abschriften wichtiger Dokumente durch Notare: Eine einfache Kopie eines Dokuments wird dadurch vertrauenswürdig, dass eine vertrauenswürdige Person, der Notar, die Korrektheit der Abschrift bestätigt. Der Empfänger vertraut zwar nicht dem Überbringer der Abschrift, aber er vertraut dem Notar (als Person oder Rechtsinstitut), und somit vertraut er auch der Abschrift. Vertrauen wurde also von einer vertrauenswürdigen Person übertragen.

Übertragen auf Public-Key-Verschlüsselung wird Vertrauen durch das Signieren von Schlüsseln durch vertrauenswürdige Personen oder Organisationen hergestellt. Die Vertrauenswürdigkeit eines öffentlichen Schlüssels und der dazugehörigen Identität ergibt sich damit aus der Vertrauenswürdigkeit der zu dieser Identität gehörenden Signaturen. Die Kombination aus öffentlichem Schlüssel, Identität und Signaturen wird als Zertifikat beszeichnet. Eine Veränderung von Teilen des Zertifikats, z. B. des öffentlichen Schlüssels, führt unmittelbar dazu, dass die dazugehörigen Signaturen, und damit das Zertifikat als ganzes, ungültig werden. Manipulationen durch Angreifer am Zertifikat selbst sind daher ausgeschlossen, solange die zugrundeliegenden Krypto-Algorithmen sicher vor Manipulationen sind.

Zertifikate

Als Zertifikat bezeichnet man eine Datenstruktur, die u. a. folgende Informationen enthält:

- ◇ Inhaber: In diesem Feld werden verschiedene Identifikationsmerkmale des Eigentümers des Zertifikats angegeben. In den meisten Fällen wird hier unter anderem der Name des Eigentümers aufgeführt. Dieses Feld muss in jedem Zertifikat enthalten sein.
- ◇ Public Key: Dieses Feld enthält den Public Key des Eigentümers. Auch dieses Feld ist unbedingt notwendig.
- ◇ Gültigkeit: Dieses Feld bezeichnet die zeitliche Gültigkeit des Zertifikats.
- ◇ Digitale Signaturen: Um zu bestätigen, dass die im Feld Inhaber angegebene Identität korrekt ist, enthält ein Zertifikat eine oder mehrere digitale Signaturen, deren Aussteller sich zweifelsfrei davon überzeugt haben, dass die Angaben über den Inhaber korrekt sind.
- ◇ Weitere Attribute: Ein Zertifikat kann noch weitere Attribute, wie zum Beispiel eine Seriennummer oder Informationen über den erlaubten Einsatzzweck des Zertifikats enthalten.

Ein Zertifikat ist gültig, wenn die enthaltenen digitalen Signaturen gültig sind. Im Laufe der Zeit wurden verschiedene Standards entwickelt, um den Inhalt eines Zertifikats zu vereinheitlichen. Der wichtigste davon ist X.509.

Netzwerkmodell

Dabei unterscheidet man zwei verschiedene Modelle: Im Netzwerk-Modell wird das Zertifikat eines Nutzers von möglichst vielen anderen Nutzern signiert, die die Echtheit der Identität aus erster Hand bezeugen können. Erhalten dritte dieses Zertifikat aus einer nicht vertrauenswürdigen Quelle, so ergibt sich die Vertrauenswürdigkeit dieses Zertifikats aus der Summe des Vertrauens, das den einzelnen Signatur-Urhebern entgegengebracht wird, wobei dieses Vertrauen ebenfalls aus erster Hand stammen muss, um Manipulationen und Missbrauch auszuschließen. Ist dem Empfänger eines Zertifikats keiner der Signatur-Urheber bekannt, so ist die Vertrauenswürdigkeit dieses Zertifikats Null. Jeder Teilnehmer dieses Netzwerks kann also die Vertrauenswürdigkeit eines Zertifikats in gleichberechtigter Weise erhöhen, wenn er die Gültigkeit des Zertifikats zweifelsfrei bestätigen kann. Dieses Modell funktioniert daher nur mit einer großen Nutzerbasis, wobei ein reger Austausch zwischen einzelnen Gruppen vorhanden sein muss, um der Entstehung von ”Vertrauensinseln” vorzubeugen.

Hierarchisches Modell

Im hierarchischen Modell dagegen hängt die Gültigkeit eines Zertifikats ausschließlich von der Signierung durch eine zentrale Zertifizierungsstelle (Certification Authority - CA) ab. Diese Wurzel-Zertifizierungsstelle (Root-CA) kann die Zuständigkeit für Benutzergruppen nach lokalen oder organisatorischen Gegebenheiten an weitere untergeordnete Zertifizierungsstellen delegieren, so dass Vertrauensstellungen baumartig organisiert sind. Die Nutzer der Zertifikate benötigen zum Verifizieren der CA-Signaturen eine Kopie des öffentlichen Schlüssels der CA, die absolutes Vertrauen genießt. Gelingt es einem Angreifer, das CA-Zertifikat eines Nutzers zu manipulieren, so eröffnet sich ihm die Möglichkeit, diesem Nutzer gefälschte Zertifikate unterzuschieben.

4.2.1 Aufgaben einer PKI

Zusammenfassend erfüllt eine PKI folgende Aufgaben:

- Schlüsselerzeugung
- Überprüfung der ursprünglichen Identität des Nutzers
- Lifecycle-Management, d. h. Ausgabe, Widerrufen und Erneuern von Zertifikaten
- Verteilung der Zertifikats an die Nutzer
- Archivierung von Zertifikaten und Schlüsseln

4.2.2 Schlüsselerzeugung

Verfahren zur Erzeugung von Schlüsseln wurden im Kapitel 4.1.2 vorgestellt. Bezüglich des Orts der Schlüsselerzeugung gibt es zwei Varianten: Schlüssel können von einem zentralen Schlüssel-Generierungssystem oder dezentral auf den Maschinen der Benutzer erzeugt werden. Die zentrale Erzeugung vereinfacht viele Aspekte des Managements von Schlüsseln, wie zum Beispiel Archivierung und Wiederherstellung von Schlüsseln (s. u.). Die Architektur wird wesentlich vereinfacht, da die Komponenten zum Schlüsselmanagement nur einmal bereitgestellt werden müssen. Problematisch beim zentralen Ansatz ist allerdings, dass dies zur Einführung einer hochsensiblen Sicherheitskomponente führt. Für Angreifer ist diese Komponente ein offensichtliches Ziel, mit deren Korrumpierung die gesamte Sicherheitsarchitektur auf einen Schlag durchbrochen wird.

Viele PKI-Architekturen fordern daher ein gemischtes Modell, bei dem Schlüssel, die für Verschlüsselungszwecke eingesetzt werden, zentral, Schlüssel für digitale Signaturen aber dezentral am Client erzeugt.

4.2.3 Überprüfung der ursprünglichen Identität des Nutzers

Zur Bestätigung, dass die im Zertifikat angegebene Identität korrekt ist, muss ein Prozess etabliert werden, der garantiert, dass das Zertifikat eines Nutzers (genannt Antragsteller) nur dann signiert wird, wenn der Aussteller der Signatur zweifelsfrei die Zusammengehörigkeit von Public Key und Zertifikatsinhaber nachvollzogen hat.

Bei Verwendung des Netzwerkmodells erfolgt diese Überprüfung oft informell, indem das zu signierende Zertifikat bspw. persönlich unter Freunden weitergegeben und signiert wird. Dabei ist darauf zu achten, dass eine Verfälschung des Zertifikats auf dem Transportweg ausgeschlossen wird. Dies kann bspw. durch die Verwendung sicherer Transportmedien wie Disketten oder durch das manuelle Überprüfen von Prüfsummen (sog. Fingerprints) erfolgen, wobei zumindest die Prüfsumme sicher transportiert werden muss.

Bei Verwendung eines Hierarchischen Modells ist für diesen Prozess eine separate Organisationseinheit, die sog. Registrierungsstelle (Registration Authority - RA) zuständig. Hier wird die Identitätsprüfung idealerweise von einem menschlichen Operator vorgenommen, der die Identität anhand eines Personalausweises überprüft. Voraussetzung für die korrekte Identifizierung ist in diesem Fall natürlich Vertrauen in die korrekte Identitätsprüfung bei Ausstellung dieses Ausweises.

4.2.4 Ausgabe, Widerrufen und Erneuern von Zertifikaten

Nachdem die Identität eines Antragstellers zweifelsfrei festgestellt wurde, wird von der Certification Authority (CA) basierend auf den öffentlichen Schlüssel des Antragstellers ein Zertifikat erzeugt und mit dem privaten Schlüssel der CA signiert. Im Netzwerkmodell werden Zertifikate vom Antragsteller erst selbst erzeugt und signiert, später kann die Rolle der CA von allen wahrgenommen werden, die aus eigenem Wissen die Echtheit des Zertifikats nachvollziehen können. Nach dem Signieren des Zertifikats durch einen oder mehrere Dritte steht dann ein vollständiges Zertifikat zur Verfügung.

Für besondere Fälle, wie zum Beispiel dem Verlust eines Zertifikats oder des dazugehörigen privaten Schlüssels muss es die Möglichkeit geben, das Zertifikat eines Nutzers für ungültig zu erklären. Weiterhin sollte das Zertifikat eines Nutzers immer dann automatisch ungültig werden, wenn der Benutzer aus der zentralen Benutzerdatenbank entfernt, beispielsweise weil er die Organisation, für die das Zertifikat erstellt wurde, verlässt. Die Sperrung eines Zertifikats geschieht, indem die CA oder eine andere zentrale Instanz davon unterrichtet wird, dass das Zertifikat nicht mehr benutzt werden darf. Dort wird eine Sperrliste (Certificate Revocation List - CRL) verwaltet, in die gesperrte Zertifikate eingetragen und veröffentlicht werden. Während der Überprüfung eines Zertifikats wird das zu prüfende Zertifikat in der CRL gesucht, so dass gesperrte Zertifikate erkannt werden.

Wenn das Zertifikat mit einer begrenzten Gültigkeitsdauer versehen wurde, so ist bei Ablauf dieser Gültigkeit eine Erneuerung des Zertifikats notwendig. Bei der Erneuerung können auch Attribute des Zertifikats geändert werden. Dadurch kann in sensiblen Bereichen Vorsorge getroffen werden, dass kompromittierte Zertifikate automatisch bei Ablauf ihre Gültigkeit verlieren. Ebenso wird verhindert, dass mit ewig gültigen Zertifikaten Missbrauch getrieben wird, beispielsweise durch ausgeschiedene Mitarbeiter, deren Zertifikate beim Ausscheiden nicht automatisch widerrufen wurden.

4.2.5 Verteilung der Zertifikate an die Nutzer

Ist ein Zertifikat erzeugt, so muss es allen potentiellen Nutzern verfügbar gemacht werden. Bei einer kleinen Anzahl von Zertifikaten kann dies durch die Weitergabe von einem Nutzer an den anderen erfolgen, für größere Lösungen ist oft die Verwendung eines Keystores sinnvoll. Dabei handelt es sich um eine Datenbank oder einen Verzeichnisdienst, aus dem mit Hilfe eines bestimmten Protokolls Zertifikate ausgelesen werden können. Der Zugriff auf diese Datenbank ist allen Nutzern zu ermöglichen, die an der sicheren Kommunikation teilnehmen wollen.

Es wurden verschiedene APIs entwickelt, um Anwendungen standardisierten Zugriff auf Keystores zu bieten.

Unter Umständen schreibt die Sicherheitspolicy einer Organisation vor, dass der Keystore nicht öffentlich ist, und der Zugriff nur bestimmten Personengruppen ermöglicht werden darf. In diesen Fällen ist ein Sicherheitskonzept zum Zugriff auf den Keystore zu entwickeln. Dabei kann bspw. eine Authentifizierung mit Passwörtern vorgeschrieben, die allerdings wegen der bekannten Schwächen von Passwörtern nur geringen Sicherheitsanforderungen genügt.

4.2.6 Schutz von Zertifikaten und privaten Schlüsseln

Während der Keystore des letzten Kapitels nur öffentliche Schlüssel enthält, die üblicherweise wenig schutzwürdig sind, da sie ausschließlich zur Verschlüsselung und zur Überprüfung digitaler Signaturen verwendet werden können, ist die sichere Speicherung besonders des privaten Schlüssels beim Anwender ein Problem ganz eigener Art. Üblicherweise wird ein Anwender sein Public-/Private-Key-Paar auf der Festplatte seines Rechners speichern. Erhält ein Angreifer Zugriff auf diese Festplatte, so kann er sich ohne Probleme der Dateien bemächtigen, die Zertifikat und privaten Schlüssel enthalten. Oft wird daher der private Schlüssel durch ein Passwort geschützt, mit dem der Schlüssel selbst verschlüsselt wird (mit Hilfe eines symmetrischen Algorithmus).

Bei höheren Sicherheitsanforderungen ist die Verwendung einer Smartcard möglich. Dadurch wird der private Schlüssel auf ein Medium ausgelagert, das der Benutzer in einfacher Weise getrennt von seinem Rechner aufbewahren kann. Weiteren Schutz ermöglicht die Verwendung eines Passworts für die Daten auf der Smartcard.

4.2.7 Archivierung von Zertifikaten und Schlüsseln

Ebenso wie Nutzer oft ihre Passwörter vergessen, die daraufhin erneuert werden, können sie ihre Schlüssel verlieren. In einfachen Szenarien ist auch hier die Erzeugung neuer Schlüssel und Zertifikate möglich. Allerdings kann es sinnvoll, die privaten Schlüssel der Anwender zentral zu archivieren, um sie bei Verlust wiederherstellen zu können. Denkbar ist hier z. B. der Fall, dass vertrauliche Dokumente unbrauchbar werden, wenn der zum Zugang benötigte Schlüssel verloren geht. Immer wieder verlangen auch Regierungsstellen Zugriff auf geheime Schlüssel, um Überwachungsmaßnahmen zu vereinfachen. Für solche Anforderungen wurden Schlüsselarchivierungssysteme entwickelt, die alle privaten Schlüssel zu ausgegebenen Zertifikaten speichern. Dabei handelt es sich natürlich um ein hochsensibles System, dessen Kompromittierung einen Zusammenbruch des gesamten Sicherheitskonzepts zur Folge hätte.

Besonders problematisch ist das Archivieren von Schlüsseln, die zur Erzeugung digitaler Signaturen vorgesehen sind: Wenn auch nur die Möglichkeit besteht, dass nicht nur der rechtmäßige Eigentümer Zugriff auf einen solchen Schlüssel hat, sondern möglicherweise auch der Administrator des Keystores oder ein Angreifer, so wird die Anforderung der Nicht-Abstreitbarkeit nicht mehr realisiert.

Ist die Schlüsselerzeugung zentralisiert, so muss der Schlüsselerzeugungsdienst jeden erzeugten Schlüssel an das Archiv weitergeben. Werden Schlüssel clientseitig erzeugt, so muss jede Applikation ebenfalls alle erzeugten Schlüssel an das Archiv übermitteln. Dabei ist auf sicheren Transport der Daten zu achten.

4.3 Sicherheitsfeatures von IPv6

Aufgrund der großen Verbreitung IP-basierter Kommunikation in den 90er Jahren haben gleichzeitig auch die Angriffe auf diese Technologie zugenommen. Daher war bei der Entwicklung von IPv6 das Thema Sicherheit von Anfang an zentraler Bestandteil aller Überlegungen, während das 20 Jahre ältere IPv4 nahezu frei von Sicherheitsüberlegungen entstand.

Vom IETF wurde daher eine eigene Arbeitsgruppe gegründet, die sich ausschließlich mit Verfahren zur Sicherung der Kommunikation beschäftigt (IPSEC-Gruppe). Die dort entwickelten Mechanismen wurden dabei so konzipiert, dass sie gleichzeitig für IPv4 und IPv6 eingesetzt werden können. Sie decken alle Bedürfnisse auf den Gebieten Verschlüsselung und Authentifizierung ab und werden unter dem Namen IPSec (s. [KeAt 98b]) zusammengefasst.

Viel diskutiert wurde die Frage, ob Sicherheit nicht besser auf einer anderen OSI-Schicht umgesetzt werden sollte. Gegen die ausschließliche Realisierung in der Transportschicht spricht allerdings, dass damit nicht allen Angriffen begegnet werden kann, so würden beispielsweise Manipulationen an der Adressierung nicht erkannt. Andererseits haben Verfahren, die direkt auf der Kommunikationsleitung, also unterhalb IP ansetzen, den Nachteil, dass jeder einzelne Router auf der Strecke den Datenverkehr erst ent- und dann wieder verschlüsseln muss. Dies kann zu Performance-Problemen auf Hochgeschwindigkeitsstrecken führen, außerdem setzt ein solchen Verfahren voraus, dass allen Routern auf dem Weg vertraut wird.

Eine Realisierung von Sicherheitsfunktionen auf IP-Ebene ist daher nur folgerichtig. Die Möglichkeit zur Verschlüsselung auf anderen Ebenen der Kommunikation bleibt dabei weiterhin bestehen ([Ditt 02]).

4.3.1 Grundlagen

Die von IPSec definierten Verfahren umfassen folgende Funktionalitäten:

- Verschlüsselung der Nachricht, um die enthaltenen Informationen gegen Angreifer zu schützen.
- Authentifizierung der Nachricht mit Hilfe einer Prüfsumme, um die Verfälschung durch Angreifer zu verhindern. Dabei wird die Nachricht selbst im Klartext übermittelt und nur die Prüfsumme mit einem privaten Schlüssel signiert. Dadurch ist der Datendurchsatz deutlich höher als beim Verschlüsseln der gesamten Nachricht. Durch die Signatur kann gleichzeitig auch die Herkunft der Nachricht sowie ihr Absender sicher identifiziert werden.

Die Basis für die in IPSec integrierten Verfahren sind dabei sog. Sicherheitsbeziehungen. Eine solche Beziehung fasst die Parameter einer (unidirektionalen) Verbindung zusammen und kann mit einem Index im Header einer Nachricht angesprochen werden. Für jede Verbindung sind daher zwei Sicherheitsbeziehungen notwendig. Jede Beziehung definiert folgende Werte:

- IP-Adresse des Ziels
- angewandtes Verfahren zur Verschlüsselung oder Authentifizierung
- aktueller Session-Key
- verfahrensspezifische Parameter

◇ zeitliche Gültigkeit des Session-Keys

Jedes Endgerät kann zu jedem Zeitpunkt viele verschieden Sicherheitsbeziehungen unterhalten, abhängig von der Anzahl der gesicherten Verbindungen. Welcher Beziehung ein IP-Paket zuzuordnen ist, wird durch einen Sicherheitsindex im Optionsheader des Pakets angegeben. Diese Indizes werden zufällig vergeben, um Angriffsmöglichkeiten zu minimieren. Der Indexwert 0 bedeutet, dass noch keine Sicherheitsbeziehungen besteht.

4.3.2 Verschlüsselung bei IPv6

IPv6 bietet zwei Möglichkeiten der Verschlüsselung von Nachrichten: Der erste Ansatz beschränkt sich auf die Verschlüsselung der Nutzdaten, der Paketheader bleibt unverschlüsselt (Transport-Mode). Der zweite Ansatz die Verschlüsselung des kompletten Pakets inklusive aller Header mit Hilfe von IPv6-over-IPv6-Tunneln. Dabei wird das Paket verschlüsselt und mit einem Header versehen zum Empfänger geschickt (Tunnel-Mode).

Die erste Methode wird verwendet, wenn nur ein Teil der Kommunikation, zum Beispiel die Übertragung von Passwörtern oder das Abholen von Mails, gesichert werden soll. Der restliche Datenverkehr, beispielsweise Web-Seiten, bleiben unverschlüsselt. Durch die Verwendung von Klartextheadern lassen sich auch weiterhin alle Verbindungsdaten zwischen Sender und Empfänger mitverfolgen.

Der Verschlüsselungsheader (ESP-Header) wird im Header nach allen Optionen eingefügt, die von Ende zu Ende gültig sind, sowie nach einem eventuell vorhandenen Authentifizierungsheader.

Wenn für eine Verbindung noch keine Sicherheitsbeziehung etabliert wurde oder der Session-Key nicht mehr gültig ist, so muss das gesamte Paket verworfen werden. Ist die Entschlüsselung des Pakets erfolgreich, so erfolgt anhand der nun entschlüsselten TCP- oder UDP-Header die Entscheidung über die weitere Verarbeitung des Pakets.

Die Verschlüsselung mit Hilfe eines Tunnels ist angezeigt, wenn die gesamte Kommunikation zwischen zwei Endpunkten gesichert werden soll. Dies ist beispielsweise sinnvoll, um den Datenstrom zwischen Hauptstelle und Filiale einer Firma zu verschlüsseln. Bei dieser Variante sind auch die Verbindungsdaten, wie Quell- und Zieladresse eines Pakets, von außen nicht mehr einsehbar. Daher kann der Tunnel-Mode nicht nur zwischen Endpunkten, sondern auch zwischen Routern eingesetzt werden.

Bei der Verschlüsselung eines Pakets wird erst ein neuer IPv6-Header für den Transport zwischen den Tunnelendpunkten konstruiert. Dieser trägt als Absender- und Empfängeradresse die Adressen der jeweiligen Tunnelendpunkte. Zusätzlich werden die Optionen des Ursprungsheaders, zum Teil modifiziert, in den neuen Header kopiert (s. [KeAt 98c]). Anschließend folgt der ESP-Header, gefolgt vom originalen Paket. Das Paket wird abgeschlossen durch den ESP-Trailer, der Padding, der Länge des Padding sowie das Next-Header-Feld enthält.

Ebenso wie beim alternativen Verfahren muss auch hier das Paket verworfen werden, wenn die Entschlüsselung scheitert. Die Spezifikation schreibt vor, dass alle Fehler in der Entschlüsselung im Logfile festgehalten werden müssen, damit Probleme später nachvollzogen werden können. Kann das Tunnelpaket entschlüsselt werden, so wird der Tunnelheader verworfen und das enthaltene IP-Paket kann weiterverarbeitet werden. Aus implementierungstechnischen Gründen weicht das Format des Verschlüsselungsheaders von den gewohnten Formaten ab. Er besteht aus dem Sicherheitsindex, einer Sequenznummer, der verschlüsselten Nutzlast sowie den Daten zur Authentifizierung (falls vorhanden). Vor der Entschlüsselung ist die Authentifizierung zu überprüfen, falls der entsprechende Block existiert. Aus dem 32bittigen Sicherheitsindex kann der Empfänger durch Vergleich mit seiner Tabelle der aktuellen Sicherheitsbeziehungen die Parameter, wie Verfahren, Session-Key usw. ermitteln.

Im Standard wird kein konkreter Algorithmus zur Verschlüsselung vorgeschrieben, jede Implementierung muss aber zumindest das Verfahren DES-CBC nach [MaDo 98] unterstützen. Selbstverständlich können alternative Verfahren nur dann eingesetzt werden, wenn bei Endpunkte der Kommunikation sie unterstützen. Der Aufbau des ESP-Headers ist in Abb. 4.1 dargestellt.

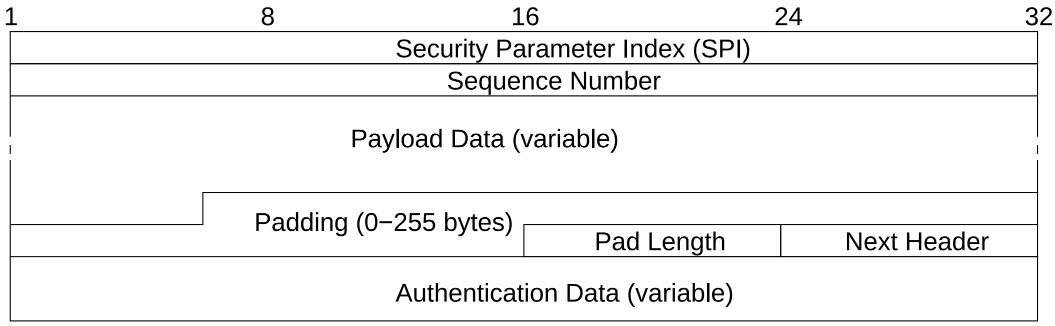

Abbildung 4.1: Format des ESP-Headers

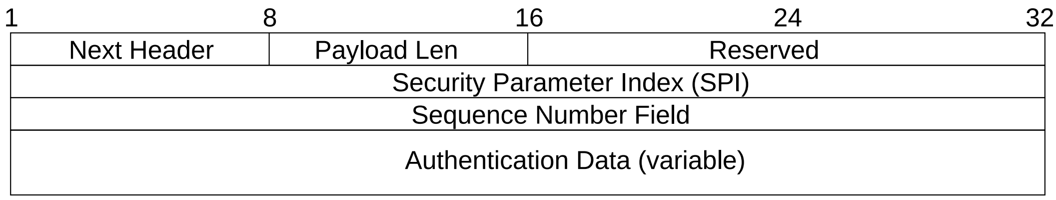

Abbildung 4.2: Format des AH-Headers

4.3.3 Authentifizierung bei IPv6

Zweck der Authentifizierung bei IPv6 ist die Sicherstellung der Identität des Absenders sowie der Integrität der übertragenen Nutzlast. Dadurch soll die Verfälschung einer Nachricht verhindert werden, wobei kein Schutz gegen Sniffing-Attacken gefordert wird. Zusätzlich erlaubt die Authentifizierung die Überprüfung der Korrektheit der Absenderadresse als Schutz gegen IP-Spoofing, ein bei IPv4 weit verbreitetes Problem, das gerade das Feststellen der Urheberschaft von beispielsweise Portscans oder DoS-Attacken erschwert. Die Definition des Verfahrens erfolgte in [KeAt 98a].

Grundlage dieser Authentifizierung sind die Verfahren zur Erstellung digitaler Signaturen, wie sie teilweise in Kapitel 4.1.2 vorgestellt wurden. Da sich diese Verfahren auf das Berechnen und Verschlüsseln von Prüfsummen beschränken, die Nutzlast selbst aber im Klartext übertragen wird, ist eine Authentifizierung sehr performant zu implementieren.

Ebenso wie Verschlüsselung müssen alle Implementierungen von IPv6 Verfahren zur Authentifizierung unterstützen. Der Standard schreibt dabei nicht vor, welches konkrete Verfahren dazu benutzt werden soll. Es muss aber zumindest das Verfahren HMAC (Keyed Hashing for Message Authentication) nach [KBC 97] mit den Hashingalgorithmen MD5 (Message Digest Algorithm Nr. 5) nach [Rive 92] und SHA-1 (Secure Hash Algorithm Version 1) nach [ErJ 01] in jeder Implementierung zur Verfügung stehen.

Der Aufbau des Authentifizierungsheaders entspricht dabei dem üblichen Format für IPv6-Optionen(s. Abb. 4.2). Nach dem Next-Feld, das auf den nächsten Header oder die Nutzdaten verweist, folgt ein Feld, das die Länge des Authentifizierungsheaders in Einheiten zu 32 Bit angibt. Diese Länge wird dabei ohne Berücksichtigung des Next- und des Length-Feldes angegeben, es werden also von der tatsächlichen Länge 2 Byte abgezogen. Der Authentifizierungsindex dient wiederum zur Auswahl der Sicherheitsbeziehung, die auf der zugehörigen Verbindung Anwendung findet. Der jeweils gültige Wert wird beim Verbindungsaufbau vom Zielsystem vorgegeben.

Im Feld Sequenznummer werden alle Pakete aufsteigend durchgezählt. Ein Überlauf dieses 32-

Bit-Feldes ist verboten; werden mehr als $2^{32} - 1$ Pakete innerhalb einer Verbindung übertragen, so muss eine neue Sicherheitsbeziehung aufgebaut werden.

Das Feld Parameter enthält Werte, die abhängig vom gewählten Authentifizierungsverfahren sind. Hier werden Parameter, Prüfsummen usw. abgelegt. Der Inhalt dieses Feldes muss auf eine 32-Bit-Grenze aufgefüllt werden.

Gesichert werden mit der Authentifizierung alle Daten im Paket ab und inklusive dem die Authentifizierung definierenden Header. Das heißt, dass auch Optionsheader, die dem Authentifizierungsheader folgen, bei der Prüfsummenberechnung berücksichtigt werden. Alle von Routern auf der Strecke veränderlichen Felder, wie zum Beispiel Teile des Flow-Labels, müssen vor der Überprüfung der Authentifizierung auf Null gesetzt werden.

Wird beim Empfänger eine Manipulation des Pakets festgestellt, so dass die Authentifizierung nicht nachvollzogen werden kann, so muss der Empfänger dieses Paket verwerfen und einen Eintrag ins Logfile vornehmem. Es ist auf keinen Fall zulässig, den Absender durch ICMP-Nachrichten o. ä. von diesem Fehler zu benachrichtigen, da dies Kompromottierungsmöglichkeiten für Angreifer eröffnen könnte.

In Spezialfällen kann die Anwendung von Authentifizierung auch in Kombination mit Verschlüsselung erfolgen. Während Verschlüsselung für die Sicherung der übertragenen Daten vor Sniffing-Attacken sorgt und die Unverfälschtheit der Nutzlast sicherstellt, garantiert die Anwendung von Authentifizierung in diesen Fällen die korrekte Identifizierung der Kommunikationspartner.

4.3.4 Schlüsselverwaltung

Alle zusammen mit IPSec vorgesehenen Verfahren verwenden zur Verschlüsselung und Authentifizierung symmetrische Algorithmen. Die Standards legen aber keinen Mechanismus fest, wie diese Schlüssel ausgetauscht werden können. Viele Implementierungen verwenden zum Schlüsselaustauch das Verfahren von Diffie und Hellman (4.1.2), wobei zur Verwaltung der dafür nötigen öffentlichen Schlüssel ein eigener Mechanismus vorgesehen werden muss. Die Grundlagen für solche Mechanismen wurden in Kapitel 4.2 besprochen.

Kapitel 5

Implementierungskonzept

Nachdem in den vorherigen Kapiteln die zur Verfügung stehenden Technologien betrachtet und erläutert wurden, ist es nun möglich, die Technologien zur Lösung in Kapitel 2 dargestellen Probleme zu nutzen.

Es soll das Protokoll IPv6 eingeführt werden, allerdings muss dazu soweit möglich die vorhandene Hard- und Software eingesetzt werden. Dabei soll die Einführung von IPv6 aber keinesfalls auf Kosten der bereits vorhandenen Funktionalität gehen, das bedeutet, dass alle Dienste, die vor der Einführung von IPv6 zur Verfügung standen, auch nach der Umstellung wie gewohnt genutzt werden können. Als Voraussetzung für die Einführung von IPv6 muss ein Adressraum ausgewählt und auf die vorhandenen Endgeräte aufgeteilt werden. Darüberhinaus ist zu klären, wie das Management dieses Adressraums gelöst werden kann, wie also die konkrete Zuordnung von Adressen zu Endgeräten sowie deren Konfiguration erfolgen soll. Desweiteren muss eine Lösung gefunden werden, wie mobile Nutzer an das Institutsnetz angebunden werden können. Dabei ist besonderer Wert auf die Sicherheit der Lösung zu legen.

5.1 Transition des IFI-Netzes

Vor der Umstellung des Institutsnetzes ist zu klären, welche Adressen zur Verfügung stehen und wie sie den Endgeräten zugewiesen werden sollen. Dabei soll die Organisationsstruktur des Instituts auf eine analoge Adressstruktur abgebildet werden.

Verwendete Software

Um im Rahmen dieser Arbeit eine erste Implementierung durchzuführen, wird IPv6-fähige Software benötigt. Diese Software teilt sich auf in einen IPv6-fähigen Protokollstack, der Teil des Betriebssystems ist, und andererseits in Anwendungsprogramme.

Bei Betriebssystem fiel die Wahl auf die USAGI-Implementierung von Linux, einerseits wegen der großen Erfahrung, die alle Administratoren im Institut mit Linux haben, andererseits aufgrund der Tatsache, dass der überwiegende Teil der Endgerät im Institut bereits mit Linux betrieben wird. Anders als die im Linux-Standardkernel enthaltene Implementierung wird das USAGI-Projekt aktiv weiterentwickelt und enthält trotz ihres noch experimentellen Status bereits die meisten der vom IETF spezifizierten Features.

Bei den Anwendungen sind verschiedene Kategorien zu unterscheiden:

- Anwendungen ohne Netzwerkfunktionen
- Anwendungen mit Netzwerkfunktionen

Letztere unterscheiden sich darin, ob die Entwickler bereits Code zur Unterstützung von IPv6 eingearbeitet haben oder nicht. Eine Aufstellung, die die wichtigsten Produkte auf ihre IPv6-Fähigkeiten untersucht, findet sich in Kapitel 6.

Adressraum	Zweck	Anzahl /64-Netze
fec0:0:0:1::/64	CIP (Server und Workstations)	1
fec0:0:0:2::/64	X-Terminals Theresienstraße	1
fec0:0:0:3::/64	Notebookarbeitsplätze CIP	1
fec0:0:0:4000::/50	Lehrstuhlnetze	2^{14}

Tabelle 5.1: Aufteilung des Site-Local-Adressraums am Institut

Adressraum	Zweck	Anzahl /64-Netze
3ffe:400:30:1::/64	CIP (Server und Workstations)	1
3ffe:400:30:2::/64	X-Terminals Theresienstraße	1
3ffe:400:30:3::/64	Notebookarbeitsplätze CIP	1
3ffe:400:30:4000::/50	Lehrstuhlnetze	2^{14}
3ffe:400:30:8000::/49	Remote-User	2^{15}

Tabelle 5.2: Aufteilung des 6bone-Adressraums am Institut

5.1.1 Dual IP-Layer

Um trotz der Einführung von IPv6 vollständige Kompatibilität zu IPv4 zu gewährleisten, sieht Abschnitt 3.1.7 die Verwendung von Dual-IP-Layern vor. Das bedeutet, dass alle Endgeräte am Institut eine Implementierung beider Protokollstacks erhalten. Die Folge davon ist, dass Anwendungen, diefür IPv4 programmiert wurden und keinerlei Kenntnisse von IPv6 haben, nicht beeinträchtigt werden und so problemlos weiterverwendet werden können.

5.1.2 Adresszuteilung im Institut

Entsprechend Abschnitt 3.1.2 hat bei IPv6 jede Schnittstelle eines jeden Endgeräts zumindest eine Link-Local-Adresse. Diese Adressen werden bei Aktivierung des IPv6-Protokollstacks automatisch generiert und sind nur zur Kommunikation von Endgeräten am selben Link zu verwenden. Das bedeutet, dass auch im Rahmen dieser Arbeit Link-Local-Adressen von den Endgeräten autokonfiguriert werden. Weitere Schritte zur Konfiguration dieses Adressraums sind nicht notwendig.

Weiterhin können Schnittstellen mit beliebigen Site-Local- oder Global-Adressen ausgestattet werden. Bereits zum jetzigen Zeitpunkt wird im CIP-Pool ein privates Netz (Adressbereich 192.168.0.0/16) verwendet, um CIP-interne Dienste abzuwickeln. Durch Verwendung von Site-Local-Adressen kann diese Funktionalität exakt abgebildet werden.

Zur Adressierung von Endgeräten innerhalb des Instituts steht damit das Site-Local-Präfix fec0::/48 zur Verfügung. Damit lassen sich $2^{64-48} = 65536$ verschiedene Netze der Größe /64 adressieren. Eine vorgeschlagene Aufteilung dieses Adressraums ist in Tabelle 5.1 aufgeführt. Den Lehrstühlen stehen insgesamt $2^{14} = 8192$ Netze zur Verfügung. Für die einzelnen Bereiche des CIP-Pools wurden insgesamt vier /64-Netze vorgesehen. Die in der Tabelle nicht aufgeführten Adressbereiche sind unbelegt.

Ersetzt man das Site-Local-Präfix fec0::/48 durch ein global geroutetes Präfix, so erhält man eine Adressaufteilung für globale Adressen. Zusätzlich kann aus einem solchen globalen Adressraum ein Präfix der Länge /49 für die Anbindung von Remote-Nutzern vorgesehen werden. Damit können Remote-Usern insgesamt $2^{15} = 16384$ /64-Netze zur Verfügung gestellt werden. Angesichts von ca. 4000 Benutzern, die derzeit am CIP-Pool aktiv sind, ist somit genügend Spielraum zur vollständigen Versorgung aller Mitarbeiter und Studenten mit einem Remote-Netz vorhanden.

Durch die freundliche Unterstützung des JOIN-Projekts, einem Projekt des DFN zur Erprobung von IPv6, konnte das bereits in der Arbeit von Hoffmann (s. [Hoff 97]) verwendete 6bone-Präfix 3ffe:400:30::/48 reaktiviert werden. Damit ergibt sich für die Zuteilung von globalen Adressen die in Tabelle 5.2 dargestellte Aufteilung.

Innerhalb des Instituts werden also alle Endgeräte in der Regel mit einer Global- und einer Site-Local-Adresse konfiguriert. Zusätzlich erfolgt die automatische Konfiguration jedes Interfaces mit einer Link-Local-Adresse, deren Vorhandensein von [HiDe 98] vorgeschrieben wird.

Verwendung von Autokonfigurationsmechanismen

Vom Standpunkt der Wartungsarmut ist für den Einsatz im Institut sicherlich das statuslose Verfahren vorzuziehen. Es ist einfach, daher kaum fehleranfällig und sehr robust. Es müssen keine weiteren Informationen als die gewünschten Präfixe konfiguriert werden. Die Verwaltung eines oder mehrerer Adresspools entfällt bei dieser Vorgehensweise.

Im Gegensatz dazu bringt die Verwendung von DHCP allerdings Vorteile bei der automatischen Verteilung weitergehender Einstellungen wie der Adressen von DNS- oder BOOTP-Servern. Da das Institut aber alle Dienste selbst betreibt, ist es möglich, den dazugehörigen Servern festgelegte Site-Local-Adressen als zusätzliche Adresse zu geben und diese auf allen Endgeräten fest zu konfigurieren (ausgenommen Diskless Clients). Dies erfordert eine einmalige Änderung der verwendeten Installationsimages für die Endgeräte. Bei Diskless Clients ist das Bootimage zu ändern. Da Server üblicherweise nicht dynamisch konfiguriert werden, sind diese von diesem Problem auch nicht betroffen.

Erhält im Laufe der Zeit einer der Server, der Dienste wie DNS oder NTP bereitstellt, eine andere IPv6-Adresse, so wird die zum Dienst gehörende Site-Local-Adresse einfach beibehalten. Somit sind keine Änderungen auf den Endgeräten notwendig. Die Verwendung von Site-Local-Adressen verhindert außerdem zuverlässig, dass diese Dienste von außerhalb des Instituts verwendet werden können.

Es ist daher auf jedem Uplink-Router ein Router-Advertisement-Daemon (radvd) zu installieren, der die Präfixinformationen allen angeschlossenen Endgeräten bei Bedarf zur Verfügung stellen kann. Jedes Endgerät setzt bei Empfang eines Router-Advertisements gleichzeitig seine Default-Route auf die Link-Local-Adresse des Endgeräts, das die Router-Advertisements gesendet hat. Da das Institut nur über einen einzigen Uplink verfügt, genügt daher die Installation eines einzigen radvd in jedem Netzwerksegment des Instituts, wobei diese Installation auf dem Router vorzunehmen ist, auf den bei manueller Konfiguration die Default-Route des Segments zeigen würde. Dabei ist genau darauf zu achten, dass der radvd nur das Segment bedient, für das er zuständig ist, da es sonst zu Konflikten bei der Propagation des Präfixes (falls zwei radvd unterschiedliche Präfixe senden) oder des Default-Routers kommen würde. Die Installation mehrerer radvd ist nur dann zulässig, wenn einem Segment mehrere Uplinks zur Verfügung stehen. Jedes Endgerät in einem solchen Segment bekäme in diesem Fall ein Präfix und eine Route für jeden zur Verfügung stehenden Uplink zugeteilt (Multihoming).

Für den in Zukunft möglichen Fall, dass der Uplink-Router außerhalb der Kontrolle des Instituts steht, ist entweder eine entsprechende Vereinbarung mit dem Betreiber dieses Routers über den Betrieb eines radvd abzuschließen oder auf die Verwendung von DHCP auszuweichen.

5.1.3 Multicast-Adressierung

Für die bandbreitenoptimale Nutzung von Streaming-Anwendungen ist Multicasting die optimale Lösung. Obwohl die Entwicklung stabiler Multicasting-Fähigkeiten zu den Zielen bei der Spezifikation von IPv6 gehörte, ist der derzeitige Stand der Implementierungen aber sehr vorläufig. Das USAGI-Projekt arbeitet derzeit daran, Linux als Multicast-Client einzusetzen. Allerdings fehlt derzeit noch jeder Support für den Einsatz als Multicast-Router.

Am weitesten fortgeschritten scheint derzeit das KAME-Projekt zu sein, deren IPv6-Implementierung für BSD sowohl Multicast-Clients als auch Multicast-Routing unterstützt.

Ein Einsatz im Produktionsbetrieb scheint daher derzeit verfrüht.

5.2 Mobility

Die dargestellen Anforderungen im Bereich Mobilität zerfallen in zwei grundlegende Szenarien: Im ersten Fall ist der Benutzer mobil und möchte von verschiedenen Endgeräten, die sich innerhalb und außerhalb des Instituts befinden, auf Dienste zugreifen, die sich ebenfalls innerhalb oder außerhalb des Institut befinden. Der zweite Fall beschäftigt sich mit der Anbindung mobiler Dienstanbieter, also Endgeräten wie Notebooks, auf die unabhängig von ihrem Standort aus zugegriffen werden soll.

5.2.1 Anbindung mobiler Nutzer an das Institutsnetz

Befinden sich sowohl Nutzer als als Anbieter eines Dienstes innerhalb des Institutsnetzes, so kann ohne Probleme eine Kommunikationsverbindung zwischen beiden Teilnehmern hergestellt werden. Befindet sich einer der Teilnehmer aber außerhalb des Institutsnetzes, so verhindern die Mechanismen zum Zugriffsschutz (Firewall) in den meisten Fällen die Aufnahme einer Kommunikationsbeziehung, es sei denn, es handelt sich um einen SSH-Zugang (s. Abschnitt 1.3).

Um umfassende Konnektivität zu ermöglichen, bietet es sich an, die Verbindung zwischen den Kommunikationsteilnehmern mit Hilfe konfigurierter Tunnel (sit-Tunnel, s. Abschnitt 3.1.7) herzustellen, da damit der gesamte Netzwerkverkehr transparent zwischen dem Endgerät im Institutsnetz sowie dem im Netz des Kommunikationspartners hergestellt werden kann. Zugleich wird dadurch sichergestellt, dass das Endgerät, das sich im Remote-Netz befindet, immer die gleiche IPv6-Adresse erhält, so dass auch die Aufnahme von Verbindungen aus dem Institutsnetz zum Remote-Endgerät ermöglicht wird.

Problematisch ist dabei, dass zum Aufsetzen des sit-Tunnels auf beiden Endpunkten die IPv4-Adresse des Partners bekannt sein muss. Die Adresse des Servers kann als bekannt vorausgesetzt werden. Um die Adresse des Nutzers an den Server zu übermitteln, soll ein einfaches Client-Server-Protokoll genutzt werden. Dabei wird per TCP der Benutzername an den Server geschickt, der daraufhin die Nummer dieses Benutzers an den Client zurückliefert. Zugleich erhält der Server von seinem TCP-Stack die IPv4-Adresse des Clients. Aus der gelieferten Benutzernummer kann der Client die zugeteilte IPv6-Adresse ermitteln (s. Skripte im Anhang). Da nun sowohl Client als auch Server die dazu nötigen Informationen haben, können beide die Konfiguration des sit-Tunnels durchführen. Da zu diesem Zeitpunkt der Nutzer aber nicht authentifiziert und die Datenübertragung in diesem Tunnel nicht geschützt ist, ist mit Hilfe eines Firewalls sicherzustellen, dass ein Nutzer, der in diesem ersten Schritt nur einen sit-Tunnel hergestellt hat, keinen Zugriff auf das Institutsnetz erhält.

Beide Probleme, sowohl die Authentifizierung des Nutzers, als auch die Verschlüsselung des Tunnels, können mit Hilfe von IPSec gelöst werden. Erst wenn der Nutzer eine IPSec-Verbindung aufgebaut hat, wird ihm der Zugriff auf das Institutsnetz gestattet. Das dazu nötige Sicherheitskonzept wird weiter unten, in Abschnitt 5.4, vorgestellt.

5.2.2 Anbindung mobiler Dienstanbieter

Mit Mobile IPv6 wurde in Abschnitt 3.2 eine umfassende Lösung zur Herstellung einer vollständigen Konnektivität von mobilen Endgeräten vorgestellt. Durch die Verwendung von konventionellen Tunneln lässt sich zwar eine funktional gleichwertige Anbindung mobiler Knoten erreichen, die jedoch durch Performanceverluste erkauft werden muss, da alle Pakete zwischen mobilem Endgerät und seinem Kommunikationspartner über den Home-Link geroutet werden müssen (Dreiecksrouting). Mobile IPv6 stellt hier eine Lösung dar, die nicht nur die Transparenz der Anbindung sicherstellt, sondern mit Hilfe des Verfahrens zur Routenoptimierung auch das Problem der Performance löst. Gleichzeitig wurde, wie bei allen neueren Arbeiten der IETF, darauf geachtet, alle denkbaren Angriffszenarien zu diskutieren und Lösungen anzubieten.

Voraussetzung für den Einsatz ist jedoch eine funktionieren IPv6-Infrastruktur am Foreign-Link. Ist diese nicht vorhanden, so kann zwar mit Hilfe von Tunneln eine Verbindung zum Home-Link hergestellt werden, jedoch bringt eine Routenoptimierung hier keinerlei Performanceverbesserung mit sich. Auch für die Anbindung von mobilen Endgeräten innerhalb des Instituts genügen konventionelle Tunnel, da eine Routenoptimierung hier ihre Stärken ebenfalls nicht ausspielen kann. Das einzige relevante Szenario besteht darin, die Routenoptimierung für Endgeräte anzuwenden, deren Home-Link das private Netz des Nutzers ist, wobei dieses Endgerät aber zeitweise im Institutsnetz betrieben werden soll, allerdings unter Beibehaltung einer IPv6-Adresse aus dem privaten Netz des Nutzers. In diesem Fall kann Routenoptimierung genutzt werden, um den Umweg von Paketen über den Home-Link zu vermeiden.

5.3 Managementmechanismen

5.3.1 Tools zur Konfiguration und Wartung

Beim Aufspüren von Fehlern sind schon bei IPv4 einige Tools sehr hilfreich, die genutzt werden können, um auf verschiedenen Ebenen einfache Tests der Konnektivität durchzuführen. Einige dieser Tools wurden vom USAGI-Projekt für die Verwendung mit IPv6 angepasst, einige andere wurden von deren Entwicklern IPv6-fähig gemacht.

Folgende Tools, die zur Fehlerbehebung genutzt werden können, werden mit USAGI-ausgeliefert:

- ping6: Dieses Programm entspricht dem bekannten ping für IPv4. Es kann genutzt werden, um die Konnektivität zwischen IPv6-Hosts zu prüfen. Im bestimmten Fällen muss bei ping6 das Interface angegeben werden, über das die Ping-Pakete versendet werden sollen. Bsp: `ping6 -I eth0 fe80::2c0:f0ff:fe3b:22f9`.
- telnet: USAGI liefert eine angepasste Version den Telnet-Clients für IPv6. Damit kann TCP-Konnektivität getestet werden.
- traceroute6: Dieses Programm entspricht dem bekannten Traceroute von IPv4. Damit kann man den Weg eines Pakets zu einem entfernten Rechner verfolgen und so Probleme beim Routing erkennen.
- tracepath6: Ähnlich traceroute6 wird ein Paket auf seinem Weg zum Ziel verfolgt; außerdem liefert tracepath6 die maximale MTU, die auf dem Pfad zum Ziel verwendet werden kann.
- ip: Sowohl das mit USAGI gelieferte ip als auch das in den meisten Linux-Distributionen enthaltene ip wird verwendet, um Netzwerkparameter zu setzen und zu lesen. ip unterstützt dabei IP-Adressierung, IP-Routing, IP-Tunneling, Neighbour-Discovery (ARP) u.a.
- radvdump: Dieses Tool wartet auf Router-Advertisements und gibt sie in einer leserlichen Form aus.
- netstat: Dieses Tool entspricht dem bekannten netstat-Tool von IPv4. Es listet Netzwerkverbindungen, Routingtabellen u. a. auf.
- pfkey: Dieses Programm gibt u. a. die existierenden Security Associations sowie die Security Policy Database aus. Damit ist es möglich, Probleme bei IPSec zu debuggen.
- whack: Mit dem Kommando `whack --status` kann unter USAGI-Linux der Zustand des Key Exchange Daemons (pluto) überprüft werden.

Von den Entwicklern selbst für die Verwendung mit IPv6 angepasst wurde das bekannte tcpdump, mit dem der Netzwerkverkehr auf einem Interface sichtbar gemacht werden kann.

5.3.2 Logfiles

Abhängig von der Konfiguration des Kernels liefert das IPv6-Subsystem Statusmeldungen, die mit Hilfe des Syslog-Facility kernel in Logfiles geschrieben werden. Bei den meisten Distributionen werden solche Meldungen in die Datei `/var/log/messages` oder `/var/log/kernel` geschrieben. Diese Meldungen können verwendet werden, um Fehlern nachzugehen und deren Ursachen zu ermitteln.

5.3.3 Namensauflösung

Um eine korrekte Namensauflösung zu gewährleisten, die auch AAAA-Adressen berücksichtigt, ist der clientseitige Resolver so zu konfigurieren, dass er neben A-Anfragen auch AAAA-Anfragen durchführt.

Damit AAAA-Anfragen beantwortet werden können, müssen auf dem Nameserver des Instituts AAAA-Records für alle Endgeräte eingetragen werden. Dazu gibt es zwei unterschiedliche Ansätze: Zum einen können die AAAA-Records in den selben Zonen wie die A-Records definiert werden, wobei dann in der selben Zone A- und AAAA-Records parallel existieren. Andererseits besteht die Möglichkeit, für die IPv6-Adressen eigene Zonen einzurichten.

Um eine Trennung der unterschiedlichen Adressräume zu gewährleisten, ist es sinnvoller, letzteren Ansatz zu wählen. Dies wird realisiert, indem man für die globalen Adressen neben den bereits bestehenden Zonen informatik.uni-muenchen.de, cip.informatik.uni-muenchen.de usw. neue Zonen ipv6.informatik.uni-muenchen.de, ipv6.cip.informatik.uni-muenchen.de usw. einrichtet. Dies ermöglicht es, explizit die IPv6-Adressen eines Endgeräts zu benutzen, ohne dem Resolver das Ausweichen auf IPv4-Adressen zu gestatten. Solange zur Kommunikation hauptsächlich IPv4 benutzt wird, bietet diese Gestaltung die Möglichkeit, IPv4- und IPv6-Konnektivität strikt zu trennen, was den Test von IPv6 erleichtern wird.

Analog dazu müssen Zonen für die verwendeten Site-Local-Adressen eingerichtet werden. In Anlehnung an die Vorgehensweise bei den globalen Adressen sind hier die Zonen site6.informatik.uni-muenchen.de, site6.cip.informatik.uni-muenchen.de usw. zu verwenden.

Für Server mit bekannten Namen wie www.informatik.uni-muenchen.de soll von der eben genannten Regel abgewichen werden. Damit sie unter dem gleichen Namen auch per IPv6 erreichbar sind, soll für diese Namen neben dem bereits existierenden A-Record auch ein AAAA-Record eingerichtet werden.

Sobald die IPv6-Infrastruktur des Instituts zur primär benutzen IP-Plattform wird, kann die Nutzung der parallelen Domains aufgegeben werden, indem die dort enthaltenen AAAA-Records in die regulären Zonen übernommen werden.

Um eine korrekte umgekehrte Namensauflösung zu ermöglichen, muss darüberhinaus eine Subdomain von IP6.INT eingerichtet werden, die die PTR-Records für alle Endgeräte enthält.

5.4 Sicherheitskonzept

In jeder Anwendung, die mit Verschlüsselung oder digitalen Signaturen arbeitet, ist die Erzeugung von Schlüsseln nach einem der oben aufgeführten oder einem ähnlichen Verfahren zwingend notwendig. Daher wird auch die im Rahmen dieser Arbeit zu erstellende PKI eine Komponente zur Erzeugung von Schlüsselpaaren zur Authentifizierung der Nutzer benötigen.

Eine offene Frage ist jedoch, ob diese Schlüssel für alle Benutzer zentral oder dezentral auf den Maschinen der Benutzer erzeugt werden sollen. Zur Überprüfung der Identität eines Nutzers beim Aufbau eines Tunnels ist auf Seiten des Tunnelausgangs (also im Institut) nur der öffentliche Teil des Schlüssels notwendig. Dies würde für die Erzeugung des Schlüsselpaares am dem Rechner des Nutzers sprechen. Allerdings fehlen derzeit noch einfach zu bedienende Anwendungen, die es einem Nutzer ohne umfangreiche Vorkenntnisse erlauben, ein solches Schlüsselpaar problemlos zu erzeugen. Daher ist eine zentrale Schlüsselerzeugung vorzuziehen. Die Forderung nach Nicht-Abstreitbarkeit wird dadurch erfüllt, indem die Software zur Schlüsselerzeugung den privaten Schlüssel eines Nutzer zwar an den Nutzer weitergibt, ihn aber nicht permanent speichert. So ist sichergestellt, dass nur einer, der Nutzer selbst, den privaten Schlüssel besitzt.

In klassischen PKIs muss mit Hilfe des so erzeugten Schlüsselpaares ein Zertifikat erstellt werden. Im konkreten Szenario kann dieser Schritt entfallen: Die Identität jedes Benutzers im Institut, insbesondere am CIP-Pool, wird bereits bei der Ausgabe der Zugangskennung zu den Rechner mit Hilfe des Personalausweises oder Passes überprüft. Authentifiziert man den Benutzer bei der Schlüsselerzeugung mit Hilfe seines Rechnerpassworts, so kann man sicher sein, dass die Zuordnung zwischen dem erzeugten Schlüsselpaar und der Nutzeridentität korrekt ist. Daher genügt es, den erzeugten öffentlichen Schlüssel zusammen mit dem Unix-Loginnamen des Nutzers im Keystore zu speichern, um so bei einem Login per IPSec die Identität des Nutzers zweifelsfrei feststellen zu können.

Die Sicherheit der Übertragung des Schlüsselpaares an den Nutzers kann durch gesicherte Datenübertragung hergestellt werden; hier bietet sich als Protokoll HTTP über SSL an.

Für die Erzeugung eines geeigneten Schlüsselpaares ist daher eine Webanwendung zur Verfügung zu stellen, bei der Nutzer sich mit ihrem Unix-Passwort authentifizieren und die berechtigten Nutzern ein Schlüsselpaar zur Verfügung stellt. Gleichzeitig ist der öffentliche Schlüssel im Keystore abzulegen.

Ein Nutzer wird damit zum Aufbau einer IPSec-Verbindung autorisiert, wenn er den Key Exchange nach [HaCa 98] (IKE) mit Hilfe eines privaten Schlüssels durchführt, dessen öffentliches Pendant im Keystore gespeichert ist. IKE führt dazu eine Authentifizierung der beiden Kommunikationspartner durch und generiert einen Schlüssel zur Anwendung symmetrischer Verschlüsselung (vgl. 4.1.4). Grundlage für den Key Exchange nach IKE ist ein RSA-Schlüsselpaar.

Als lokaler Keystore für die öffentlichen Schlüssel der Nutzer soll die Datenbank verwendet werden, die im CIP-Pool bereits existiert und die Benutzerdaten enthält. Dadurch entfällt die Notwendigkeit für die Nutzer, einen dedizierten Verzeichnisdienst zu benutzen.

Vorkehrungen für den Verlust privater Schlüssel müssen nicht getroffen werden, da in solchen Fällen problemlos ein neues Schlüsselpaar/Zertifikat ausgestellt werden kann. Anders als bei der Verschlüsselung von Dokumenten gibt es bei der vorgesehenen Verwendung keine Situation, in der ein privater Schlüssel benötigt werden könnte, der verloren gegangen ist.

Ebenso kann ein Schlüssel für ungültig erklärt werden, indem er aus dem Keystore entfernt wird. Der dazugehörige Nutzer ist damit nicht mehr in der Lage, einen Key Exchange mit IKE durchzuführen.

Sicherung des sit-Tunnels und Routing eines /64-Netzes

Nachdem nun die sicherheitstechnischen Konzepte erläutert wurden, kann der Zugang für mobile Nutzer, dessen Aufbau in Abschnitt 5.2.1 begonnen wurde, nun vollendet werden.

Die Verschlüsselung des bereits konfigurierten sit-Tunnels erfolgt nach Verbindungsaufbau durch symmetrische Verfahren (s. 4.1.1) mit AH-ESP. Die Voraussetzung dafür ist der im Verlauf des Key Exchange erzeugte Schlüssel. Die für dieses Konzept notwendigen Mechanismen, also der Key Exchange mit IKE und die Verschlüsselung der Verbindung mit symmetrischen Algorithmen (ESP, s. 4.3.2), sind Bestandteil der IPSec-Implementierung des USAGI-Projekts. Um über diesen sit-Tunnel das /64-Netz des Nutzers routen zu können, wobei nicht nur die Pakete, deren Sender/Empfänger die beiden Router sind, verschlüsselt werden, sondern auch die Pakete, die zwischen dem /64-Netz und dem Institutsnetz versendet werden, muss IPSec im Tunnelmodus betrieben werden. Dies erfordert die Konfiguration eines IPIP-Tunnels, der über dem sit-Tunnel aufgebaut wird. Alle Pakete aus dem /64-Netz des Nutzers zum Institut und umgekehrt müssen über diesen IPIP-Tunnel geroutet werden und werden dadurch verschlüsselt.

Die Notwendigkeit für den zweiten Tunnel ist durch die Konzeption von IPSec bedingt (s. [KeAt 98c]). Die naheliegende Vorstellung, dass alle Pakete, die den sit-Tunnel betreten, automatisch auf der einen Seite ver- und auf der anderen Seite entschlüsselt werden, trifft nicht zu.

5.5 Aufwandsabschätzung

Der Aufwand für Entwurf und Implementierung setzt sich zusammen als den Aufwänden für die einzelnen Schritte. Außerdem müssen unterschiedliche Arbeiten zu verschiedenen Zeiten erledigt werden. Es gibt

- einmalige Arbeiten zur Einrichtung des Systems
- regelmäßig wiederkehrende Arbeiten zur Wartung
- zufällig auftretende Arbeiten zur Problembehebung im Betrieb

Task	Vorauss. Aufwand in Tagen
Quellenstudium	20
Anforderungsanalyse	20
Konzept Netzwerk	5
Konzept Parallelbetrieb	10
Konzept Sicherheit	10
Implementierung Router/Tunnelingserver	10
Implementierung IFI-Clients	5
Implementierung Clients mobiler Nutzer	5
Test	15
Summe	100

Tabelle 5.3: Aufwandsabschätzung

5.5.1 Entwurf und Implementierung

Der Aufwand für diesen Schritt setzt sich zusammen aus dem Quellenstudium, um sich in die Grundlagen der verwendeten Technologien einzuarbeiten. Sodann sind die Anforderungen an die Lösung zu definieren, wie es in diesem Kapitel geschehen ist. Anschließend sind basierend auf diesen Anforderungen Konzepte zu entwerfen, wie das einzurichtende Netzwerk aussehen soll, wie ein Parallelbetrieb hergestellt und die Sicherheitsanforderungen umgesetzt werden können. Stehen diese Konzepte fest, kann mit der Implementierung eines IPv6-Routers begonnen werden sowie eine Clientinstallation vorgenommen werden. Die Implementierung endet mit Tests der Leistungsfähigkeit von IPv6 im Vergleich zu IPv4. Eine Zusammenfassung der Aufwände findet sich in Tabelle 5.3.

5.5.2 Wartung

Ist die gefundene Lösung in Betrieb, so fallen regelmäßige Arbeiten an, deren Automatisierungsgrad den dafür nötigen Aufwand bestimmt.

Darunter fallen

- Einrichten von neuen Benutzern und deren Ausstattung mit Zugangsberechtigungen für den Tunnelingserver.
- Prüfen des Basisbetriebssystems von Servern und Clients auf verfügbare Updates.
- Erstellen regelmäßiger Backups.
- Überwachung des Betriebszustandes, möglichst automatisch.
- Prüfung der Logfiles der Server auf Unregelmäßigkeiten, soweit sie nicht durch das automatische Monitoring gemeldet werden. Falls dabei Probleme festgestellt werden, sind unter Umständen weitere Maßnahmen zur Problembehebung nötig.

5.5.3 Problembehebung

Probleme im Betrieb von Rechnersystemen entstehen oft durch Hardwareprobleme. Defekte Teile sind dann auszutauschen. Durch unbeabsichtigtes Ausschalten oder aufgrund eines Stromausfalls kann das Dateisystem der Rechner beschädigt werden, ein Problem, das in schweren Fällen nur durch Neuinstallation vom Backup behoben werden kann. Die Aufwände für solche nicht vorhersehbar auftretenden Probleme sind nicht abzuschätzen. Da die für diese Arbeit verwendete Ausstattung von Hard- und Software aber zum Standardinventar des Instituts gehört, dürfte die Problembehebung keine außergewöhnlichen Belastungen verursachen. Neu und bisher nicht im

Einsatz im Instituts ist lediglich die im Rahmen dieser Arbeit eingesetzte Spezialsoftware für IPv6, deren Stabilität und Betriebssicherheit unbekannt ist.

Außer rein technischen Problemen ist weiterhin der Aufwand zur Benutzerbetreuung zu beachten. Gerade zu Beginn der Testphase ist hier mit einem erhöhten Beratungsaufwand zu rechnen, der durch die Neuheit der verwendeten Technologien bedingt ist. Um den daraus resultierenden Aufwand zu minimieren, sollten während einer Einführungsphase nur erfahrene Nutzer Zugang zur IPv6-Infrastruktur erhalten. Sollte sich während dieser Einführungsphase die Produktionsreife der Installation bestätigen, kann diese Beschränkung aufgehoben werden.

Kapitel 6

Implementierung

Um die neuen Features von IPv6 zu demonstrieren, soll in diesem Kapitel nach einer kurzen Marktübersicht eine Implementierung des im letzten Kapitel vorgestellten Konzepts erfolgen. Ziel ist es dabei, möglichst viele der oben als nützlich erkannten Funktionalitäten zu realisieren.

6.1 Verfügbare Implementierungen von IPv6

Bei der Standardisierung des neuen Protokolls wurde vom IETF großer Wert auf die Mitwirkung der Industrie gelegt. So sind viele Dokumente unter Mitwirkung von Mitarbeitern von Firmen wie Microsoft oder Cisco entstanden, um so die Akzeptanz von IPv6 zu stärken. Die für das Institut wichtigsten Implementierungen sind im Folgenden aufgeführt, eine vollständige Marktübersicht findet sich in [Hind 02].

6.1.1 Linux

Linux enthält einen IPv6-Stack seit Kernel-Version 2.1.8. Alle Kernel der Version 2.2. und 2.4 enthalten diese Implementierung, die aber unvollständig ist und nicht mehr weiterentwickelt wird. Daher wurde mit Unterstützung mehrerer japanischer Universitäten und Firmen das USAGI-Projekt[1] ins Leben gerufen, dessen Aufgabe es ist, eine vollständige und robuste IPv6-Implementierung zu entwickeln. Dieses Projekt befindet sich derzeit noch in der Entwicklungsphase, allerdings ist die Software bereits auf einem Stand angelangt, der die Nutzung für erste Implementierungen erlaubt.

Ein jeweils aktueller Entwicklersnapshot kann von den Web-Seiten des Projekts bezogen werden. Dieser Arbeit liegt der Snapshot vom 6. Januar 2003 zu Grunde.

6.1.2 Die BSD-Familie

Für FreeBSD, NetBSD, OpenBSD sowie BSD/OS wird vom Kame-Projekt[2] eine vollständige IPv6-Implementierung entwickelt. Diese unterstützt auch die IPSec-Funktionalitäten und ist sehr weit fortgeschritten. Wie auch das USAGI-Projekt wird Kame von mehreren japanischen Firmen getragen.

6.1.3 Microsoft

Microsoft arbeitet seit 1998 an einer Implementierung von IPv6 für seine Betriebssysteme[3]. Für Windows 2000 existiert eine als Technology Preview bezeichnete Vorabversion. 2001 erschien eine weiterentwickelte Version für Windows XP. Microsoft hat angekündigt, dass der für das Jahr

[1] http://www.linux-ipv6.org
[2] http://www.kame.net
[3] http://www.microsoft.com/ipv6/

2003 angekündigte Windows .NET-Server die erste Produktionsversion eines IPv6-Stacks enthalten wird.

Leider enthalten die Microsoft-Versionen keine Unterstützung für Verschlüsselung bei Verwendung von ESP. Dies ist den strengen Exportrichtlinien der USA geschuldet. Vermutlich werden hier, wie schon bei ähnlichen Problemen in der Vergangenheit, bei Bedarf Drittanbieter mit geeigneter Zusatzsoftware einspringen.

6.1.4 Sun Microsystems

Sun hat mit dem Betriebssystem Solaris 8 einen IPv6-Protokollstack ausgeliefert[4]. Leider enthält auch diese Implementierung keine Unterstützung für IPSec. Sie ist aber als zukünftige Erweiterung angekündigt.

6.2 Einrichtung eines Parallelbetriebs am CIP-Pool

Um Linux mit einem IPv6-Protokollstack auszustatten, genügt es, bei der Konfiguration des USAGI-Kernels die Option zum Übersetzen von IPv6 auszuwählen. Diese Funktionalität kann direkt in den Kernel oder als Modul übersetzt werden.

Besondere Beachtung verdient die Option "IPv6: Privacy Extensions (RFC 3041) support". [NaDr 01] problematisiert die Sichtbarkeit der MAC-Adresse eines mit radvd konfigurierten Endgeräts im Interface Identifier einer IPv6-Adresse, die unabhängig vom Standorts eines Endgeräts immer gleich ist. Um die Identifikation von Nutzern im Internet über dieses konstante Erkennungsmerkmal auszuschließen, schlägt [NaDr 01] einen Mechanismus vor, um Interfaces mit zusätzlichen Adressen zu versehen, deren Interface Identifier zufällig ist. Ist dieses Verhalten erwünscht, so ist diese Option zu aktivieren. Im Rahmen des Betriebs des Instituts muss allerdings eine IPv6-Adresse immer einem Endgerät zugeordnet werden können, um einerseits das Debugging zu erleichtern, andererseits gegen Missbrauch vorgehen zu können. Daher ist diese Option in diesem Fall zu deaktivieren.

Neben dem Kernel enthält das USAGI-Paket mehrere Benutzeranwendungen, wie einen Inetd-Server oder die benötigten Tools zur Konfiguration einzelner Funktionalitäten.

Wird nur einfache IPv6-Konnektivität benötigt, so genügt auch ein Standardkernel. Daher können alle CIP-Workstations und -Server ohne Austausch des Kernels für einen Parallelbetrieb eingerichtet werden. Es genügt, das im Standard-Kernel des CIP enthaltene IPv6-Modul zu laden:

```
modprobe ipv6
```

Sobald das Modul geladen ist, werden automatisch alle Ethernet-Interfaces mit einer Adresse vom Typ Link-Local versehen.

Durch diese Vorgehensweise lassen sich die Endgeräte im CIP problemlos für die Verwendung von IPv6 einrichten, ohne Stabilitätsprobleme aufgrund der Verwendung eines Entwicklerkernels befürchten zu müssen. Der USAGI-Kernel ist nur dann nötig, wenn erweiterte Funktionalitäten wie Verschlüsselung verwendet werden sollen. Seine Verwendung innerhalb des CIP kann sich daher auf den im Rahmen dieser Arbeit einzurichtenden Router für Remote-Nutzer sowie die per IPSec anzubindenden Router in den Remote-Netzen beschränken.

6.3 Implementierung eines Routers für IPv6

Weil die Funktionalitäten, die insb. für die Anbindung von Remote-Nutzern benötigt werden, über die Fähigkeiten des im Standardkernel enthaltenen IPV6-Protokollstacks hinausgehen, müssen auf dem Router der USAGI-Kernel sowie die mitgelieferten Anwendungen installiert werden.

Hier die Vorgehensweise zum Übersetzen des USAGI-Kernels im einzelnen, durchgeführt mit dem am 6. Januar 2003 erschienenen Snapshot.

[4] http://wwws.sun.com/software/solaris/ipv6/

```
#!/bin/sh

cd /usr/src
wget ftp://ftp.linux-ipv6.org/pub/usagi/snap/kit/usagi-linux24-s20030106.tar.bz2
tar xfj usagi-linux24-s20030106.tar.bz2

# kernel
cd /usr/src/usagi
make prepare TARGET=linux24
cd kernel/linux24
make mrproper
make menuconfig
make dep clean bzImage modules modules_install

# tools
cd /usr/src/usagi/usagi
./configure
make
make install
```

Der neu übersetzte Kernel steht dann als Datei `/usr/src/usagi/kernel/linux24/arch/i386/boot/bzImage` zur Verfügung und kann installiert werden.

Die mitgelieferten Tools finden sich nach der Installation in den Verzeichnissen `/usr/local/v6/{bin|sbin}`. Dort wurden auch die Manual-Pages im Unterverzeichnis `man` installiert.

Nach dem Start des Netzwerks und dem evtl. Laden des IPv6-Modules werden automatisch alle Netzwerkinterfaces mit Link-Local-Adressen ausgestattet. Von diesem Zeitpunkt an ist die Kommunikation zwischen Endgeräten am gleichen Link möglich.

Um die Endgeräte auch mit Site-Local- oder globalen Adressen auszustatten, ist ein Router mit konfiguriertem Router-Advertisement-Daemon radvd notwendig. Die Konfiguration dieses Servers findet in der Datei `/usr/local/v6/etc/radvd.conf` statt. Eine minimale Konfiguration, um das Netzwerk des CIP-Pools auf die vorgeschlagenen Site-Local-Adressen zu konfigurieren, hat folgende Gestalt:

```
interface eth0
{
        AdvSendAdvert on;

        # CIP-6bone
        prefix 3ffe:400:30:1::/64
        {
        };

        # CIP-Site-local
        prefix fec0:0:0:1::/64
        {
        };

};
```

Bevor der Router-Advertisement-Daemon gestartet werden kann, muss auf dem Router noch das IPv6-Forwarding aktiviert werden. Anschließend kann der Dienst selbst gestartet werden:

```
#!/bin/sh
```

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
/usr/local/v6/sbin/radvd
```

Alle Endgeräte, die an den Link angeschlossen sind, für die der radvd konfiguriert wurde, erhalten nun im Abstand von 200-600 Sekunden (Standardeinstellung) Router Advertisements, die sie nutzen, um sich selbst mit den propagierten Präfixen auszustatten. Zugleich setzen sie eine Default-Route auf die Link-Local-Adresse des Routers. Damit ist die grundlegende Netzwerkkonfiguration aller Workstations abgeschlossen.

Da der Router selbst keine Router Advertisements beachtet, muss er manuell auf eine globale sowie die Site-Local-Adresse konfiguriert werden. Eine Route auf dieses Netz wird bei der Konfiguration des Interfaces automatisch eingerichtet. Dies geschieht mit:

```
#!/bin/sh

ip addr add 3ffe:400:30:1::1/64 dev eth0
ip addr add fec0::2:0:0:0:1/64 dev eth0
```

Ein Skript zum automatisierten Start der grundlegenden Netzwerkkonfiguration und des radvd beim Bootvorgang ist im Anhang zu finden.

Uplink-Routing

Das Uplink-Routing erfolgt über den 6bone-Router des JOIN-Projekts in Münster. Zu diesem Router muss ein Konfigurierter Tunnel (s. 3.1.7) aufgebaut werden. Dies wird von folgenden Befehlen bewerkstelligt:

```
#!/bin/sh

TUNNEL=sit1
V4_REMOTEADDR=128.176.191.82
V6_REMOTENET=2000::/3

ip tunnel add $TUNNEL mode sit local any remote $V4_REMOTEADDR ttl 64
ip link   set $TUNNEL up
ip addr add 3ffe:401:1::30:2/112 dev $TUNNEL

ip route  add $V6_REMOTENET dev $TUNNEL
```

Das Präfix 2000::/3 bezeichnet dabei alle globalen Unicast-Adressen (s. Tabelle 3.1) und entspricht damit einer Default-Route. Die Adresse 3ffe:401:1::30:2/112 ist Teil eines bei JOIN-Tunneln üblichen Transfernetzes.

Für die Gegenrichtung ist auf dem JOIN-Router eine analoge Konfiguration durchzuführen, wobei der Eintrag in die Routing-Tabelle sich auf die CIP-Netze beschränkt.

Netfilter-Konfiguration

Mit Hilfe der Linux-Netfilter-Funktionalitäten[5] wird schon jetzt im Rahmen des IPv4-Betriebes der Zugriff von nichtauthorisierten Endgeräten auf CIP-Geräte bereits auf Netzwerkebene verhindert. Diese Netfilter-Konfiguration ist so konzipiert, dass alle Endgeräte aus dem MWN Zugriff auf die vom CIP-Gerät bereitgestellten Dienste haben und sie nach Authentifizierung und Autorisierung auf Anwendungsebene auch nutzen können. Realisiert wird dieses Verhalten, indem alle Netzwerkzugriffe aus dem MWN erlaubt werden. Für die Zugriffe von außerhalb gelten folgende Regeln:

[5] http://www.netfilter.org

◇ UDP-Traffic wird abgelehnt.

◇ TCP-Traffic wird nur dann erlaubt, wenn das SYN-Flag nicht gesetzt ist.

Die Regel für TCP-Traffic besagt, dass alle TCP-Verbindungen erlaubt sind, die vom CIP-Gerät selbst initiiert wurden, Verbindungswünsche, die von außen an das CIP-Gerät herangetragen werden, jedoch abgelehnt werden.

Um beim Betrieb von IPv6 die gleiche Sicherheit herzustellen, soll eine äquivalente Konfiguration für das neue Protokoll implementiert werden. Linux bietet dazu sowohl im Standardkernel als auch in der USAGI-Implementierung Schnittstellen zur Steuerung der Netfilterregeln an. Diese werden mit Hilfe des Userland-Tools `ip6tables` implementiert. Voraussetzung für die Nutzung ist die Aktivierung der Netfilter-Kernelerweiterungen.

Es soll also jeder IPv6-Traffic, der aus dem dem Institut zugewiesenen Adressraum kommt, gestattet werden, Traffic von außerhalb nur dann, wenn das SYN-Flag nicht gesetzt ist oder auf dem entsprechenden Port ein öffentlich zugänglicher Dienst betrieben wird. Dies wird mit folgenden Regeln erreicht:

```
#!/bin/sh

# Alle alten Regeln löschen
ip6tables -F

# Zuerst die Default-Policies
ip6tables -P INPUT    ACCEPT
ip6tables -P FORWARD  ACCEPT
ip6tables -P OUTPUT   ACCEPT

# Erlaube alles vom IFI-Netz
ip6tables -A INPUT -j ACCEPT -s 4ffe:400:30::/48

# Verbiete alle SYN-Pakete
ip6tables -A INPUT -p tcp --syn -j REJECT

# Verbiete UDP
ip6tables -A INPUT -p udp -j REJECT
```

Es sei darauf hingewiesen, dass das Target REJECT nur im USAGI-Kernel zur Verfügung steht und zur Steuerung die aktuelle ip6tables-Implementierung (enthalten in iptables-1.2.7a) benötigt wird. Bei Verwendung älterer Software muss statt REJECT das Target DROP verwendet werden.

DNS-Server

Als DNS-Server im Rahmen dieser Arbeit wurde Bind9[6] verwendet. Er unterstützt sowohl AAAA-Records als auch die Nibble-Darstellung für die Reverse-Auflösung von IPv6-Adressen.

Die Konfiguration von Zonen erfolgt ebenso wie für IPv4 und muss daher hier nicht weiter diskutiert werden.

DNS-Client

Die Festlegung, wie ein Linux-Rechner eine Namensauflösung durchführt, wird in der Datei `/etc/nsswitch.conf` festgelegt. Um neben A-Records auch AAAA-Records abzufragen, muss in dieser Datei die Regel für die Auflösung von Hostnamen geändert und um den Eintrag "dns6" erweitert werden:

```
hosts:          files dns6 dns
```

[6] http://www.isc.org/products/BIND/

Dadurch wird bei der Auflösung von Hostnamen erst die Datei `/etc/hosts` befragt, anschließend wird der Nameserver nach einem AAAA- und einen A-Record befragt. Die Verarbeitung bricht ab, sobald eine Adresse gefunden wurde.

IPv6-Fähigkeiten ausgewählter Anwendungen

Im Rahmen dieser Arbeit soll nicht nur die Netzwerkebene betrachtet werden, sondern auch die IPv6-Fähigkeiten der verschiedenen, im CIP eingesetzten Dienste und Anwendungen. Die folgenden Informationen über den Stand der Implementierung des neuen Protokolls stammen von den Webseiten der Software, aus dem Usenet oder wurden durch eigenen Recherchen ermittelt. Umfangreiche Informationen dazu finden sich auch bei [Bier 03]. Für das Debian-Projekt steht ein eigenes IPv6-Archiv zur Verfügung (Debian-IPv6[7]), in dem sich IPv6-fähige Anwendungen als Binärpakete finden. Desweiteren wurden im Rahmen des KAME-Projekts Patches für viele Pakete entwickelt[8], die aber häufig nicht sehr aktuell sind.

Apache Der Apache-Webserver unterstützt IPv6 seit Version 2.0.39. Die Direktiven `Listen, VirtualHost, NameVirtualHost` akzeptieren als Parameter IPv6-Adressen, die in eckigen Klammern angegeben werden müssen[9].

Postfix Postfix bietet keine native IPv6-Unterstützung. Patches für verschiedene Versionen finden sich bei KAME, es existieren Binärpakete bei Debian-IPv6.

Courier-Imap IPv6-Support wird beim Übersetzen integriert, wenn das configure-Skript erkennt, dass das Betriebssystem IPv6 unterstützt.

NFS Laut [Bier 03] ist NFS nicht IPv6-fähig. Weder KAME noch Debian-IPv6 bieten entsprechende Pakete an.

Mozilla Mozilla kann IPv6-Adressen verarbeiten, wenn sie in eckigen Klammern stehen. Bei Namensauflösungen werden auch AAAA-Records berücksichtigt.

KDE KDE unterstützt IPv6.

OpenSSH OpenSSH unterstützt IPv6.

Perl Im CPAN[10] sind verschiedene Erweiterungen aufgelistet, die Perl um IPv6-bezogenen Funktionen erweitern. Dazu gehören

- **Net::IP** Ein Modul zur Manipulation von IPv4/IPv6-Adressen
- **Net::IPv6Addr** Ein Modul, mit dem die Gültigkeit von IPv6-Adressen geprüft werden kann.
- **Socket6** Ein Modul, das die IPv6-bezogenen Socket-Funktionen der libc in Perl zur Verfügung stellt.

Ruby Ruby unterstützt IPv6 nativ.

XFree86 IPv6-fähige Pakete von XFree86 4.2.1 finden sich bei Debian-IPv6. Dort werden auch verschiedene weitere Pakete aus dem XFree-Umfeld angeboten, wie xfs u. a.

Cups Cups unterstützt IPv6 nativ.

MySQL Die Unterstützung von IPv6 ist geplant, aber noch nicht realisiert.

PostgreSQL Ein Patch für PostgreSQL wurde von Nigel Kukard entwickelt[11].

---

[7] http://debian.fabbione.net/
[8] ftp://ftp.kame.net/pub/kame/misc/
[9] http://httpd.apache.org/docs-2.0/new_features_2_0.html
[10] http://www.cpan.org
[11] http://www.lbsd.net/

INN Laut [Bier 03] wurden bereits IPv6-Patches in die CVS-Version von INN aufgenommen. Ihr Erscheinen ist mit Version 2.4.0 geplant.

CVS Patches für verschiedene Versionen von CVS finden sich bei KAME. Sie ergänzen den pserver von CVS um IPv6-Funktionalität.

Bind Bind unterstützt IPv6 nativ in den Version Bind8 und Bind9.

HP JetDirect Als Drucker werden im Institut überwiegend Geräte der Firma HP eingesetzt, die zur Kommunikation mit den Druckservern das HP JetDirect-Protokoll nutzen. Die Implementierung von JetDirect in den Druckern ist derzeit nicht IPv6-fähig.

6.4 Implementierung eines Tunneling-Servers

Das Problem, Remote-Nutzern einen gesicherten Zugang zum CIP-Netz zur Verfügung zu stellen, gliedert sich in zwei Bereiche: Zuerst muss dem Nutzer eine IPv6-Adresse aus dem für Remote-Nutzer vorgesehenen Adressbereich des CIP-Pools zugeordnet werden. Ist sodann eine Verbindung zwischen CIP und Remote-Nutzer hergestellt, so muss diese Verbindung gesichert werden, um den Anforderungen an die Sicherheit gerecht zu werden.

Voraussetzung für die Nutzung der IPSec-Features für IPSec ist eine korrekte Kernelkonfiguration. USAGI empfiehlt dazu die Aktivierung folgender Optionen:

```
Cryptographic options  --->
        <*> CryptoAPI support
        [*] Cipher Algorithms
        --- 128 bit blocksize
        <*>  AES (aka Rijndael) cipher
        --- 64 bit blocksize
        <*>  3DES cipher
        --- Deprecated
        <*>  NULL cipher (NO CRYPTO)
        <*>  DES cipher (DEPRECATED)
        [*] Digest Algorithms
        <*>  MD5 digest
        <*>  SHA1 digest
Networking options  --->
        <*/M>  IP: tunneling
        [*]    The IPsec protocol (EXPERIMENTAL)
        [*]     IPsec: IPsec Debug messages
        [*]     IPsec: IPsec Debug disable Default
        <*/M>   The IPv6 protocol (EXPERIMENTAL)
        ...
        [*]     IPv6: IP Security Support (EXPERIMENTAL)
        <M>     IPv6: IPv6 over IPv6 Tunneling (EXPERIMENTAL)
```

6.4.1 Aufbau des IPv6-Tunnels

Zur Herstellung der Konnektivität bietet sich das Mittel des konfigurierten Tunnels an, wie es in Abschnitt 3.1.7 dargestellt wurde. Damit ist es möglich, über eine IPv4-Infrastruktur IPv6-Konnektivität herzustellen. Problematisch dabei ist jedoch, dass zum Aufbau eines solchen Tunnels auf dem im CIP gelegenen Tunnelendpunkt die IPv4-Adresse des Nutzers bekannt sein muss. Um dieses Problem zu lösen, wurde im Rahmen dieser Arbeit ein einfacher Server in Perl realisiert, dem per TCP der Wunsch eines Remote-Nutzers zum Aufbau eines konfigurierten Tunnels mitgeteilt wird. Der Nutzer übermittelt dazu seine CIP-Kennung an den Server, der die dazugehörige IPv4-Adresse aus den Verbindungsparametern erfährt. Der Server ruft daraufhin ein kleines Shell-Skript auf, das den eigentlichen Tunnelaufbau am CIP-Ende vornimmt. Gleichzeitig teilt der Server dem Client die User-ID des Remote-Nutzers mit, die der Client zur Bildung einer IPv6-Adresse nutzt (s. Abschnitt 5.2.1), woraufhin am Client-Ende ebenfalls die Konfiguration des Tunnels vorgenommen werden kann.

6.4.2 Sicherung der Verbindung

Der Nutzer ist zu diesem Zeitpunkt allerdings nicht authentifiziert. Es muss daher verhindert werden, dass er bereits jetzt Zugriff auf Dienste im CIP-Pool erhält. Daher ist auf dem Tunneling-Server eine Netfilter-Regel zu setzen, die das Forwarding von Paketen, die aus dem Adressbereich stammen, der den Remote-Nutzern zugeordnet ist, verbieten. Dies geschieht mit folgenden Befehlen:

```
#!/bin/sh

# erzeuge neuen Chain für die Remote-Nutzer
ip6tables -N remote

# und schicken deren Traffic dahin
ip6tables -A FORWARD -s 3ffe:400:30:8000::/50 -j remote

# verbiete alles
ip6tables -A remote -j REJECT
```

Sind Benutzer erfolgreich authentifiziert, so wird in den so definierten Chain eine Regel eingetragen, die das Forwarding von Paketen dieses einen Users erlaubt. Dies geschieht mit Hilfe eines Updown-Skripts, das Pluto aufruft, sobald die Verbindung korrekt hergestellt ist.

Der zweite Schritt bei der Herstellung von Konnektivität ist daher die Authentifizierung des Nutzers. Da der Remote-Nutzers bereits mit einer IPv6-Adresse ausgestattet ist, kann hier IPSec für IPv6 verwendet werden. Idealerweise werden Tunneling-Server und Nutzer mit RSA-Keys ausgestattet, die einerseits die gegenseitige Authentifizierung mit Hilfe digitaler Signaturen erlauben, andererseits die Verwendung eines automatischen Key-Exchange nach IKE (s. 5.4) zur Verschlüsselung ermöglichen.

IPSec für IPv6 ist Bestandteil der USAGI-Software. Die USAGI-Version von IPSec basiert auf Freeswan 1.91[12]. Diese Software kann sowohl für Verbindungen im Tunnel- als auch im Transport-Mode verwendet werden (s. Abschnitt 4.3.2). Da ihm Rahmen dieser Arbeit Netzes miteinander verbunden werden sollen, ist die Verwendung des Tunnel-Mode angebracht. Die zentrale Konfigurationsdatei für IPSec ist `/usr/local/v6/etc/ipsec.conf`. Die geheimen RSA-Schlüssel der Teilnehmer werden in `/usr/local/v6/etc/ipsec.secrets` abgelegt. Zum Erzeugen eines RSA-Schlüssels der Länge 2048 Bit und dem syntaktisch korrekten Eintragen in `ipsec.secrets` kann das folgende Shellskript verwendet werden:

```
#!/bin/sh

(
echo ": RSA {"
ipsec rsasigkey 2048
echo "        }"
) > ipsec.secrets
```

[12]http://www.freeswan.org

Der Public-Key kann aus der Datei `ipsec.secrets` extrahiert werden. Er ist enthalten in der Zeile, die mit `#pubkey=` beginnt.

In der Datei `ipsec.conf` erfolgen die Einstellungen für den Aufbau der verschlüsselten Verbindung.

```
# basic configuration
config setup
        interfaces=%defaultroute
        klipsdebug=all
        plutodebug=all
        plutoload=%search
        plutostart=%search

# defaults that apply to all connection descriptions
conn %default
        keyingtries=1
        af=inet6
        type=tunnel
        authby=rsasig
        leftrsasigkey=0sAQON7fqjb69v+ygkrvD....
        left=fec1::1
        rightnexthop=fec1::1
        leftsubnet=3ffe:400:30::/49


conn leinfeld
        rightrsasigkey=0sAQN6qrXZtzPrxfqVCQ4....
        right=fec1::8106
        rightsubnet=3ffe:400:30:8106::/64
        leftnexthop=fec1::8106
```

Die ersten beiden Bereiche der Konfiguration, `config setup` und `conn %default` sind identisch für Server und Remote-Nutzer einzutragen, mit einer Ausnahme: Da auf dem Server das updown-Skript bei Auf- und Abbau der Verbindung aufgerufen werden muss, muss im Bereich `conn %default` nach der Anweisung `leftsubnet` folgende Zeile eingefügt werden:

```
leftupdown=/usr/local/v6/sbin/tunnel.updown.sh
```

Zusätzlich enthält die Datei für jede gewünschte Verbindung einen zusätzlichen `conn`-Bereich. Für den Remote-Nutzer genügt ein einziger solcher Eintrag, nämlich der für den Nutzer selbst, während auf dem Tunneling-Server für jeden berechtigten Nutzer ein eigener Bereich eingetragen werden muss.

Der Parameter left beinhaltet die IPv6-Adresse des Tunnelingservers. Der Parameter leftrsasigkey enthält den Public-Key des Tunnelingservers. Die Parameter right und rightrsasigkey enthalten die entsprechenden Daten für den Remote-Nutzer. Die Public-Keys sind aus typographischen Gründen nicht vollständig wiedergegeben, wie die Punkte andeuten. Der Parameter leftupdown verweist auf ein Shell-Skript, dass beim Eintreten bestimmter Ereignisse, wie Tunnelauf- und -abbau aufgerufen wird. Dies wird genutzt, um dem oben erwähnten Remote-Chain Regeln hinzuzufügen bzw. daraus zu löschen, wenn ein Benutzer seinen Tunnel aktiviert oder deaktiviert. Die

Name	Bedeutung	Beispiel
PLUTO_CONNECTION	Name der Connection, wie in ipsec.conf definiert	leinfeld
PLUTO_ME	Tunneladresse Server	fec1::1
PLUTO_MY_CLIENT	Netzadresse serverseitig	3ffe:400:30::/49
PLUTO_MY_CLIENT_MASK	Netzmaske serverseitig	ffff:ffff:ffff:8000::
PLUTO_MY_CLIENT_NET	Netzadresse serverseitig	3ffe:400:30::
PLUTO_NEXT_HOP	Tunneladresse Client	fec1::8106
PLUTO_PEER	Adresse Client	fec1::8106
PLUTO_PEER_CLIENT	Netzadresse clientseitig	3ffe:400:30:8106::/64
PLUTO_PEER_CLIENT_MASK	Netzmaske clientseitig	ffff:ffff:ffff:ffff::
PLUTO_PEER_CLIENT_NET	Netzadresse clientseitig	3ffe:400:30:8106::
PLUTO_VERB	Aktion, durch die das Skript aufgerufen wurde	client-v6
PLUTO_VERSION	Version des Pluto	1.1

Tabelle 6.1: Umgebungsparameter bei Aufruf des Updown-Skripts durch Pluto (Auswahl)

Parameter leftnexthop, rightnexthop, leftsubnet und rightsubnet beschreiben die Netzwerktopologie, die genutzt wird, um die korrekten Security Policies einzutragen.

Leider unterstützt der USAGI-Pluto, im Gegensatz zum Original von Freeswan, den Aufruf eines Updown-Skripts von Haus aus nicht. Allerdings ist der Original-Code nach wie vor auch im USAGI-Pluto enthalten, auch wenn er zur Übersetzungszeit mit Hilfe einer Precompiler-Anweisung deaktiviert wird. Ein Grund für dieses Vorgehen konnte, auch nach Rückfrage auf der Entwicklermailingliste des USAGI-Projekts, nicht erkannt werden. Um Updown-Skripts zu reaktivieren, müssen die entsprechende Precompiler-Anweisungen im Quellcode entfernt werden. Dies erledigt folgender Patch:

```
--- kernel.c.orig       Tue Jan 14 10:31:10 2003
+++ kernel.c    Tue Feb 11 13:51:50 2003
@@ -736,7 +736,6 @@
 static bool
 do_command(struct connection *c, const char *verb)
 {
-#ifndef _USAGI
     char cmd[1024];    /* arbitrary limit on shell command length */
     const char *verb_suffix;

@@ -907,7 +906,6 @@
        }
     }
 #endif /* KLIPS */
-#endif /* _USAGI */
     return TRUE;
 }
```

Die Parameterübergabe von Pluto an das Updown-Skript erfolgt über das Environment; die definierten Parameter sind in Tabelle 6.1 aufgeführt.

Ist die ipsec.conf eingerichtet, so muss auf dem Tunnelingserver die IP-Adresse des Tunnelausgangs konfiguriert sowie die Dienste für den automatischen Key-Exchange und das Starten des konfigurierten Tunnels aktiviert werden. Außerdem wird per ip6tables der Firewall-Chain für Remote-Nutzer (s. o.) eingerichtet. Ein vollständiges Startskript findet sich im Anhang.

```
#!/bin/sh

modprobe ipv6_tunnel

ip addr add fec1::1/128 dev eth0

# das interface braucht etwas, bis es hochkommt
sleep 2

start-stop-daemon --start --quiet \
       --pidfile /var/run/pluto.pid --exec /usr/local/v6/sbin/pluto

# fuer debugging
# pluto  --nofork --stderrlog --debug-all &> /tmp/plog &

ipsec auto --ready

/usr/local/v6/sbin/tunnelserver.pl  &> /var/log/tunnelserver.log &

# erzeuge neuen Chain für die Remote-Nutzer
ip6tables -N remote

# und schicken deren Traffic dahin
ip6tables -A FORWARD -s 3ffe:400:30:8000::/50 -j remote

# verbiete alles
ip6tables -A remote -j REJECT

iptables -I INPUT -p tcp --dport 2345 -j ACCEPT

# erlaube verbindung zum pluto (500/udp)
ip6tables -I INPUT -p udp --sport 500 --dport 500 -j ACCEPT
```

Der eigentliche Verbindungsaufbau wird von den Skripten ausgeführt, die im Anhang wiedergegeben sind. Dazu wird auf Seiten des Remote-Nutzers das Client-Skript aufgerufen, dem als Parameter die Login-Kennung des Nutzers übergeben wird:

```
tunnelclient.sh leinfeld
```

Daraufhin wird auf beiden Seiten IP6-Konnektivität mittels eines konfigurierten Tunnels hergestellt. Anschließend wird ein IPIP-Tunnel aufgebaut, der den Tunnel-Mode in IPSec ermöglicht. Sodann wird durch Pluto der Austausch der Session-Keys und das Aufsetzen der Security Policies vorgenommen. Pluto richtet dabei eine Verbindung ein, die mit AH-ESP verschlüsselt. Als Algorithmen kommen hmac-md5 für AH und 3des-cbc für ESP zum Einsatz. Es besteht keine Möglichkeit, durch Änderungen in der ipsec.conf andere Algorithmen zu verwenden. Es ist davon auszugehen, dass im weiteren Entwicklungsprozess solche Möglichkeiten geschaffen werden.

6.4.3 Test der Verbindung

Wenn eine Verbindung aufgebaut ist, kann sie getestet werden. Dies geschieht am einfachsten durch das Versenden von ICMP-Echo-Requests an ein Endgerät im CIP-Netz mit dem Tool ping6:

```
ping6 -pfeed 3ffe:400:30:1:210:5aff:fe31:25f6
```

Der Parameter -pfeed bewirkt, dass die Payload des ICMP-Echo-Pakets mit den Hexadezimalwerten feed ausgefüllt wird. Diesen Umstand kann man sich zu Nutze machen, um die Wirksamkeit der Verschlüsselung zu prüfen. Das Tool tcpdump ermöglicht es, den Netzwerkverkehr auf Link-Ebene abzuhören und den Inhalt jedes einzelnen Pakets zu prüfen. Um nicht allen Netzwerkverkehr am Interface des Servers untersuchen zu müssen, wird tcpdump mit einem Filter aufgerufen, der nur Pakete sichtbar macht, die als Absender- oder Empfängeradresse die IPv4-Adresse des Clients enthält. Durch Sichten der IPv4-Verbindung kann man darüberhinaus sicher sein, keine Pakete zu erhalten, die bereits von IPSec verarbeitet und damit entschlüsselt wurden. tcpdump wird mit folgender Kommandozeile aufgerufen:

```
tcpdump -x -n -s 1500 -i eth0 host 217.185.123.239
```

Der Parameter -x weist das Tool an, den Inhalt der Pakete in hexadezimaler Form auszugeben. `-s 1500` bewirkt, dass die ersten 1500 Byte eines Pakets dargestellt werden. -i bezeichnet das Interface, an dem tcpdump die Pakete abgreifen soll. -n unterdrückt die Auflösung von IP-Adressen in DNS-Namen.

Um zu vermeiden, dass tcpdump seine Pakete am falschen Punkt der kernelinternen Verarbeitung abgreift, muss tcpdump auf einer dritten Maschine installiert werden, die aufgrund der Netzwerktopologie in der Lage ist, den Netzwerkverkehr zwischen den beiden Kommunikationsteilnehmern zu beobachten.

Bei Senden eines Echo-Requests vom Client zu einem Endgerät im Netz des CIP-Pools liefert tcpdump folgende Ausgabe:

```
tcpdump: listening on eth0
13:21:42.580468 217.185.123.239 > 141.84.214.70: fec1::5378 > fec1::1:
ESP(spi=0xfcebb7fe,seq=0x403a40) (encap)
0x0000   4500 00c8 0000 4000 3329 8dc9 d9b9 7bef
0x0010   8d54 d646 6000 0000 0068 32ff fec1 0000
0x0020   0000 0000 0000 0000 0000 5378 fec1 0000
0x0030   0000 0000 0000 0000 0000 0001 fceb b7fe
0x0040   948f 1fa3 59ee be5e 84ec fcac 8576 4c8e
0x0050   d967 1a62 ccfc 2509 1b58 57dd 70d7 5c4f
0x0060   104e 0b34 2e57 49ea 2aa1 56a4 42c5 9445
0x0070   8683 aafd ab5d ab5a b955 d8c8 8d64 0ee7
0x0080   0f31 2760 2373 e950 1835 3ec0 e43c d423
0x0090   fc08 f41f dae4 9adc 34b1 77b4 ca01 ecf4
0x00a0   b01c 7b30 456c 43bb 1382 4979 ea3d bade
0x00b0   18ec eb64 b831 a647 625d 1259 f16b 2ca2
0x00c0   5490 6cc5 5097 19f7
13:21:42.580929 141.84.214.70 > 217.185.123.239: fec1::1 > fec1::5378:
ESP(spi=0xfcebb834,seq=0x9) (encap)
0x0000   4500 00c8 0000 4000 4029 80c9 8d54 d646
0x0010   d9b9 7bef 6000 0000 008c 32ff fec1 0000
0x0020   0000 0000 0000 0000 0000 0001 fec1 0000
0x0030   0000 0000 0000 0000 0000 5378 fceb b834
0x0040   0000 0009 d9b3 feb7 161d 59e1 a9f5 b5fc
0x0050   765b 578b 14b9 e968 8487 d4a4 ff22 d5ea
0x0060   0fb5 2386 b6c5 8f40 25f7 bdb8 7b85 c00e
```

```
0x0070   2e02 52da d5f2 0322 d390 cd0a 5487 75da
0x0080   a9ac 1968 53cf 01e6 12db 33f3 d25f c837
0x0090   1d59 38ae aead 4c86 de33 7444 e635 974c
0x00a0   a449 b1b8 b9c6 aa67 a7d5 cd00 94d7 059d
0x00b0   8679 011e 5bd5 4bb2 3039 f9e9 c458 1da0
0x00c0   5ccb 7d2e bc8d 860b
```

Eine Analyse der beiden Pakete ergibt folgendes: Das erste Paket ist der Echo-Request, der vom Remote-Client versendet wurde. Es handelt sich dabei um ein IPv4-Paket, das von 217.185.123.239 (hexadezimal d9.b9.7b.ef) an 141.84.214.70 (hexadezimal 8d.54.d6.46) versendet wurde. Darin enthalten ist ein IPv6-Paket von fec1::5378 (Remote-Client, Tunneladresse) an fec1::1 (Tunnelserver, Tunneladresse). Die folgenden Daten sind verschlüsselt.

Betrachtet man das zweite Paket, so erkennt man den zum ersten Paket passenden Echo-Reply. Die Adressdaten der beiden sichtbaren Kapselungsebenen sind identisch mit denen des ersten Pakets, nur vertauscht. Der Inhalt beider Pakets ist nicht lesbar, also korrekt verschlüsselt. Wäre keine Verschlüsselung angewendet worden, so wäre die übertragene Payload “feed” deutlich erkennbar.

6.4.4 Firewalls

Um mit Hilfe der hier vorgestellten Tools einen Tunnel aufbauen zu können, sind netzseitig folgende Voraussetzungen zu erfüllen:

- Um den sit-Tunnel aufbauen zu können, muss zwischen Client und Server eine Verbindung mit IPv4-Protokoll 41 (IPv6) hergestellt werden können. Dieses Protokoll darf nicht auf Routern oder Paketfiltern zwischen Client und Server blockiert werden.
- Auf dem sit-Tunnel wird der verschlüsselte IPIP-Tunnel aufgebaut. Daher darf auf dem sit-Tunnel das IPv6-Protokoll 50 (ESP) nicht blockiert werden.
- Ist der IPIP-Tunnel aufgebaut, so muss zwischen Client und Server eine IPv6-UDP-Verbindung auf Port 500 (isakmp) aufgebaut werden, um den Key-Exchange durchzuführen. Diese Verbindung darf nicht durch einen IPv6-Paketfilter blockiert werden.
- Ist der Tunnel aufgebaut, darf keine der gewünschten IPv6-Verbindungen durch einen IPv6-Paketfilter blockiert werden.

6.4.5 Clientseitige Fehlersuche

Ist der Ping des letzten Abschnitts gescheitert, so gibt es dafür mehrere mögliche Gründe:

1. Der Verbindung für den angeforderten User ist nicht definiert. Dieser Fehler wird vom Client-Skript gemeldet. Die Ursache ist entweder ein Tippfehler bei der Angabe des Usernamens, oder der Benutzer hat sich nicht für die Benutzung des Tunnels angemeldet und ist daher auch nicht freigeschaltet.
2. Der sit-Tunnel konnte nicht aufgebaut werden. Ursache dafür ist mit hoher Wahrscheinlichkeit ein Paketfilter, der das Protokoll 41 sperrt. Möglicherweise wird zwischen Client und Server auch NAT durchgeführt, ohne Protokoll 41 dabei zu berücksichtigen.
3. Der IPIP-Tunnel konnte nicht aufgebaut werden. Dieses Problem tritt nur dann auf, wenn der sit-Tunnel nicht korrekt gestartet werden konnte.
4. Die Security-Policies konnten nicht aufgebaut werden. Diese Problem tritt dann auf, wenn die gegenseitige Authentifizierung beim Key Exchange gescheitert ist. Möglicherweise konnte auch keine Verbindung auf UDP-Port 500 hergestellt werden.

Um ein Problem einzukreisen, bietet es sich als ersten Schritt an, zu überprüfen, ob alle Tunnelinterfaces korrekt installiert wurden. Der Befehl `ip address show` sollte daher neben dem IPv4-Interface des Clients (eth0, ppp0 o. ä.) folgende Interfaces anzeigen:

```
6: sit1@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/sit 0.0.0.0 peer 141.84.214.70
    inet6 fec1::101/128 scope site
    inet6 fe80::d9b9:7bec/64 scope link
    inet6 fe80::c0a8:1/64 scope link
7: tnl0@sit1: <POINTOPOINT,NOARP,UP> mtu 1440 qdisc noqueue
    link/[803]
    inet6 fe80::10/128 scope link
    inet6 3ffe:400:30:8106::1/128 scope global
```

Die hier angegebenen IPv6-Adressen variieren von Benutzer zu Benutzer.

Zu diesen Interfaces und Adressen gehören auch Routing-Einträge, die mit dem Befehl `ip -6 route` angezeigt werden können und etwa folgendes Aussehen haben sollten. Aufgeführt sind hier nur die tunnelbezogenen Routen, soweit es sich nicht um Routen für Link-Local-Adressen handelt:

```
2000::/3 dev tnl0  metric 1024  mtu 1440 advmss 1380
fec1::1 dev sit1  metric 1024  mtu 1480 advmss 1420
default dev tnl0  proto kernel  metric 256  mtu 1440 advmss 1380
```

Zusätzlich kann mit dem Befehl `pfkey -L` überprüft werden, ob die beiden Security Policies mit den dazugehörigen zwei Security Associations korrekt aufgesetzt wurden. Die Ausgabe dieses Befehls sollte etwa folgendermaßen aussehen:

```
SADB:
sa:d54d4e00
fec1:0000:0000:0000:0000:0000:0000:0001/0 500 fec1:0000:0000:0000:0000:0000:0000:8106/0
0 0 3 0xa6bf7989 3 3
216551dcd07fa9d649714bd688a08103637e408f
4d67336e85a145132d9c7eb20e36c29ada3ea3284cd20daf
0 0 0 0 0 0 0 0 0 0 35794 0 1 1

sa:c1b52400
fec1:0000:0000:0000:0000:0000:0000:8106/0 0 fec1:0000:0000:0000:0000:0000:0000:0001/0
500 0 3 0x40b53645 3 3
51c4b162c63c5b520cac88170d663738606a7305
a930514289b232ec30e50899a3d661a1f15d21002b6e02ef
0 0 0 0 0 0 0 0 0 0 35794 0 1 1


SPD:
spd:c1b52000
3ffe:0400:0030:8106:0000:0000:0000:0000/64 0 3ffe:0400:0030:0000:0000:0000:0000:0000/49
0 0 1 0
sa(esp):00000000 fec1:0000:0000:0000:0000:0000:0000:0001/0 3 0x40b53645

spd:c1b52a00
```

```
3ffe:0400:0030:0000:0000:0000:0000:0000/49 0 3ffe:0400:0030:8106:0000:0000:0000:0000/64
0 0 1 0
sa(esp):00000000 fec1:0000:0000:0000:0000:0000:0000:8106/0 3 0xa6bf7989
```

Die einzelnen Bestandteile des Mehrfachtunnels können auch einzeln getestet werden. Um den sit-Tunnel zu überprüfen, sendet man einen Echo-Request an die andere Seite des sit-Tunnels: `ping6 -Isit1 -pfeed fec1::1`. Um den IPIP-Tunnel zu testen, verwendet man die Adresse dessen Ausgangs: `ping6 -I tnl0 -p feed fe80::20` Da die Verbindung zur Link-Local-Adresse des Tunnels nicht verschlüsselt wird, ist ein Echo-Request an diese Adresse jedoch nicht von korrekten Security Policies abhängig.

6.4.6 Mobility-Support in IPv6

Die Konfiguration von Home Agent (HA), Correspondent Node (CN) und Mobile Node (MN) erfolgt bei der USAGI-Implementierung in der Datei `/usr/local/v6/etc/network-mip6.conf`. Diese Konfiguration wird durch das mitinstallierte Skript `/usr/local/v6/etc/init.d/mobile-ip6` ausgewertet. Dieses Skript lädt das Kernelmodul mobile_ipv6, dem als Parameter die gewünschte Funktion des Knotens (HA, CN, MN) mitgeteilt wird. Soll der Knoten als HA oder MN verwendet werden, so wird das Kernelmodul mit dem Tool mipdiag mit weiteren Parametern wie Adressen konfiguriert. Um die Mobile-IPv6-Funktionalität im Kernel zu aktivieren, muss bei der Konfiguration die Option "IPv6: Mobility Support" ausgewählt werden.

Um die Funktionalität von Mobile-IPv6 zu testen, wurde ein entferntes Endgerät mittels des oben vorgestellten Tunnels an den CIP-Pool angebunden. Das an das entfernte Endgerät geroutete /64-Netz mit der Adresse 3ffe:400:30:8106::/64 stellt den Home Link des Mobilen Knotens dar. Das entfernte Endgerät mit der Adresse 3ffe:400:30:8106::1/64 soll als HA konfiguriert werden. Der Mobile Node soll die Home Address 3ffe:400:30:8106::2/64 erhalten.

Ein einfacher Ping zwischen CIP-Router und dem entfernten Rechner bestätigt die Konnektivität. Um den entfernten Rechner als HA zu betreiben, sind folgende Einstellung in `network-mip6.conf` vorzunehmen. Der Übersichtlichkeit halber wurden die enthaltenen Kommentare entfernt.

```
FUNCTIONALITY=ha
HOMELINK=eth0
```

Zusätzlich muss auf dem HA der radvd für den mobilen Knoten konfiguriert und gestartet werden. Die Konfigurationsdate `/usr/local/v6/etc/radvd.conf` hat folgenden Inhalt:

```
interface tnl0
{
        AdvSendAdvert on;
        AdvHomeAgentFlag on;
        AdvHomeAgentInfo on;

        prefix 3ffe:400:30:8106::1/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
```

Anschließend ist der Mobile Knoten zu konfigurieren. Die Datei `network-mip6.conf` enthält folgende Zeilen:

```
FUNCTIONALITY=mn
HOMEDEV=eth0
HOMEADDRESS=3ffe:400:30:8106::2/64
HOMEAGENT=3ffe:400:30:8106::1/64
```

Um die Kommunikation zwischen dem Kernelmodul und mipdiag zu ermöglichen, ist zusätzlich auf HA und MN ein passendes Character-Device einzurichten:

```
mknod /dev/mipv6_dev c 0xf9 0
```

Anschließend kann auf HA und MN durch Aufruf des Startskripts die Mobile-IPv6-Funktionalität gestartet werden:

```
/usr/local/v6/etc/init.d/mobile-ip6 start
```

Die Konfiguration eines Correspondent Node (CN) ist trivial: Es genügt folgende Einstellung in `network-mip6.conf`:

```
FUNCTIONALITY=cn
```

Danach kann die Mobile-IPv6-Funktionalität auch auf dem CN gestartet werden.

Zum Test des Konfiguration wurde versucht, einen ICMP-Echo-Request vom CN zum MN zu senden. Leider konnte auf diese Art und Weise keine Konnektivität zur Home Address des MN hergestellt werden. Als Ursachen kommen entweder Implementierungsfehler in der Software oder Fehler in obiger Konfiguration in Frage. Möglicherweise ist der Einsatz von Mobile IPv6 erfolgversprechender, wenn die Software ausgereifter ist und ausreichend Dokumentation zur Verfügung steht.

6.5 Performance

Um die Performance verschiedener Betriebsarten zu testen, wurden mit Hilfe des Tools netperf[13] Benchmark-Tests durchgeführt. Dieses Tool baut bei Testbeginn eine Kontrollverbindung zwischen Client und Server auf, auf der Testparameter übermittelt werden. Zur Durchführung des Tests wird sodann eine gesonderte Verbindung etabliert. Die genutzten Rechner sowie die Verbindung zwischen Client und Server waren unbelastet.

Die durchgeführten Tests lassen sich in zwei Kategorien einordnen:

- TCP_STREAM: Dieser Test misst den maximalen Datendurchsatz bei TCP-Streams. Es wurde die Übertragung von TCP-Paketen mit einer Größe von 4096 und 65536 Bytes vom Client zum Server getestet.
- TCP_RR: Dieser Test führt Transaktionen nach dem Request/Response-Schema durch und misst die maximale Anzahl an Transaktionen. Es wurden Requests/Responses mit je einem 1 Byte Größe sowie Requests zu 128 Byte und Responses zu 8192 Bytes getestet, wobei alle Anfragen als TCP-Pakete versendet wurden.

6.5.1 Testdurchführung

Die einzelnen Pakete werden nacheinander übertragen. Zusätzlich messen alle Tests die CPU-Nutzung auf Sender- und Empfängerseite. Die Tests, deren Ergebnisse in den Tabellen 6.2 bis 6.5 dargestellt sind, wurden mit Hilfe des im Anhang angegebenen Skripts durchgeführt. Dieses Skript sichert die Verbindung mit Hilfe der unterschiedlichen Verfahren und Algorithmen. Dabei wird, anders als in der Implementierung des Tunnelingservers, die Technik des manuellen Keyings benutzt, das heißt, es findet kein automatischer Schlüsselaustausch mit Hilfe von Public-Key-Algorithmen statt, sondern die Schlüssel für die symmetrischen Verfahren stehen von vornherein fest und werden mittels des Tools `pfkey` in die Kerneltabellen (Security Policy Database und Security Associations) eingetragen. Bei allen Tests werden zur Authentifizierung Schlüssel der Länge 128 Bit (für die Algorithmen hmac-sha1 und hmac-md5) und zur Verschlüsselung Schlüssel der Länge 196 Bit (für die

[13] http://www.netperf.org

Algorithmen des-cbc, 3des-cbc und aes-cbc) eingesetzt. Ist die Verbindungssicherheit hergestellt, wird vom Skript der eigentliche Test gestartet und das Ergebnis ausgegeben.

Die Tests, deren Ergebnisse in den Tabellen 6.2 bis 6.5 dargestellt sind, dienten dazu, den Overhead verschiedener Verschlüsselungsverfahren zu ermitteln. In der ersten Spalte ist das Protokoll sowie die angewandte Sicherung der Verbindung aufgeführt. Als Ausgangswert für die Analysen dienen die Werte für IPv4. Die Testanordnung umfasste zwei Rechner mit einer CPU vom Typ Intel Pentium II 300 MHz, 128 MB RAM und einer 3Com 3c905B Netzwerkkarte. Die beiden Rechner waren über einen Switch an ein 100Mbit-Ethernet-Netzwerk angeschlossen. Alle Tests wurden über einen Zeitraum von 60 Sekunden durchgeführt. Die Testsoftware lässt es nicht zu, dass mehrere Clients parallel auf einen Server zugreifen.

6.5.2 Testergebnisse

Die Tests zeigen, dass bei unverschlüsselter Übertragung TCP über IPv6 einen nur wenig niedrigeren Datendurchsatz zulässt als bei IPv4. Tabelle 6.2 zeigt aber auch, dass die Aufnahme einer Verbindung mit TCP über IPv6 erheblich aufwändiger ist als mit IPv4. Während in den Tabellen 6.3 bis 6.5 die Verwendung von IPv6 eine Senkung der Transaktionsanzahl bzw. des Durchsatzes von nur 2-3% ergibt, so zeigt Tabelle 6.2 einen um 5% niedrigeren Durchsatz. Dieser Test (Request/Response mit je 1 Byte Größe) unterscheidet sich von den anderen dreien dadurch, dass kaum Nutzdaten übertragen werden. Es ist also davon auszugehen, dass die Aufnahme einer Verbindung mit TCP über IPv6 im Vergleich zu TCP über IPv4 deutlich höheren Overhead produziert als die Übertragung von Daten bei bestehenden Verbindungen.

Nicht überraschend ist, dass die Anwendung von Verschlüsselung den Datendurchsatz und die Transaktionsanzahl senken. Diese Reduzierung ist unabhängig vom Test und der Größe der Nutzdaten zu beobachten. Allerdings sind unterschiedliche Sicherungsverfahren und die dabei verwendeten Algorithmen unterschiedlich effizient. Die Anwendung von AH allein ist effizienter als die Anwendung von AH in Kombination mit ESP.

Im Vergleich der Authentifizierungsalgorithmen zeigt sich hmac-md5 in allen vier Tests deutlich effizienter als hmac-sha1. Dies gilt sowohl bei Betrachtung des Durchsatzes als auch der CPU-Last. Unter den Verschlüsselungsalgorithmen null-Verschlüsselung, des-cbc, 3des-cbc und aes-cbc ist die Anwendung der null-Verschlüsselung am effizientesten. Da dabei keine Verschlüsselung durchgeführt wird, sondern nur ein ESP-Header eingefügt wird, können die Ergebnisse dieser Tests als Vergleichsbasis bei Anwendung der anderen Algorithmen herangezogen werden. Diese Vorgehensweise hat den Vorteil, dass im Vergleich der Algorithmen der Protokolloverhead durch ESP außer Acht gelassen wird. Unter den "echten" Algorithmen des-cbc, 3des-cbc und aes-cbc zeigt des-cbc die beste Leistung bei den Request/Response-Tests. Allerdings ist dieser Algorithmus nicht praxistauglich, da er als unsicher und anfällig für Angriffe gilt. Seine Weiterentwicklung, 3des-cbc, gilt zwar als sicher, halbiert aber auch die Transaktionsanzahl. Bei den Stream-Tests halbiert sich der Durchsatz im Vergleich zu des-cbc ebenfalls, wobei hier bei allen Algorithmen die Leistung der CPU den begrenzenden Faktor darstellt. Als bester Verschlüsselungsalgorithmus zeigt sich aes-cbc, der bei vergleichbarer CPU-Last den höchsten Durchsatz bei den Stream-Tests und die zweithöchste Transaktionsanzahl bei den Request/Response-Tests liefert. Zugleich gilt aes-cbc als ebenso sicher wie 3des-cbc.

Tabelle 6.6 vergleicht den Durchsatz bei einer Verbindung zwischen einem Remote-Rechner, der via ISDN-64K-Uplink an das Internet angeschlossen ist, und dem Tunnel-Server im CIP. Der Remote-Rechner ist mit einer Pentium II-CPU (450 MHz) ausgestattet, der Tunnelserver mit einem Athlon 1800+. Allerdings zeigen die Werte für die CPU-Last, dass beide Rechner kaum belastet sind. Unabhängig von der Größe des TCP-Paketes liegt der Durchsatz bei der gesicherten IPv6-Verbindung um ca. 10% niedriger als bei ungesicherter IPv4-Übertragung. Angesichts des Overheads durch die Übertragung des größeren IPv6-Headers sowie der zusätzlichen Header, die durch die Sicherung der Übertragung notwendig sind, war dieses Ergebnis zu erwarten.

Die Tests, deren Ergebnissen in den Tabellen 6.7 bis 6.9 aufgeführt sind, wurden mit Hilfe von Rechner mit einer Athlon 1800+-CPU durchgeführt.

Verbindung	TA/Sek.	CPU lokal in %	CPU remote in %
IPv4	5012.45	52.91	52.79
IPv6	4770.34	44.44	40.85
IPv6 AH (hmac-md5)	2428.21	49.18	49.11
IPv6 AH-ESP (hmac-md5,null)	2361.86	48.58	46.79
IPv6 AH-ESP (hmac-md5,des-cbc)	2119.50	52.60	52.06
IPv6 AH-ESP (hmac-md5,3des-cbc)	1799.74	59.93	71.57
IPv6 AH-ESP (hmac-md5,aes-cbc)	2116.82	53.10	52.40
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,null)	1727.96	49.96	47.85
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,des-cbc)	1586.39	44.61	44.66
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,3des-cbc)	1399.03	51.24	58.69
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,aes-cbc)	1592.05	40.74	39.03
IPv6 AH (hmac-sha1)	1677.97	48.06	46.48
IPv6 AH-ESP (hmac-sha1,null)	1588.31	43.03	42.28
IPv6 AH-ESP (hmac-sha1,des-cbc)	1467.53	50.21	49.08
IPv6 AH-ESP (hmac-sha1,3des-cbc)	1305.17	58.10	57.27
IPv6 AH-ESP (hmac-sha1,aes-cbc)	1478.99	47.48	46.88
IPv6 AH (hmac-sha1) ESP (hmac-sha1,null)	1020.19	50.49	51.35
IPv6 AH (hmac-sha1) ESP (hmac-sha1,des-cbc)	968.67	49.01	49.16
IPv6 AH (hmac-sha1) ESP (hmac-sha1,3des-cbc)	896.42	34.69	33.26
IPv6 AH (hmac-sha1) ESP (hmac-sha1,aes-cbc)	983.87	46.64	45.48

Tabelle 6.2: Testergebnisse TCP_RR 1/1

Verbindung	TA/Sek.	CPU lokal in %	CPU remote in %
IPv4	922.52	45.06	25.37
IPv6	911.04	50.06	33.64
IPv6 AH (hmac-md5)	617.70	68.85	63.68
IPv6 AH-ESP (hmac-md5,null)	581.63	68.93	63.00
IPv6 AH-ESP (hmac-md5,des-cbc)	249.19	70.80	71.02
IPv6 AH-ESP (hmac-md5,3des-cbc)	121.49	82.15	81.22
IPv6 AH-ESP (hmac-md5,aes-cbc)	338.43	75.43	72.82
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,null)	411.28	74.47	66.42
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,des-cbc)	211.99	80.65	76.82
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,3des-cbc)	112.14	82.52	80.58
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,aes-cbc)	273.06	77.52	73.87
IPv6 AH (hmac-sha1)	377.78	73.86	72.07
IPv6 AH-ESP (hmac-sha1,null)	368.41	74.38	71.55
IPv6 AH-ESP (hmac-sha1,des-cbc)	199.52	47.80	50.48
IPv6 AH-ESP (hmac-sha1,3des-cbc)	108.36	82.65	80.90
IPv6 AH-ESP (hmac-sha1,aes-cbc)	251.14	79.48	76.05
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,null)	221.14	78.58	75.40
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,des-cbc)	146.72	80.02	78.08
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,3des-cbc)	90.75	82.48	81.23
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,aes-cbc)	172.40	78.65	78.12

Tabelle 6.3: Testergebnisse TCP_RR 128/8192

Verbindung	Durchsatz 10^6 bits/Sek.	CPU lokal in %	CPU remote in %
IPv4	94.14	39.13	62.05
IPv6	92.19	42.94	65.49
IPv6 AH (hmac-md5)	64.91	78.27	87.02
IPv6 AH-ESP (hmac-md5,null)	60.15	78.30	88.43
IPv6 AH-ESP (hmac-md5,des-cbc)	21.28	89.85	93.32
IPv6 AH-ESP (hmac-md5,3des-cbc)	9.77	93.30	94.75
IPv6 AH-ESP (hmac-md5,aes-cbc)	30.29	85.68	92.22
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,null)	39.75	80.85	90.27
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,des-cbc)	18.00	89.55	93.33
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,3des-cbc)	9.00	92.58	94.90
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,aes-cbc)	24.06	86.27	93.03
IPv6 AH (hmac-sha1)	36.22	82.83	90.93
IPv6 AH-ESP (hmac-sha1,null)	34.64	81.99	90.00
IPv6 AH-ESP (hmac-sha1,des-cbc)	16.97	89.45	93.52
IPv6 AH-ESP (hmac-sha1,3des-cbc)	8.73	89.04	97.15
IPv6 AH-ESP (hmac-sha1,aes-cbc)	21.96	87.04	92.42
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,null)	19.72	85.29	92.72
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,des-cbc)	12.36	89.43	93.79
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,3des-cbc)	7.26	94.79	97.19
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,aes-cbc)	14.83	87.51	93.55

Tabelle 6.4: Testergebnisse TCP_STREAM 4k

Verbindung	Durchsatz 10^6 bits/Sek.	CPU lokal in %	CPU remote in %
IPv4	94.09	35.54	62.02
IPv6	92.17	38.57	65.68
IPv6 AH (hmac-md5)	64.88	76.33	87.08
IPv6 AH-ESP (hmac-md5,null)	60.18	75.68	88.37
IPv6 AH-ESP (hmac-md5,des-cbc)	21.31	89.42	93.32
IPv6 AH-ESP (hmac-md5,3des-cbc)	9.78	92.99	94.55
IPv6 AH-ESP (hmac-md5,aes-cbc)	30.25	84.87	91.98
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,null)	39.70	79.43	91.05
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,des-cbc)	18.06	88.38	93.60
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,3des-cbc)	9.00	92.42	94.85
IPv6 AH (hmac-md5) AH-ESP (hmac-md5,aes-cbc)	24.04	85.48	93.05
IPv6 AH (hmac-sha1)	36.22	81.95	91.20
IPv6 AH-ESP (hmac-sha1,null)	34.64	80.67	90.32
IPv6 AH-ESP (hmac-sha1,des-cbc)	16.96	89.17	93.62
IPv6 AH-ESP (hmac-sha1,3des-cbc)	8.73	92.74	96.38
IPv6 AH-ESP (hmac-sha1,aes-cbc)	21.94	86.28	92.63
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,null)	19.75	84.82	92.62
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,des-cbc)	12.35	89.22	93.50
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,3des-cbc)	7.27	94.77	97.02
IPv6 AH (hmac-sha1) AH-ESP (hmac-sha1,aes-cbc)	14.79	87.56	93.80

Tabelle 6.5: Testergebnisse TCP_STREAM 64k

Die Tabellen 6.7 und 6.8 untersuchen den Einfluss der Rechenleistung auf den Datendurchsatz im Stream-Test und die Anzahl der Transaktionen im Request/Response-Tests bei Verwendung von TCP über IPv4 einerseits und über gesichertes IPv6 andererseits. Als Sicherungsverfahren wurde AH-ESP mit den Algorithmen hmac-md5 und 3des-cbc eingesetzt, die selbe Sicherung, die auch bei der Anbindung von Remote-Usern angewendet wird (s. Abschnitt 6.4.2). Tabelle 6.7 zeigt, dass der Durchsatz im Stream-Test unabhängig von der Größe des TCP-Pakets um den Faktor 4,5 höher ist als bei Einsatz einer deutlich langsameren CPU (vgl. Tabellen 6.4 und 6.5). Dennoch ist auch bei diesem Test die CPU der begrenzende Faktor, so dass eine weitere Steigerung des Durchsatzes zu erwarten ist, wenn man den Test mit einer schnelleren CPU durchführt. Tabelle 6.8 wiederum zeigt, dass deutliche Auswirkungen der CPU-Leistungen auf die Transaktionsanzahl beim Request/Response-Test über eine ungesicherte IPv4-Verbindung nur bei minimalen Nutzdaten erkennbar sind (vgl. Tabellen 6.2 und 6.3). Offensichtlich beschleunigt eine schnelle CPU den Aufbau einer IPv4-TCP-Verbindung. Ebenso beschleunigt wird sowohl Verbindungsaufbau als auch Datendurchsatz bei gesicherten IPv6-Verbindungen.

Tabelle 6.9 versucht, die Skalierbarkeit hinsichtlich der Größe der Security Policy Database (SPD) zu ermitteln. Mit Hilfe des im Anhang angegebenen Skripts wurden zu diesem Zweck 500,1000,1500 und 2000 Einträge in die SPD gemacht, zusätzlich zu den Einträgen, die die Testverbindung sichern. Dann wurden ein Request/Response-Test mit minimaler Nutzlast und ein Stream-Tests mit einer Paketgröße von 4 kB durchgeführt. Diese Tests wurden dann auf Rechner mit den minimal nötigen Einträgen in die SPD wiederholt. Betrachtet man die beiden Extreme, so erhöht sich die Transaktionsanzahl bei minimaler SPD um den Faktor 18, der Datendurchsatz nur um den Faktor 6. Das bedeutet, dass beim Request/Response-Test weitaus mehr Lookups in der SPD notwendig sind als beim Stream-Test. Weiterhin lässt dieses Ergebnis auf erhebliches Optimierungspotential bei der Implementierung der SPD schließen.

Folgerungen

Geht man von den genannten Werten für den Datendurchsatz und die CPU-Nutzung aus, lassen sich Aussagen über die Anzahl maximal sicher anzubindender Client-Rechner am Tunnelserver machen. Zu untersuchen sind dabei drei Gruppen von Endgeräten, die sich im Wesentlichen in der Bandbreite Ihrer IPv4-Anbindung an den Server unterscheiden. Basis für diese Schätzung ist Tabelle 6.7, die den maximal erziehlbaren Durchsatz einer verschlüsselten IPv6-Verbindung angibt. Dieser liegt bei ca. 44 MBit/Sekunde.

Anbindung über 100 MBit-Ethernet Schon mit einer einzigen Verbindung zu einem per 100 MBit angebundenem Endegerät steigt die CPU-Last auf 90%. Daher ist der maximale Durchsatz von 44 MBit wohl auch nur dann zu erreichen, wenn nur ein einziges Endgerät angebunden ist. Voraussetzung dafür ist, dass das Endgerät über eine vergleichbare CPU-Leistung verfügt wie der Tunnel-Server.

Anbindung über A-DSL A-DSL bietet Endgeräten eine Upstream-Datenrate von 768 kBit/Sekunde. Berücksichtigt man die Reduzierung des Datendurchsatzes durch die Tunnel-Anbindung von etwa 10% (s. Tabelle 6.6), so kommen davon ca. 700 kBit/Sekunde beim Server an. Basierend auf dem maximalen Durchsatz von 44 MBit/Sekunde ergibt sich damit eine Höchstzahl anzubindender Endgeräte von 63, die dann die volle Leistungsfähigkeit Ihrer Anbindung nutzen können. Berücksichtigt man die Ergebnisse von Tabelle 6.9, so dürfte der tatsächliche Maximalwert aber deutlich darunter liegen, vermutlich bei etwa 40-50 Verbindungen.

Anbindung über ISDN Bei einer Anbindung über einen Remote-Tunnel steht ISDN-Nutzern eine Datenrate von 57 kBit/Sekunde zur Verfügung (s. Tabelle 6.6). Umgerechnet auf den Maximaldurchsatz des Servers von 44 MBit/Sekunde können damit maximal 772 ISDN-Nutzer bei voller Bandbreite angebunden werden. Allerdings müssen dafür 1544 Einträge in die Security Policy Database vorgenommen werden, was die am Server verfügbare Bandbreite auf etwa 9 MBit/Sekunde redizieren würde (s. Tabelle 6.9). Dies reicht zur Anbindung von

Verbindung	Paketgröße	Durchsatz 10^3 bits/Sek.	CPU lokal in %	CPU remote in %
IPv4	4096	60.92	1.91	0.07
IPv6 AH-ESP (hmac-md5,3des-cbc)	4096	56.90	2.07	0.26
IPv4	65536	61.00	5.06	0.09
IPv6 AH-ESP (hmac-md5,3des-cbc)	65536	56.91	3.75	0.05

Tabelle 6.6: Testergebnisse TCP_STREAM verschiedener Größen über ISDN-Uplink

Verbindung	Paketgröße	Durchsatz in 10^6 bits/Sek.	CPU lokal in %	CPU remote in %
IPv4	4096	94.12	14.14	20.14
IPv6 AH-ESP (hmac-md5,3des-cbc)	4096	44.57	89.53	90.68
IPv4	65536	94.05	14.09	19.44
IPv6 AH-ESP (hmac-md5,3des-cbc)	65536	44.21	89.11	90.16

Tabelle 6.7: Testergebnisse TCP_STREAM verschiedener Größen mit 2x AMD Athlon 1800+

gerade mal 158 ISDN-Nutzern bei voller Datenrate. Der tatsächlich erreichbare Wert dürfte damit im Bereich von etwa 400 bis 500 Endgeräten liegen.

6.6 Schlüsselmanagement

Um die Benutzer des Tunnelservers zu verwalten, werden allen Benutzerdaten zentral in einer MySQL-Datenbank abgelegt. Diese Datenbank enthält eine einzige Tabelle users mit folgendem Schema:

```
CREATE TABLE users (
  id int(11) NOT NULL auto_increment,
  username varchar(20) NOT NULL default '',
  hostid int(11) NOT NULL default '0',
  gesperrt int(11) NOT NULL default '0',
  pubkey text,
  UNIQUE KEY hostid (hostid),
  UNIQUE KEY username (username),
  KEY id (id)
) TYPE=MyISAM;
```

Verbindung	Paketgröße	TA/Sek.	CPU lokal in %	CPU remote in %
IPv4	1/1	7431.70	22.71	25.04
IPv6 AH-ESP (hmac-md5,3des-cbc)	1/1	4692.47	25.76	24.81
IPv4	128/8192	928.45	17.79	12.46
IPv6 AH-ESP (hmac-md5,3des-cbc)	128/8192	481.64	68.26	68.01

Tabelle 6.8: Testergebnisse TCP_RR verschiedener Größen mit 2x AMD Athlon 1800+

Test	Größe SPD	Durchsatz		CPU lokal in %	CPU remote in %
TCP_RR 1/1	2002	263.06	TA/Sek.	47.37	49.86
TCP_STREAM 4096	2002	7.64	10^6 bits/Sek.	61.92	93.32
TCP_RR 1/1	1502	317.43	TA/Sek.	48.01	49.02
TCP_STREAM 4096	1502	9.02	10^6 bits/Sek.	65.95	92.91
TCP_RR 1/1	1002	481.51	TA/Sek.	48.23	48.50
TCP_STREAM 4096	1002	12.67	10^6 bits/Sek.	68.56	92.26
TCP_RR 1/1	502	1189.22	TA/Sek.	37.75	37.73
TCP_STREAM 4096	502	21.48	10^6 bits/Sek.	60.92	91.88
TCP_RR 1/1	2	4714.10	TA/Sek.	35.36	29.12
TCP_STREAM 4096	2	44.59	10^6 bits/Sek.	89.87	90.58

Tabelle 6.9: Testergebnisse bei unterschiedlicher Anzahl Security Policies

Das Feld id ist Primärschlüssel. Das Feld username enthält den Unix-Login-Namen des Benutzers. Das Feld hostid enthält eine laufende Nummer, aus der zusammen mit dem Netzwerkpräfix die IPv6-Adresse des Nutzers gebildet werden kann. Die Hostid wird dabei als Interface-Identifier der Länge 64 Bit an das Präfix angehängt. Das Feld gesperrt ist 0, wenn der Benutzer zur Nutzung des Tunnelservers freigegeben ist. Jeder andere Wert verbietet die Nutzung des Servers. Das Feld pubkey schließlich enthält den öffentlichen Schlüssel des Nutzers in hexadezimaler Darstellung. Die Felder hostid und username müssen jeweils eindeutig sind.

Aus dieser Tabelle wird durch das im Anhang angegebene Skript `make-ipsec-conf.pl` eine gültige Konfigurationsdatei für Pluto erzeugt. Dies kann manuell erfolgen oder mit Hilfe eines Cronjobs automatisiert ablaufen. Zu diesem Zweck ist in der Datei `/etc/crontab` folgender Eintrag (einzeilig) vorzunehmen:

```
*/30 * * * * root /usr/local/v6/sbin/make-ipsec-conf.pl >
/usr/local/v6/etc/ipsec.conf.neu && mv /usr/local/v6/etc/ipsec.conf.neu
/usr/local/v6/etc/ipsec.conf
```

Um die die Abschnitt 5.4 definierten Wartungsarbeiten an der Benutzerdatenbank vorzunehmen, wurde ein weiteres Skript implementiert (`ipsec-admin.pl`, s. Anhang), das von den Administratoren des Tunnelservers genutzt werden kann. Dieses Skript unterstützt folgende Operationen auf der Benutzerdatenbank:

- ◇ Anlegen eines neuen Nutzers mit automatischer Vergabe der Hostid und Anlegen eines leeren öffentlichen Schlüssels.
- ◇ Ändern eines öffentlichen Schlüssels.
- ◇ Sperren/Freigeben eines Nutzers.
- ◇ Löschen eines Nutzers.
- ◇ Anzeigen aller Parameter eines Nutzers.

Änderungen werden beim nächsten Lauf von `make-ipsec-conf.pl` in die Datei `ipsec.conf` übernommen. Als hostid wird beim Anlegen eines Nutzers immer die kleinste noch nicht vergebene Hostid vergeben. Da vor jeder Aufnahme einer IPSec-Verbindung pluto angewiesen wird, die Benutzerparameter neu aus der Datei `ipsec.conf` zu laden, ist ein Neustart des pluto nicht nötig.

Zusätzlich zum Skript für Administratoren wurde ein PHP-Skript entwickelt, mit Hilfe dessen ein CIP-Benutzer sich für die Nutzung des Tunnelservers anmelden kann. Die gesamte Kommunikation zwischen dem WWW-Browser des Nutzers und dem Webserver wird dabei durch eine SSL-Verbindung geschützt. Nach Eingabe seinen CIP-Login-Namens und des dazugehörigen Passworts wird die Gültigkeit des Passworts überprüft und der Nutzer dadurch autorisiert. Anschließend wird für diesen Nutzer ein Zertifikat erzeugt und der öffentliche Schlüssel extrahiert. Danach wird `ipsec-admin.pl` aufgerufen, um den Nutzer in der Datenbank des Tunnelservers anzulegen, den öffentlichen Schlüssel abzulegen und den Nutzer freizugeben. Der Nutzer erhält abschließend in seinem WWW-Browser eine Seite mit Hinweisen zur Konfiguration seines Rechners zusammen mit dem erzeugten Zertifikat. Dieses Zertifikat, dass ja auch den privaten Schlüssel enthält, wird auf dem Server nicht gespeichert, um die Anforderung der Nicht-Abstreitbarkeit zu erfüllen.

6.7 Bewertung der Sicherheit

Um die Sicherheit bei der Nutzung des Tunnelservers zu beurteilen, ist es nötig, alle dabei durchgeführten Prozesse zu betrachten.

Die im letzten Abschnitt vorgestellte Anmeldung am Tunnelserver ist sicher, wenn man voraussetzt, dass die Verwaltung der Passwörter im CIP sowie beim Nutzer selbst sicher ist. Ist der Nutzer angemeldet, so besitzt er ein Zertifikat, dass ihn zur Benutzung des Tunnelservers berechtigt. Um eine Tunnelverbindung aufzubauen, sendet er allein seinen Benutzernamen über eine unverschlüsselte Verbindung durch ein unsicheres Netz. Dieser Vorgang kann daher von Angreifern jederzeit ebenfalls durchgeführt werden. Eine Sicherung dieser Übertragung sollte daher durchgeführt werden. Ein Angreifer, der jedoch unberechtigterweise einen fremden Benutzernamen an den Server sendet, erhält jedoch keinen Zugang zum CIP-Netz, da dieser Zugang erst dann gewährt wird, wenn eine erfolgreiche Authentifizierung per IPSec durchgeführt wurde. Diese Authentifizierung ist jedoch nur dem Besitzer des ausgestellten Zertifikats möglich. im schlimmsten Fall führt dieser Missbrauch also keinesfalls zu einer Kompromittierung des CIP-Netzes, sondern einzig zur Unterbrechung der Verbindung des Benutzers, dessen Benutzername missbraucht wurde. Selbst dies setzt voraus, dass der rechtmäßige Besitzer zum Zeitpunkt des Angriffs verbunden war.

Wenn auch die Sicherheit des CIP-Netzes nicht bedroht ist, so kann ein Angreifer dennoch Denial-of-Service-Angriffe durchführen, die jede sinnvolle Nutzung des Tunnelservers verhindern. Aus diesem Grund sollte an dieser stelle ein weiterer Sicherheitsmechanismus eingeführt werden.

Die Sicherheit der weiteren Prozesse, die beim Aufbau und dem Betrieb der gesicherten Verbindung durchgeführt werden, ist abhängig von der Sicherheit der USAGI-Implementierung des IPv6-Stacks sowie des Pluto-Dienstes zum Schlüsselaustausch. Da das Projekt selbst sich noch in der Entwicklungsphase befindet, kann davon ausgegangen werden, dass die Anforderungen an die Sicherheit zum jetzigen Zeitpunkt noch nicht erfüllt werden können. Vom Einsatz im Produktionsbetrieb in einer sensiblen Umgebung wird daher abgeraten.

Kapitel 7

Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Parallelbetrieb von IPv4 und IPv6 am CIP-Pool konzipiert und realisiert. Dabei wurde die USAGI-Implementierung des IPv6-Stacks für Linux verwendet, die erheblich mehr Features bietet als die im Standardkernel enthaltene Implementierung. Darüberhinaus wurden umfangreiche Tests hinsichtlich der Performance dieser Implementierung im Vergleich zu IPv4 durchgeführt.

Um Remote-Nutzern die Möglichkeit zum Zugriff zum CIP-Pool zu bieten, wurde eine Zugangsmöglichkeit geschaffen, wobei besonders Aspekte der Sicherheit berücksichtigt wurden. Diese Möglichkeit ist sowohl entfernten stationären als auch mobilen Nutzern zugänglich und stattet Nutzer mit einer festen IPv6-Adresse aus. Für das Management dieser Benutzer wurde eine einfache PKI realisiert, die die grundlegenden Funktionalitäten des Zertifikatsmanagements zur Verfügung stellt. Zusätzlich wurden Tools vorgestellt, die bei der Überwachung der Dienstgüte sowie der Fehlerbehebung von Nutzen sind.

Einführung und Anforderungsanalyse

In der Einführung wurde die grundlegende Problemstellung zusammengefasst und in Form eines Szenarios verdeutlicht. Anschließend wurden die neuen Möglichkeiten, die IPv6 bietet, angedeutet und wie diese Möglichkeiten Ansätze zu Lösung des Problems liefern können.

Im Rahmen der Anforderungsanalyse wurde der Problembereich in verschiedene Einheiten unterteilt:

- Als technische Anforderungen wurde im wesentlichen festgelegt, dass soweit möglich auf vorhandene Infrastruktur zurückzugreifen ist, um die Einrichtung des Parallelbetriebs ohne tiefgreifende Änderungen am Gesamtsystem durchführen zu können.

- Die primäre Anforderung der Benutzer besteht darin, dass alle bereits vorhandenen Dienste wie gewohnt auch nach der Einführung des Parallelbetriebs funktionieren müssen. Zusätzlich sollen den Benutzern neue Möglichkeiten eröffnet werden: Es soll Remote-Nutzern gestattet werden, auf sichere Art und Weise Zugang zum Institutsnetz zu erhalten, wenn sie sich in einem fremden Netz befinden. Gleichzeitig soll es ihnen ermöglicht werden, für ihre Arbeit relevante Dienste aus einem entfernten Netz im Institut zur Verfügung stellen zu können.

- Im Bereich Management wurden verschiedene Anforderungen bezüglich Fehlermanagement, Konfigurationsmanagement, Abrechnungs- und Benutzermanagement, Leistungsmanagement sowie des Sicherheitsmanagements formuliert.

- Als organisatorische Anforderungen wurde die Notwendigkeit der Abstimmung mit dem Leibniz-Rechenzentrum, insbesondere in Fragen der Sicherheit und der Netzanbindung, betont.

- Um die Sicherheitsanforderungen zu identifizieren, wurden zuerst Angreifer und Angriffsmethoden kategorisiert und darauf basierend Folgerungen entwickelt. Diese umfassen die Forderungen nach Vertraulichkeit, Authentifizierung, Autorisierung, Integrität und Nicht-Abstreitbarkeit.

Grundlagen von IPv6

Um eine möglichst umfassende Lösung für die Problemstellung anbieten zu können, wurden daraufhin die neuen Möglichkeiten des Protokolls IPv6 dargestellt und analysiert. Neben den neuen Konzepten wie dem immens vergrößerten Adressraum, alternativen Adressierungsmöglichkeiten wie Anycast und Multicast oder verketteten Headern wurden die verschiedenen Möglichkeiten der Autokonfiguration sowie der Transition von IPv4 auf IPv6 vorgestellt. Hinsichtlich des Anforderungskatalogs wurde dabei auch das Protokoll Mobile-IPv6 betont, das Benutzern die Nutzung der selben IPv6-Adresse unabhängig von ihrem Standort erlaubt.

Sicherheitsaspekte

Ein besonderes Anliegen dieser Arbeit ist die Sicherung der Kommunikation zwischen Remote-Nutzer und Institut. Daher wurden die wesentlichen Grundlagen der Kryptographie vorgestellt und die Konzepte der asymmetrischen Verschlüsselung als ideal zur Lösung der Sicherheitsanforderungen identifiziert. Zur Nutzung dieser Art Algorithmen ist jedoch ein hoch entwickeltes Schlüsselmanagement nötig, das man als Public Key Infrastructure (PKI) bezeichnet. Aufgabe einer PKI sind Schlüsselerzeugung, Überprüfung der Identität eines Nutzers, sowie Ausgabe, Widerruf und Erneuern von Zertifikaten und Schlüsselpaaren. Als hilfreich erweist es sich, dass mit AH und ESP bereits Mechanismen in IPv6 enthalten sind, die die Sicherung einer Kommunikationsverbindung erlauben.

Implementierungskonzept

Auf diesen Grundlagen wurde ein Konzept entwickelt, das die Anforderungen umsetzen soll. Um grundsätzliche Konnektivität mit IPv6 herzustellen, soll parallel zum bestehenden IPv4-Stack ein IPv6-Stack installiert werden (Dual-IP-Layer). Dadurch ist zusätzlich sichergestellt, dass Anwendungen, die nur IPv4 kennen, ohne Beeinträchtigung weiterhin verwendet werden können. Nachdem der zur Verfügung stehende Adressraum entsprechend den Anforderungen für die verschiedenen Anwendungen sowie die unterschiedlichen Bereiche des Instituts aufgeteilt wurde, wurde festgelegt, dass zur automatischen Konfiguration der Endgeräte die statuslose Konfiguration mit Router Advertisements verwendet werden. Dieser Mechanismus ist einfach und daher robust und wenig fehleranfällig.

Um Remote-Nutzer an das Institutsnetz anzubinden, wurde ein Verfahren des mehrfachen Tunnelings entwickelt. Die grundlegenden Annahmen setzen dabei voraus, dass Nutzer mit IPv4 angebunden sind, keine feste IP-Adresse haben und die Kommunikation zwischen Nutzern und Institut verschlüsselt werden soll.

Um Managementmöglichkeiten zu bieten, enthält die gewählte Implementierung des IPv6-Stacks mehrere Tools, die teilweise ihre Entsprechungen für IPv4 haben, teils aber auch auf die neuen Features von IPv6 zugeschnitten sind.

Implementierung

Basierend auf dem entwickelten Konzept wurde mit Hilfe der neuen USAGI-Implementierung für Linux ein Router implementiert, der folgende Aufgaben erfüllt:

- Herstellung einer Verbindung zum Uplink
- Automatische Adresskonfiguration der Institutsrechner
- Herstellen einer Möglichkeit zum sicheren Zugang für Remote-Nutzer

◇ Paketfilter zum Schutz vor Angriffen

Mit der nun vorhandenen Infrastruktur wurde der Einsatz des Protokolls Mobile-IPv6 getestet, dass Benutzern die Möglichkeit eröffnet, unabhängig von ihrem Standort die gleiche IPv6-Adresse zu verwenden, ohne dabei die Performanceverluste durch Dreiecksrouting in Kauf nehmen zu müssen.

Zur Verwaltung der Remote-Benutzer wurde ein Mechanismus entwickelt, der ohne den Overhead einer vollständigen PKI auf die Anforderungen des Instituts zugeschnitten ist. Dadurch wird es den Nutzern erlaubt, sich selbstständig ohne manuellen Eingriff durch Institutsmitarbeiter für die Nutzung des Remote-Zugangs anzumelden, während den Administratoren weiterhin die Möglichkeit offensteht, die Benutzerdatenbank manuell zu kontrollieren.

Abschließend wurde die Performance von IPv6 unter verschiedenen Aspekten getestet. Während Leistungsfähigkeit des neuen Protokolls bei Verwendung unverschlüsselter Verbindungen nur wenig niedriger als bei IPv4 ausfällt, sinkt der Datendurchsatz immens, wenn die Verbindung verschlüsselt wird. Als begrenzender Faktor hat sich hierbei die CPU-Leistung der getesteten Endgeräte erwiesen. Außerdem scheint die getestete Implementierung noch nicht vollständig optimiert zu sein, so dass bei weiterer Entwicklung eine nicht unerhebliche Leistungssteigerung zu erwarten ist.

Ausblick

Weitere interessante Features von IPv6 sind mangels Anwendungen, die sie nutzen könnten, nicht behandelt worden. Darunter fallen Multicasting und Quality-of-Service. Da sowohl die USAGI-Implementierung selbst als auch die zugrundeliegenden Spezifikation der IETF noch im Wandel begriffen sind, ist für die Zukunft zu erwarten, dass sowohl bei den Features als auch deren Implementierung noch Änderungen stattfinden werden. Darüberhinaus wird die Einführung nativer IPv6-Zugänge für Endbenutzer gerade die Anbindung von Remote-Nutzern erheblich vereinfachen, da die Protokollumsetzung von IPv4 auf IPv6 entfallen kann.

Anhang A

Skripte

A.1 Tunneling-Server

A.1.1 Boot-Skripte

Das folgende Skript konfiguriert das IPv6-Interface des Servers/Routers und aktiviert den radvd, der die anderen Endgeräte im Netz mit einem Präfix versorgt.

```
#!/bin/sh

PATH=/usr/local/v6/sbin:/usr/local/v6/bin:/sbin:/bin:/usr/sbin:/usr/bin

test -x /usr/local/v6/sbin/radvd || exit 0

case "$1" in
    start)
        echo -n "Starting IPv6 router advertising service: radvd"
        echo 1 > /proc/sys/net/ipv6/conf/all/forwarding

        ip addr add 3ffe:400:30:1::1/64 dev eth0

        ip addr add fec0::1:0:0:0:1/64 dev eth0

        start-stop-daemon --start --quiet \
            --pidfile /var/run/radvd.pid --exec /usr/local/v6/sbin/radvd -- $OPTS
        echo "."
    ;;

    stop)
        echo -n "Stopping IPv6 router advertising service: radvd"
        kill `cat /var/run/radvd.pid`

        ip addr del 3ffe:400:30:1::1/64 dev eth0

        ip addr del fec0::1:0:0:0:1/64 dev eth0

        echo "."
    ;;
```

```
    restart)
        $0 stop
        sleep 2
        $0 start
    ;;

    *)
        echo "Usage: /etc/init.d/radvd {start|stop|restart}" >&2
        exit 1
    ;;
esac

exit 0
```

Das folgende Skript startet die Infrastruktur, die Remote-Clients benötigen, um die Tunnelverbindung zu starten.

```
#!/bin/sh

PATH=/usr/local/v6/sbin:/usr/local/v6/bin:/sbin:/bin:/usr/sbin:/usr/bin


case "$1" in
    start)
        echo  "Starting IPSec-IPv6 Tunneling Server"

        modprobe ipv6_tunnel

        ip addr add fec1::1/128 dev eth0

        # das interface braucht etwas, bis es hochkommt
        sleep 2

        start-stop-daemon --start --quiet \
            --pidfile /var/run/pluto.pid --exec /usr/local/v6/sbin/pluto

        # fuer debugging
        # pluto  --nofork --stderrlog --debug-all &> /tmp/plog &


        ipsec auto --ready

        /usr/local/v6/sbin/tunnelserver.pl  &> /var/log/tunnelserver.log &


        # erzeuge neuen Chain für die Remote-Nutzer
        ip6tables -N remote

        # und schicken deren Traffic dahin
        ip6tables -A FORWARD -s 3ffe:400:30:8000::/50 -j remote

        # verbiete alles
```

```
        ip6tables -A remote -j REJECT

        iptables -I INPUT -p tcp --dport 2345 -j ACCEPT

    ;;

    stop)
        echo  "Stopping IPSec-IPv6 Tunneling Server"
        kill `cat /var/run/pluto.pid`
        kill `cat /var/run/tunnelserver.pid`

        iptables -D INPUT -p tcp --dport 2345 -j ACCEPT

        ip6tables -D FORWARD -s 3ffe:400:30:8000::/50 -j remote
        ip6tables -F remote
        ip6tables -X remote

        ip addr del fec1::1/128 dev eth0

        rmmod ipv6_tunnel
    ;;

    restart)
        $0 stop
        sleep 2
        $0 start
    ;;

    *)
        echo "Usage: /etc/init.d/$0 {start|stop|restart}" >&2
        exit 1
    ;;
esac

exit 0
```

A.1.2 Skripte zum Aufbau einer gesicherten Tunnel-Verbindung

Serverseitig teilt sich die Funktionalität zum Aufbau einer gesicherten Verbindung auf zwischen einem einfachen, multithreaded Server in Perl, die die TCP-Verbindung entgegennimmt, die User-ID ermittelt und alle relevanten Parameter einem Shell-Skript übergibt, das die eigentliche Arbeit übernimmt.

```
#!/usr/bin/perl -w

use strict;
use Socket;
use Carp;
use DBI;

$ENV{'PATH'}="/usr/local/v6/sbin:/bin:/usr/bin";
```

```
my $dbserver="ip6gate-a.rz";
my $dbname="ip6gate";
my $dbuser="ip6gate";
my $dbpass="ip6gate";
my $dbdriver="mysql";
my $dbport=3306;

my $dsn = "DBI:$dbdriver:database=$dbname;host=$dbserver;port=$dbport";

sub spawn;  # forward declaration
sub logmsg { print "$0 $$: @_ at ", scalar localtime, "\n" }

my $port = shift || 2345;
my $proto = getprotobyname('tcp');

my $prefix="3ffe:400:30:";


($port) = $port =~ /^(\d+)$/                        or die "invalid port";

open PID,">/var/run/tunnelserver.pid";
print PID $$;
close PID;

socket(Server, PF_INET, SOCK_STREAM, $proto)        || die "socket: $!";
setsockopt(Server, SOL_SOCKET, SO_REUSEADDR,
                                    pack("l", 1))   || die "setsockopt: $!";
bind(Server, sockaddr_in($port, INADDR_ANY))        || die "bind: $!";
listen(Server,SOMAXCONN)                            || die "listen: $!";

logmsg "server started on port $port";

my $waitedpid = 0;
my $paddr;

sub REAPER {
    $waitedpid = wait;
    $SIG{CHLD} = \&REAPER;  # loathe sysV
    logmsg "reaped $waitedpid" . ($? ? " with exit $?" : '');
}

$SIG{CHLD} = \&REAPER;


for ( $waitedpid = 0;
      ($paddr = accept(Client,Server)) || $waitedpid;
      $waitedpid = 0, close Client)
{
    next if $waitedpid and not $paddr;
    my($port,$iaddr) = sockaddr_in($paddr);
    my $name = gethostbyaddr($iaddr,AF_INET);

    logmsg "connection from $name [",
            inet_ntoa($iaddr), "]
```

```
            at port $port";

    spawn sub {
        $|=1;
        print STDERR "Woker called\n";
        my $username=<STDIN>;
        chomp $username;
        if ($username!~/^[a-zA-Z0-9_-]*$/) {
                die("inavlid username $username");
        }

        my $dbh = DBI->connect($dsn, $dbuser, $dbpass);

        my $query="select hostid from users where gesperrt=0 and username=\"$username\"";
        my $sth = $dbh->prepare($query);
        $sth->execute;
        my @row=$sth->fetchrow_array();
        my $uid=$row[0];
        $sth->finish();
        $dbh->disconnect();


        print STDERR "Uid: $uid\n";
        if (defined $uid && $uid>99 && $uid<16384) {
            print STDERR "starting tunnel for $uid\n";
            print "$uid\n";
            my $iaddra=inet_ntoa($iaddr);
                delspd(sprintf("%x",$uid));

            system("/usr/local/v6/sbin/setup_tunnel.sh $username $uid $iaddra $prefix");

        }
        else {
            print "User unknown\n";
        }
        print STDERR "Worker end\n";
    }



}



sub spawn {
    my $coderef = shift;

    unless (@_ == 0 && $coderef && ref($coderef) eq 'CODE') {
        confess "usage: spawn CODEREF";
    }

    my $pid;
    if (!defined($pid = fork)) {
        logmsg "cannot fork: $!";
```

```
        return;
    } elsif ($pid) {
        logmsg "begat $pid";
        return; # I'm the parent
    }
    # else I'm the child -- go spawn

    $SIG{ALRM} = sub { exit 99; };
    alarm 300;
    open(STDIN,  "<&Client")   || die "can't dup client to stdin";
    open(STDOUT, ">&Client")   || die "can't dup client to stdout";
    ## open(STDERR, ">&STDOUT") || die "can't dup stdout to stderr";
    exit &$coderef();
}



# die spd muss manuell geloescht werden, da es sonst zu konflikten kommt
sub delspd {
    my $suffix=shift;
    my ($line,@a,$pfkey);

    open PFKEY,"pfkey -L |";
    while ($line=<PFKEY>) {
      if ($line=~/^spd/) {
        $line=<PFKEY>;
        if ($line=~/$suffix/) {
          @a=split / /,$line;
          $pfkey="pfkey -D sp -s $a[0] -d $a[2]";
          $line=<PFKEY>;
          chomp $line;
          @a=split / /,$line;
          $pfkey.=" --sad $a[1] -S $a[3] --sport 0 --dport 0 -p any --tunnel -T esp";
          system $pfkey;
        }
     }
   }
   close PFKEY;
}
```

Hier das dazugehörige Shell-Skript:

```
#!/bin/bash

username=$1
uid=$2
netid=$(($uid+32768))
hexnetid=`printf "%x" $netid`
iaddr=$3
prefix=$4
```

```
ipv6tunnel del tnl$uid
ip route del ${prefix}${hexnetid}::/64
ip tunnel del sit$uid
ipsec auto --delete $username

pfkey -D sp -T esp -s 3ffe:400:30:1::/48 -d ${prefix}${hexnetid}::/64
pfkey -D sp -T esp -d 3ffe:400:30:1::/48 -s ${prefix}${hexnetid}::/64


## und das neue hochfahren
ip tunnel add sit$uid mode sit local any remote $iaddr ttl 64
ip link set sit$uid up
ip route add fec1::${hexnetid}/128 dev sit$uid

ipv6tunnel add tnl$uid --tunnel-local-packets encaplimit 0 \
   remote fec1::${hexnetid}  local fec1::1
ip link set tnl$uid up
ip addr add fe80::20 dev tnl$uid

ip -6 route add ${prefix}${hexnetid}::/64 dev tnl$uid


ipsec auto --add $username

sleep 5
ipsec auto --up $username
```

Sobald Pluto die Verbindung aufgebaut hat, aber auch beim Abbau der Verbindung, wird folgendes Skript aufgerufen:

```
#!/bin/sh

case "$PLUTO_VERB" in

        up-client-v6)
                ip6tables -I remote -s $PLUTO_PEER_CLIENT -j ACCEPT
                ip6tables -I remote -d $PLUTO_PEER_CLIENT -j ACCEPT
        ;;

        down-client-v6)
                ip6tables -D remote -s $PLUTO_PEER_CLIENT -j ACCEPT
                ip6tables -D remote -d $PLUTO_PEER_CLIENT -j ACCEPT
        ;;

esac
```

A.1.3 Skripte zum Schlüsselmanagement

Das folgende Skript dient dem Zweck, Remote-User zu verwalten. Nutzer können angelegt, gelöscht, gesperrt und mit einem anderen öffentlichen Schlüssel ausgestattet werden.

```
#!/usr/bin/perl

use DBI;

$dbserver="ip6gate-a.rz";
$dbname="ip6gate";
$dbuser="ip6gate";
$dbpass="ip6gate";
$dbdriver="mysql";
$dbport=3306;

$dsn = "DBI:$dbdriver:database=$dbname;host=$dbserver;port=$dbport";

$dbh = DBI->connect($dsn, $dbuser, $dbpass);

$error=0;

if ($#ARGV<1) {
        usage();
        exit(1);
}

$action=$ARGV[0];

if ($action eq "setkey" && $#ARGV<2) {
        print "werew".$#ARGV;
        usage();
        exit(1);
}

if ($action ne "show" && $action ne "disable" && $action ne "enable" &&
        $action ne "add" && $action ne "delete" && $action ne "setkey") {
        print "asds".$#ARGV;
        usage();
        exit(1);
}

$username=$ARGV[1];
$newkey=$ARGV[2];       # ist definiert, wenn action==setkey

if ($action eq "show") {
        $query="select username,hostid,gesperrt,pubkey from users ";
        $query.="where username=\"$username\"";
}
if ($action eq "disable") {
        $query="update users set gesperrt=1 where username=\"$username\"";
}
if ($action eq "enable") {
        $query="update users set gesperrt=0 where username=\"$username\"";
}
if ($action eq "delete") {
        $query="delete from users where username=\"$username\"";
}
if ($action eq "setkey") {
```

```
        $query="update users set pubkey=\"$newkey\" where username=\"$username\"";
}

if ($query) {
        $sth = $dbh->prepare($query);
        $sth->execute();

        if ($action eq "show") {
                if (@row=$sth->fetchrow_array()) {
                        print "Username: $row[0]\n";
                        print "Hostid:   $row[1]\n";
                        print "Gesperrt: $row[2]\n";
                        print "Pubkey:   $row[3]\n";
                }
                else {
                        print "User $username exisitiert nicht!\n";
                        $error=1;
                }
        }
        else {
                if ($sth->rows==1) {
                        print "Aktion $action ausgefuehrt.\n";
                }
                else {
                        if ($dbh->errstr) {
                                print "  Mysql-Fehler: ".$dbh->errstr."\n";
                        }
                        print "Aktion $action wurde nicht ausgefuehrt.\n";
                        $error=1;
                }
        }
}
else {          # action=add
        $sth=$dbh->prepare("select id from users where username=\"$username\"");
        $sth->execute();
        if ($sth->rows>0) {
                print "Benutzer $username existiert bereits.\n";
                $error=1;
        }
        else {
                # sperre tabelle fuer read und write
                $sth=$dbh->prepare("lock tables users write");
                $sth->execute();

                $sth=$dbh->prepare("select max(hostid) from users");
                $sth->execute();

                @row=$sth->fetchrow_array();
                $newhostid=$row[0]+1;

                $query="insert into users (username,hostid,gesperrt) ";
                $query.="values (\"$username\",$newhostid,1)";
                $sth=$dbh->prepare($query);
                $sth->execute();
```

```
                if ($sth->rows!=1) {
                        print "Fehler: Insert gescheitert.".$dbh->errstr."\n";
                        $error=1;
                }
                else {
                        print "Benutzer $username hinzugefuegt. Hostid: $newhostid\n";
                }

                $sth=$dbh->prepare("unlock tables");
                $sth->execute();
        }
}


if ($sth) { $sth->finish(); }

$dbh->disconnect();

exit($error);


sub usage() {
        print <<EOF;
Usage: $0 [show|disable|enable|add|delete] <username>
   or: $0 setkey <username> <newkey>

EOF
}
```

Um die mit obigen Skript verwalteten Benutzerdaten dem IKE-Server (pluto) zur Verfügung zu stellen, muss aus der Datenbank regelmäßig die Konfigurationsdatei `/usr/local/v6/etc/ipsec.conf` erzeugt werden. Dies erledigt folgendes Skript.

```
#!/usr/bin/perl

use DBI;

$dbserver="ip6gate-a.rz";
$dbname="ip6gate";
$dbuser="ip6gate";
$dbpass="ip6gate";
$dbdriver="mysql";
$dbport=3306;

$dsn = "DBI:$dbdriver:database=$dbname;host=$dbserver;port=$dbport";

$dbh = DBI->connect($dsn, $dbuser, $dbpass);

$query="select username,hostid,pubkey from users where gesperrt=0";
$sth = $dbh->prepare($query);
$sth->execute;
```

```
print <<EOF;
config setup
        interfaces=%defaultroute
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search

conn %default
        keyingtries=1
        af=inet6
        type=tunnel
        authby=rsasig
        leftrsasigkey=0sAQON7fqjb69v+ygk.....
        left=fec1::1
        rightnexthop=fec1::1
        leftsubnet=3ffe:400:30::/49
        leftupdown=/usr/local/v6/sbin/tunnel.updown.sh

EOF


while (@row=$sth->fetchrow_array()) {
        $hexnetid=sprintf("%x",$row[1]+32768);
        print <<EOF;

conn $row[0]
        rightrsasigkey=$row[2]
        right=fec1::$hexnetid
        rightsubnet=3ffe:400:30:${hexnetid}::/64
        leftnexthop=fec1::$hexnetid

EOF
}
$sth->finish();

$dbh->disconnect();
```

A.2 Remote-Nutzer

A.2.1 Skripte zum Aufbau einer gesicherten Tunnel-Verbindung

Clientseitig teilt sich die Funktionalität zum Aufbau einer gesicherten Verbindung auf zwischen einem einfachen TCP-Client, der den Login-Namen des Nutzers übermittelt, die User-ID vom Server entgegennimmt und alle relevanten Parameter einem Shell-Skript übergibt, das die eigentliche Arbeit übernimmt.

```
#!/usr/bin/perl -w

use IO::Socket;

$ENV{'PATH'}="/usr/local/v6/sbin:/usr/bin:/bin";
```

```
my $server="141.84.214.70";
my $prefix="3ffe:400:30:";
my $route="2000::/3";

$remote = IO::Socket::INET->new(
                    Proto    => "tcp",
                    PeerAddr => "$server",
                    PeerPort => "2345",
                )
              or die "cannot connect to $server";


print $remote "$ARGV[0]\n";
my $answer=<$remote>;
chomp $answer;
if ($answer=~/^\d+$/) {
    system("bash client.sh $ARGV[0] $prefix $route $server $answer &> /tmp/log");
    sleep 5;
    print "Tunnel up\n";
}
else {
    print "Fehler: $answer\n";
}
```

Hier das dazugehörige Shell-Skript:

```
#!/bin/sh

PATH=$PATH:/sbin:/usr/local/v6/sbin

username=$1
prefix=$2
route=$3
server=$4
userid=$5

netid=$(($userid+32768))
hexnetid=`printf "%x" $netid`

## erstmal alles runterfahren, fehler ignorieren
modprobe ipv6_tunnel
ipv6tunnel del tnl0

ip route del fec1::1/128 dev sit1
ip route del $route dev tn0
ip tunnel del sit1
killall pluto
rm /var/run/pluto.*
pfkey -F any
ipsec auto --add $username
ipsec auto --ready
```

```
echo ...
## und jetzt hochfahren
ip tunnel add sit1 mode sit local any remote $server ttl 64
ip link set sit1 up
ip addr add fec1::${hexnetid}/128  dev sit1
ip route add fec1::1/128 dev sit1

ipv6tunnel add tnl0 --tunnel-local-packets encaplimit 0  \
   remote fec1::1 local fec1::${hexnetid}
ip link set tnl0 up
ip addr add fe80::10 dev tnl0

ip addr add ${prefix}${hexnetid}::1/64 dev tnl0
ip -6 route add $route dev tnl0

pluto  --nofork --stderrlog --debug-all &> /tmp/plog &
ipsec auto --add $username
ipsec auto --ready
```

A.3 Testskripte

A.3.1 Aufbau einer IPSec-Verbindung mit manuellem Keying

Das folgende Skript wird verwendet, um zwischen zwei Endgeräten eine IPSec-Verbindung mit manuellen Keying aufzubauen. Es wird AH-ESP mit den Algorithmen hmac-md5 und 3des-cbc verwendet. Die Parameter SRC und DEST enthalten die IPv6-Adressen der beiden Endgeräte. Auf beiden Rechner ist dasselbe Skript auszuführen.

```
#!/bin/sh

PATH=/usr/local/v6/sbin:$PATH
pfkey -F any

SRC=fec0:0:0:2::1
DEST=fec0::2:204:75ff:fe7c:9d05

PROTO=any

pfkey -A sa -s $SRC -d $DEST -T esp -S 0x5678 -p $PROTO --auth hmac-md5 \
  --authkey 0x0123456789abcdef0123456789abcdef --esp 3des-cbc \
  --espkey 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7

pfkey -A sp -s $SRC -d $DEST -T esp -S 0x5678 -p $PROTO

pfkey -A sa -d $SRC -s $DEST -T esp -S 0xdef0 -p $PROTO  --auth hmac-md5 \
  --authkey 0x0123456789abcdef0123456789abcdef --esp 3des-cbc \
  --espkey 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7

pfkey -A sp -d $SRC -s $DEST -T esp -S 0xdef0 -p $PROTO
```

A.3.2 Massenkonfiguration von Sicherheitsbeziehungen

Dieses Skript wurde eingesetzt, um auf zwei Endgeräten insgesamt jeweils 2002 Einträge in die Security Policy Database vorzunehmen. Die Parameter SRC und DEST sind auf der jeweils anderen Maschine zu vertauschen, damit jede der redundanten Regeln die Adresse des Endgeräts selbst enthält.

```
#!/bin/sh

PATH=/usr/local/v6/sbin:$PATH
pfkey -F any

SRC=fec0:0:0:2::1
DEST=fec0::2:204:75ff:fe7c:9d05
PROTO=any

count=1000

function setup_pfkey_trash() {
    DESTNET=$1
    SPIPRE1=$2
    SPIPRE2=$3
    host=0
    while test $host -lt $count ; do
        host=$(($host+1))
        echo $host
        hostx=`printf %04x $host`
        authkey=0x0123456789abcdef0123456789ab$hostx
        espkey=0xa7a36ebd91863edfba763fa7edcba64d89123ace6359$hostx
        pfkey -A sa -s $SRC -d $DESTNET$hostx -T esp -S 0x${SPIPRE1}${hostx} \
          -p $PROTO --auth hmac-md5 --authkey $authkey --esp 3des-cbc --espkey $espkey
        pfkey -A sp -s $SRC -d $DESTNET$hostx -T esp -S 0x${SPIPRE1}${hostx} \
          -p $PROTO

        pfkey -A sa -d $SRC -s $DESTNET$hostx -T esp -S 0x${SPIPRE2}${hostx} \
          -p $PROTO  --auth hmac-md5 --authkey $authkey --esp 3des-cbc --espkey $espkey

        pfkey -A sp -d $SRC -s $DESTNET$hostx -T esp -S 0x${SPIPRE2}${hostx} \
          -p $PROTO
    done
}

function setup_pfkey_real {

    PROTO=any

    pfkey -A sa -s $SRC -d $DEST -T esp -S 0x5678 -p $PROTO --auth hmac-md5\
      --authkey 0x0123456789abcdef0123456789abcdef --esp 3des-cbc \
      --espkey 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7

    pfkey -A sp -s $SRC -d $DEST -T esp -S 0x5678 -p $PROTO

    pfkey -A sa -d $SRC -s $DEST -T esp -S 0xdef0 -p $PROTO  --auth hmac-md5 \
      --authkey 0x0123456789abcdef0123456789abcdef --esp 3des-cbc \
```

```
      --espkey 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7

    pfkey -A sp -d $SRC -s $DEST -T esp -S 0xdef0 -p $PROTO
}



setup_pfkey_trash fec2:: aaaa bbbb
setup_pfkey_real
setup_pfkey_trash fec3:: cccc dddd
```

A.3.3 Test des maximalen Durchsatzes zwischen Endgeräten

Das folgende Skript testet des maximalen Durchsatz zwischen zwei Endgeräten bei unterschiedlicher Verbindung (IPv4/IPv6) und Verschlüsselungen.

```
#!/bin/sh

me4="141.84.220.163"
me6="fec0::2:210:5aff:fe31:25f6"

host4="141.84.220.173"
host6="fec0::2:210:5aff:fe31:3931"

RR_SIZES="1,1 128,8192"
SEND_SIZES="4096 32768 65536"

pfkey=/usr/local/v6/sbin/pfkey
netperf=/usr/local/netperf/netperf

authkey=0x0123456789abcdef0123456789abcdef
espkey=0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7


function test4() {

for size in $RR_SIZES; do
        echo Test TCP_RR size $size
        $netperf -l 60  -H $host4 -t TCP_RR -cC -- -r $size -s 0 -S 0
done

for size in $SEND_SIZES; do
        echo Test TCP_STREAM size $size
        $netperf -l 60  -H $host4 -t TCP_STREAM -cC -- -m $size -s 0 -S 0
done

}


function test6() {

for size in $RR_SIZES; do
```

```
        echo Test TCPIPV6_RR size $size
        # dieser test funktioniert nur, wenn debugging (-d) ein
        $netperf -d -l 60  -H $host6 -t TCPIPV6_RR -cC -- -r $size -s 0 -S 0
done


for size in $SEND_SIZES; do
        echo Test TCPIPV6_STREAM size $size
        $netperf -l 60  -H $host6 -t TCPIPV6_STREAM -cC -- -m $size -s -S
done

}


function setup_ah() {
        algo=$1
        $pfkey -A sa -s $me6 -d $host6 -T ah -S 0x1234 -p any  --auth $algo \
                --authkey $authkey
        $pfkey -A sp -s $me6 -d $host6 -T ah -S 0x1234 -p any

        $pfkey -A sa -d $me6 -s $host6 -T ah -S 0x9abc -p any  --auth $algo \
                --authkey $authkey
        $pfkey -A sp -d $me6 -s $host6 -T ah -S 0x9abc -p any


        ssh $host4 $pfkey -A sa -d $me6 -s $host6 -T ah -S 0x9abc -p any  \
                --auth $algo  --authkey $authkey
        ssh $host4 $pfkey -A sp -d $me6 -s $host6 -T ah -S 0x9abc -p any

        ssh $host4 $pfkey -A sa -s $me6 -d $host6 -T ah -S 0x1234 -p any  \
                --auth $algo --authkey $authkey
        ssh $host4 $pfkey -A sp -s $me6 -d $host6 -T ah -S 0x1234 -p any

}

function setup_ahesp() {
        authalgo=$1
        espalgo=$2

        $pfkey -A sa -s $me6 -d $host6 -T esp -S 0x5678 -p any \
                --auth $authalgo --authkey $authkey \
                --esp $espalgo --espkey $espkey
        $pfkey -A sp -s $me6 -d $host6 -T esp -S 0x5678 -p any
        $pfkey -A sa -d $me6 -s $host6 -T esp -S 0xdef0 -p any \
                --auth $authalgo --authkey $authkey \
                --esp $espalgo --espkey $espkey
        $pfkey -A sp -d $me6 -s $host6 -T esp -S 0xdef0 -p any


        ssh $host4 $pfkey -A sa -d $me6 -s $host6 -T esp -S 0xdef0 -p any \
                --auth $authalgo --authkey $authkey \
                --esp $espalgo --espkey $espkey
        ssh $host4 $pfkey -A sp -d $me6 -s $host6 -T esp -S 0xdef0 -p any
        ssh $host4 $pfkey -A sa -s $me6 -d $host6 -T esp -S 0x5678 -p any \
```

```
                --auth $authalgo --authkey $authkey \
                --esp $espalgo --espkey $espkey
        ssh $host4 $pfkey -A sp -s $me6 -d $host6 -T esp -S 0x5678 -p any
}


function flush_pfkey() {
        $pfkey -F any
        ssh $host4 $pfkey -F any
}

### Start tests
echo Test Connectivity

flush_pfkey
ping6 -c2  $host6
setup_ah hmac-md5
ping6 -c2 -pfeed  $host6
flush_pfkey
setup_ah hmac-md5
setup_ahesp hmac-md5  3des-cbc
ping6 -c2  -pfeed $host6
flush_pfkey

echo Start Test

test4

test6

for ah in hmac-md5 hmac-sha1; do
        echo Test AH $ah
        setup_ah $ah
        test6
        flush_pfkey

        for esp in null des-cbc 3des-cbc aes-cbc; do
                echo Test AH-ESP $ah $esp
                setup_ahesp $ah $esp
                test6
                flush_pfkey
        done

        for esp in null des-cbc 3des-cbc aes-cbc; do
                echo Test AH $ah AH-ESP $esp
                setup_ah $ah
                setup_ahesp $ah $esp
                test6
                flush_pfkey
        done

done
```

Literaturverzeichnis

[aCPo 01] aCampo, Markus und Norbert Pohlmann: *Virtual Private Networks.* mipt, 2001.

[add 02] *IPv6 Address Allocation and Assignment Policy*, 2002, http://www.iana.org/ipaddress/ipv6-allocation-policy-26jun02 .

[Aust 01] Austin, T.: *PKI.* John Wiley and Sons, 2001.

[BDH 99] Borman, D., S. Deering und R. Hinden: *IPv6 Jumbograms*, August 1999, ftp://ftp.isi.edu/in-notes/rfc2675.txt . RFC 2675.

[Bier 03] Bieringer, Peter: *Peter Bieringer's Linux-Section: IPv6*, 2003, http://www.bieringer.de/linux/IPv6/index.html .

[BuKa 00] Buchegger, B. und C. Kaindel: *n-gen - Nutzung neuer Medien durch Wiener Jugendliche*, 2000, http://www.netbridge.at/dloads/ngenstudie.pdf .

[Caes 95] Caesar, Gaius Julius: *De bello Gallico.* Vandenh. u. R, 1995.

[CaMo 01] Carpenter, B. und K. Moore: *Connection of IPv6 Domains via IPv4 Clouds*, Februar 2001, ftp://ftp.isi.edu/in-notes/rfc3056.txt . RFC 3056.

[Cent 02] Center, China Internet Network Information: *Semiannual Survey Report on the Development of China's Internet(July. 2002)*, 2002, http://www.cnnic.net.cn/develst/2002-7e/index.shtml .

[Deer 89] Deering, S.E.: *Host extensions for IP multicasting*, August 1989, ftp://ftp.isi.edu/in-notes/rfc1112.txt . RFC 1112.

[DeHi 98] Deering, S. und R. Hinden: *Internet Protocol, Version 6 (IPv6) Specification*, Dezember 1998, ftp://ftp.isi.edu/in-notes/rfc2460.txt . RFC 2460.

[Ditt 02] Dittler, Hans Peter: *IPv6 - das neue Internetprotokoll.* dpunkt-verlag, Zweite Auflage, 2002.

[DuHu 01] Durand, A. und C. Huitema: *The H-Density Ratio for Address Assignment Efficiency An Update on the H ratio*, November 2001, ftp://ftp.isi.edu/in-notes/rfc3194.txt . RFC 3194.

[ErJ 01] Eastlake, D., 3rd und P. Jones: *US Secure Hash Algorithm 1 (SHA1)*, September 2001, ftp://ftp.isi.edu/in-notes/rfc3174.txt . RFC 3174.

[FLYV 93] Fuller, V., T. Li, J. Yu und K. Varadhan: *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, September 1993, ftp://ftp.isi.edu/in-notes/rfc1519.txt . RFC 1519.

[FT 02] F. Templin, T. Gleeson, M. Talwar et al.: *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*, 2002, http://www.ietf.org/internet-drafts/draft-ietf-ngtrans-isatap-08.txt .

[Gent 98] GENTLE, JAMES E.: *Random Number Generation and Monte Carlo Methods.* Springer-Verlag Telos, 1998.

[GiNo 96] GILLIGAN, R. und E. NORDMARK: *Transition Mechanisms for IPv6 Hosts and Routers*, April 1996, ftp://ftp.isi.edu/in-notes/rfc1933.txt . RFC 1933.

[HaCa 98] HARKINS, D. und D. CARREL: *The Internet Key Exchange (IKE)*, November 1998, ftp://ftp.isi.edu/in-notes/rfc2409.txt . RFC 2409.

[HAN 99] HEGERING, HEINZ-GERD, SEBASTIAN ABECK und BERNHARD NEUMAIR: *Integriertes Management vernetzter Systeme.* dpunkt-verlag, 1999.

[HDFH 00] HINDEN, R., S. DEERING, R. FINK und T. HAIN: *Initial IPv6 Sub-TLA ID Assignments*, September 2000, ftp://ftp.isi.edu/in-notes/rfc2928.txt . RFC 2928.

[HFd 98] HINDEN, R., R. FINK und J. POSTEL (DECEASED): *IPv6 Testing Address Allocation*, Dezember 1998, ftp://ftp.isi.edu/in-notes/rfc2471.txt . RFC 2471.

[HiDe 98] HINDEN, R. und S. DEERING: *IP Version 6 Addressing Architecture*, Juli 1998, ftp://ftp.isi.edu/in-notes/rfc2373.txt . RFC 2373.

[HiDe 02] HINDEN, R. und S. DEERING: *IP Version 6 Addressing Architecture*, 2002, http://www.ietf.org/internet-drafts/draft-ietf-ipngwg-addr-arch-v3-11.txt .

[Hind 98] HINDEN, R.: *Proposed TLA and NLA Assignment Rule*, Dezember 1998, ftp://ftp.isi.edu/in-notes/rfc2450.txt . RFC 2450.

[Hind 02] HINDEN, R.: *IPng Implementations*, 2002, http://playground.sun.com/pub/ipng/html/ipng-implementations.html .

[Hoff 97] HOFFMANN, R.: *Das Internet-Protokoll Version 6 - Migrationsverfahren und Managementanforderungen.* Diplomarbeit, Universität München, 1997, http://wwwhegering.informatik.tu-muenchen.de/common/Literatur/MNMPub/Diplomarbeiten/hoff97/hoff97.shtml .

[Huit 00] HUITEMA, CHRISTIAN: *IPv6 - die neue Generation.* Addison Wesley, 2000.

[Hust 00] HUSTON, G.: *Next Steps for the IP QoS Architecture*, November 2000, ftp://ftp.isi.edu/in-notes/rfc2990.txt . RFC 2990.

[IAIE 01] IAB und IESG: *IAB/IESG Recommendations on IPv6 Address*, September 2001, ftp://ftp.isi.edu/in-notes/rfc3177.txt . RFC 3177.

[JPA] JOHNSON, DAVID B., CHARLES E. PERKINS und JARI ARKKO: *Mobility Support in IPv6*, http://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-19.txt .

[KBC 97] KRAWCZYK, H., M. BELLARE und R. CANETTI: *HMAC: Keyed-Hashing for Message Authentication*, Februar 1997, ftp://ftp.isi.edu/in-notes/rfc2104.txt . RFC 2104.

[KeAt 98a] KENT, S. und R. ATKINSON: *IP Authentication Header*, November 1998, ftp://ftp.isi.edu/in-notes/rfc2402.txt . RFC 2402.

[KeAt 98b] KENT, S. und R. ATKINSON: *IP Encapsulating Security Payload (ESP)*, November 1998, ftp://ftp.isi.edu/in-notes/rfc2406.txt . RFC 2406.

[KeAt 98c] KENT, S. und R. ATKINSON: *Security Architecture for the Internet Protocol*, November 1998, ftp://ftp.isi.edu/in-notes/rfc2401.txt . RFC 2401.

[LW 96] LARRY WALL, TOM CHRISTIANSEN, RANDAL L. SCHWARTZ: *Programming Perl.* O'Reilly, 1996.

[MaDo 98] MADSON, C. und N. DORASWAMY: *The ESP DES-CBC Cipher Algorithm With Explicit IV*, November 1998, ftp://ftp.isi.edu/in-notes/rfc2405.txt . RFC 2405.

[NaDr 01] NARTEN, T. und R. DRAVES: *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, Januar 2001, ftp://ftp.isi.edu/in-notes/rfc3041.txt . RFC 3041.

[Nash 02] NASH, A.: *PKI - e-security implementieren.* mitp, 2002.

[PeEd 96] PERKINS, C. und ED.: *IP Mobility Support*, Oktober 1996, ftp://ftp.isi.edu/in-notes/rfc2002.txt . RFC 2002.

[Post 81] POSTEL, J.: *Internet Protocol*, September 1981, ftp://ftp.isi.edu/in-notes/rfc791.txt . RFC 791.

[RD 02] R. DROMS, J. BOUND, B. VOLZ ET AL.: *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, 2002, http://www.ietf.org/internet-drafts/draft-ietf-dhc-dhcpv6-28.txt .

[Rive 92] RIVEST, R.: *The MD5 Message-Digest Algorithm*, April 1992, ftp://ftp.isi.edu/in-notes/rfc1321.txt . RFC 1321.

[RMKdG 94] REKHTER, Y., B. MOSKOWITZ, D. KARRENBERG und G. DE GROOT: *Address Allocation for Private Internets*, März 1994, ftp://ftp.isi.edu/in-notes/rfc1597.txt . RFC 1597.

[Silv 01] SILVERMAN, ROBERT D.: *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths*, 2001, http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html .

[ThHu 95] THOMSON, S. und C. HUITEMA: *DNS Extensions to support IP version 6*, Dezember 1995, ftp://ftp.isi.edu/in-notes/rfc1886.txt . RFC 1886.

[ThNa 98] THOMSON, S. und T. NARTEN: *IPv6 Stateless Address Autoconfiguration*, Dezember 1998, ftp://ftp.isi.edu/in-notes/rfc2462.txt . RFC 2462.