IPSec Protokoll Implementierung

Collin R. Mulliner

Bachelorarbeit an der Fachhochschule Darmstadt

Fachbereich Informatik

25. April 2003

IPSec Protokoll Implementierung

Collin R. Mulliner

Matrikelnummer: 564928

Referent
Prof. Dr. Woldemar Fuhrmann

Koreferent
Prof. Dr. Johannes Reichardt

Ehrenwörtliche Versicherung

Hiermit versichere ich, das ich die vorliegende Arbeit selbständig und ausschließlich mit den aufgeführten Hilfsmitteln angefertigt habe.

INHALTSVERZEICHNIS

KAPITEL

EINS

Einleitung

1.1 Motivation

Heute werden vernetzte Computersysteme großflächig eingesetzt, sei es im privaten, kommerziellen oder im staatlichen Bereich. Dabei werden gerade im kommerziellen und staatlichen Bereich oft sensible Daten verarbeitet.

Diese Daten und die dazugehörige Infrastruktur gilt es zu schützen.

Der Schutz sensibler Daten ist allerdings keine triviale Angelegenheit. Generell kann man Schutzmaßnahmen in zwei Kategorien unterteilen: in Zugriffs-Schutz (welche Person darf auf welche Daten zugreifen) und in Daten-Handhabung (wie müssen sensible Daten gehandhabt werden). Das generelle Problem stellen dabei die Benutzer dar, da diese sich häufig nicht darüber im klaren sind ob die Daten eventuell sensibel sind und wenn, wie sie diese zu behandeln haben. Selbst wenn Benutzer von der Sensibilität der Daten wissen, schützen sie diese oft aus Faulheit oder aus Unwissenheit über die mögliche Gefahren nicht.

Heute existieren eine Reihe von Lösungen welche jeweils einen Teilbereich (z.B. den sicheren Versand von Daten über Netzwerke oder die sichere Speicherung von Daten) des zu schützenden Ganzen abdecken. Eine Lösung welche alle Teilbereiche als ein Ganzes schützt, sucht man jedoch vergebens.

Am Fraunhofer Institut für Graphische Datenverarbeitung wurde aus diesem Grund das ehrgeizige COSEDA Projekt ins Leben gerufen. COSEDA steht für: Comprehensive Security for Distributed Architectures, ein System für die umfassende Sicherung von verteilten vernetzten Computersystemen.

1.2 Umfeld

Um nicht zu weit ins Detail zu gehen wird das Gesamtsystem nur kurz umschrieben. Detailliertere Informationen sind in [1] Vorhanden.

Das COSEDA System basiert auf Sicherheitspolitiken, das heißt Regeln die festlegen wie Daten behandelt werden müssen.

Einzelne Sicherheitspolitiken bestimmen zum Beispiel, daß Dateien oder bestimmte Netzwerk-Verbindungen verschlüsselt werden müssen.

Der generelle Ansatz von COSEDA ist die zentrale Definition von Sicherheitspolitiken (einfach Administration) und deren lokale (auf dem jeweiligen End-System) Durchsetzung (mehr Sicherheit).

Das wichtigste Merkmal des COSEDA Systems besteht in der Transparenz der Durchsetzung dieser Sicherheitspolitiken. Das bedeutet der Benutzer muß sich nicht um die Sicherung kümmern beziehungsweise bemerkt sie gar nicht erst.

Der Teil des COSEDA Systems der für die Durchsetzung der Sicherheitspolitiken zuständig ist, wird als eine Betriebssystem-Erweiterung entwickelt. Diese ist dann auf jedem Computersystem lokal für die Durchsetzung zuständig.

Ein wichtiger Haupt Bestandteil des COSEDA Systems ist die Kontrolle der Netzwerk-Aktivitäten. Diese Netzwerk-Sicherungs-Komponente des COSEDA Systems soll durch die Implementierung der *IP Security Architecture* (IPSec) realisiert werden.

1.3 Aufgabenstellung

Die an mich gestellte Aufgabe war:

Das Design und die Umsetzung einer IPSec Protokoll Implementierung

1.3.1 Anforderungen

Da die IPSec Architektur grob aus zwei komplett untschiedlichen Teilen besteht:

- Protokoll Implementierung
- Schlüssel-Aushandlungs Implementierung(en)

und da die Schlüssel-Aushandlung für die Implementierung der Protokolle nicht zwingend erforderlich, beziehungsweise prinzipiell überhaupt nicht erforderlich ist, sollte diese erst später gemacht werden, daher bestand die Anforderung nur aus der Protokoll Implementierung.

Bei der Protokoll Implementierung sollte besonders auf die Portierbarkeit geachtet werden. Eine portierbare Implementierung währe in diesem Fall eine Implementierung die auf verschiedenen Betriebssystemen und Hardware-Architekturen, durch einfache Neu-Kompilierung des Quellcodes, lauffähig ist.

KAPITEL

ZWEI

Einführung

Zu Beginn sollen erst einmal Grundlagen gebildet werden um so einen Einstig in das Thema zu finden. Den Anfang macht eine Übersicht über den generellen Aufbau von Schichten basierten Netzwerken. Weiter wird das Internet Protokoll und seine Eigenschaften betrachtet.

2.1 IP Netzwerke

Dieser erste Abschnitt soll ein Verständnis für den Aufbau und die Funktion von IP basierten Netzwerken vermitteln. Zu Beginn soll zunächst der Aufbau des ISO/OSI Modells, welches eine Referenz für die Bildung von Kommunikationsstandards darstellt, erläutert werden. Anhand der Beschreibung des OSI Modells folgt dann eine kurze Einführung in Thematiken des Internet Protokolls.

2.1.1 Das OSI Modell

Die *International Organisation for Standardisation* (ISO) entwarf 1977 das Open Systems Interconnection (OSI) Modell [2] als Referenz Modell für die zukünftige Entwicklung von Kommunikationsstandards. Das OSI Modell ist, wie in der nachfolgenden Abbildung zu sehen, in mehrere übereinander liegende Schichten aufgeteilt.

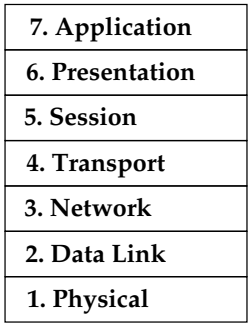

Abbildung 2.1: Das OSI Schichten Modell

Jede der Schichten übernimmt dabei eine definierte Aufgabe und besitzt eine Schnittstelle zu der jeweilig darüber- beziehungsweise darunterliegenden Schicht. Die wesentlichen Eigenschaften

des Schichten basierten Aufbaus ist, daß einzelne Schichten nicht übersprungen werden können, da jede Schicht die Funktionalität der darunter liegenden Schicht mit einschließt (und somit kapselt) und das jeweils zwei gleiche Schichten auf verschiedenen System miteinander interagieren. Die Aufgaben der einzelnen Schichten (Layer) werden nachfolgend beschrieben.

Physical Layer bietet den Zugang zum Übertragungsmedium (zum Kabel) und ist für die Übertragung der einzelnen Bits zuständig.

Data Link Layer ist für das Übertragungsformat, die Adressierung und die Sicherung (Vermeidung von Fehlern) der Daten auf dem Physikalischen Medium zuständig.

Network Layer ist für die Vermittlung einer Ende-zu-Ende Verbindung zwischen zwei System, auch über mehrere Netze hinweg, zuständig.

Transport Layer zuständig für die garantierte Übertragung von Daten

Session Layer zuständig für den Aufbau, die Verwaltung und den Abbau von Verbindungen zwischen Applikationen

Presentation Layer ist für die Darstellung der Daten zuständig

Application Layer beinhaltet eine Reihe von Anwendungsprotokollen.

2.1.2 Die TCP/IP Protokollfamilie

Die TCP/IP Protokollfamilie entstand bereits vor dem OSI Modell und basiert somit nicht auf ihm. Die TCP/IP Protokollfamilie wurde Mitte der 70er Jahre als Nachfolger des Network Control Protocol (NCP) entwickelt und wurde 1980 zum Standard des United States Department of Defense erhoben. Die eigentliche offizielle Spezifikation von TCP/IP (in der Version 4) erfolgte aber erst 1981. Seit dieser Zeit ist die TCP/IP Protokollfamilie der weltweite quasi Standard für die Vernetzung von Computersystemen. Die TCP/IP Protokollfamilie wird auch als DoD (siehe oben: Department of Defense) Architektur bezeichnet. Die DoD Architektur ist ebenfalls wie die OSI Architektur in Schichten aufgeteilt. Die folgende Abbildung zeigt eine Gegenüberstellung der beiden Architekturen.

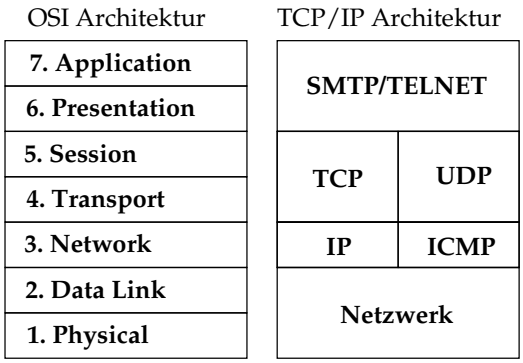

Abbildung 2.2: Vergleich OSI und TCP/IP

Die Abbildung zeigt eine Übereinstimmung der beiden Modelle, unter der Berücksichtigung das mehrere Schichten des OSI Modells in einer Schicht der TCP/IP Architektur abgebildet werden. Die Unterteilung der Architektur in mehrere Schichten ist für die weitere Betrachtung sehr wichtig.

Das Internet Protokoll

Das Internet Protokoll (IP) [12] bildet die Netzwerk Schicht der TCP/IP Architektur. Es ist wie auch die Netzwerk Schicht des OSI Modells (Layer 3) für die logische Adressierung und den Versand von Daten zuständig. IP implementiert zwei wichtige Grundfunktionen: die Adressierung und die Fragmentierung von Daten Paketen. Einzelne Daten Pakete werden im IP Umfeld als IP-Datagramme beziehungsweise als IP-Fragmente (wenn es sich um ein Fragment eines Datagramms handelt) bezeichnet. IP-Fragmentierung wird benötigt wenn ein IP-Datagramm die maximal zulässige Größe eines Paketes des darunter liegenden Netzwerkes übersteigt. Bei der Fragmentierung wird das Datagramm in mehrere Stücke geteilt; die einzelnen Fragmente werden versendet und vom Empfänger vor der Verarbeitung wieder zusammengesetzt. Zur Erfüllung der genannten Grundfunktionen wird jedem IP-Paket (IP-Paket bezeichnet ein IP-Datagramm oder ein IP-Fragment) ein Header vorangestellt. Dieses IP-Header enthält alle nötigen Angaben wie Sende- und Empfangs-Adressen. Der Aufbau des IP-Headers beziehungsweise der eines IP-Paketes ist in der folgende Abbildung dargestellt.

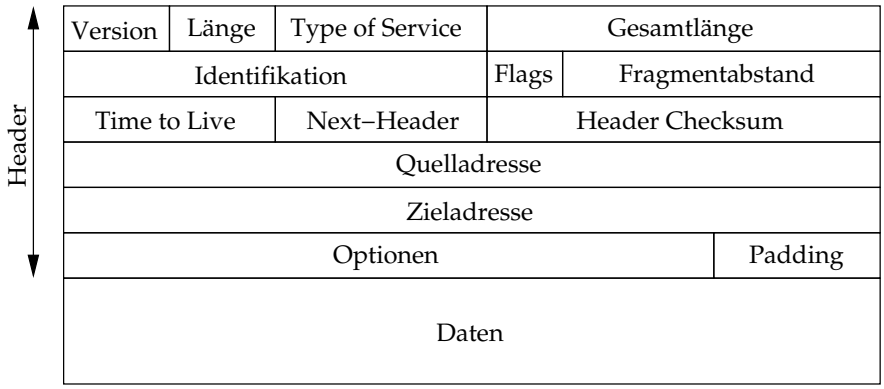

Abbildung 2.3: Aufbau eines IP-Datagramms

Die einzelnen Felder des IP-Headers sollen hier zum besseren Verständnis kurz erläutert werden, die Erläuterungen sind allerdings keines Falls vollständig beziehungsweise umfassend. Detaillierte Informationen können [12] entnommen werden.

Version Enthält die IP-Protokoll-Version.

Header-Länge Enthält die Länge des IP-Headers.

Type of Service Enthält Angaben über eine eventuell bevorzugte Behandlung des IP-Pakets.

Gesamtlänge Gibt die Gesamtlänge des IP-Pakets in Bytes an. Anhand der Größe des Feldes ergibt sich die maximal Größe von 65535 Bytes für ein IP-Datagramm.

Identifikation Bei Fragmentierung eines IP-Datagramms werden alle Teilfragmente anhand dieses Feldes identifiziert, alle Teilfragmente eines Datagramms haben die gleiche Identifikation. Komplette Datagramme haben eine Identifikation von Null (0).

Flags Die Flags machen Angaben über Fragmentierung beziehungsweise nicht Fragmentierung von Datagrammen.

Fragment Offset Dieser Wert gibt den Versatz an, an der das aktuelle Fragment innerhalb des zusammengesetzten Datagramms eingefügt wird.

Time to Live Dieses Feld gibt die Anzahl der Router an, durch die das IP-Paket passieren kann bevor es verworfen wird. Dabei verringert jeder Router durch den das IP-Paket passiert diesen Wert um eins (1). Bei Erreichung des Wertes Null (0) wird das Paket verworfen.

Next-Header Angabe über die Art der transportierten Nutzdaten beziehungsweise der Identifier des höher liegenden Protokolles.

Header-Prüfsumme Prüfsumme über das IP-Header.

Quell-IP-Adresse Die IP-Adresse des Senders des Pakets.

Ziel-IP-Adresse Die IP-Adresse des Empfängers des Pakets.

Optionen Dieses Feld enthält optionale Informationen und muß nicht vorhanden sein.

IP hat einige signifikante Eigenschaften, auf diese soll hier weiter eingegangen werden. IP ist Verbindungslos das heißt alle Datagramme werden völlig unabhängig voneinander übertragen, aus der Sicht von IP besteht kein Zusammenhang zwischen mehreren Datagrammen. Weiterhin ist IP unzuverlässig das bedeutet: Datagramme werden einfach gesendet egal ob der Empfänger existiert oder nicht, Datagramme können verworfen werden oder doppelt ankommen und in einer Sequenz gesendete Datagramme können in einer völlig anderen Reihenfolge beim Empfänger ankommen. Außerdem ist nicht garantiert, daß die Daten so ankommen wie sie gesendet wurden, da keine Fehlerkontrolle mit Ausnahme des IP-Headers stattfindet.

Das Internet Control Message Protocol

Das Internet Control Message Protocol (ICMP) [13] ist Teil der Netzwerk Schicht (daher wird ICMP meist innerhalb von IP dargestellt, siehe Abbildung 2.2). da es unter anderem zur Fehlersignalisierung dient. Im wesentlichen ist ICMP aber nur ein höheres Protokoll welches auf IP basiert. ICMP gleicht unter anderem einige Schwächen von IP aus, wie zum Beispiel die Signalisierung über die nicht Erreichbarkeit eines Hosts. ICMP Nachrichten können aufgrund von Fehler in verschiedenen Schichten (namentlich genannt sind: IP, TCP und UDP) generiert werden. Es gibt noch einige Ausnahmen welcher aber für das Grund-Verständnis nicht weiter relevant sind.

Das User Datagramm Protocol

Das User Datagramm Protocol (UDP) [11], ist ein verbindungsloses, unzuverlässiges Transport-Protokoll welches auf IP basiert. Es verfügt lediglich über zwei zusätzliche Eigenschaften gegenüber von IP. Erstere sind die sogenannten Port Informationen, durch diese können verschiedene Applikationen auf einem Host angesprochen werden. Weiterhin gibt es die Möglichkeit eine Prüfsumme über die Nutzdaten zu berechnen um mögliche Übertragungsfehler zu erkennen.

Das Transmission Control Protocol

Das Transmission Control Protocol (TCP) [14], stellt ein alternatives Transport-Protokoll dar. TCP ist im Gegensatz zu UDP ein verbindungsorientirtes Protokoll welches die Übertragen von Daten garantiert, hierzu besitzt TCP Mechanismen um Fehler, doppelt oder gar nicht übertragene Daten zu erkennen und entsprechend darauf zu reagieren. TCP besitzt genau wie UDP Port Informationen und kann somit mehrere Verbindungen zwischen zwei Hosts unterscheiden.

KAPITEL

DREI

Die IPSec Architektur

IPSec - Die *IP Security Architecture* wurde entwickelt um einen einheitlichen Standard für die kryptographische Sicherung von IP-Netzwerken (siehe 2.3) zu schaffen. Eine erste Version der IPSec Architektur wurde bereits 1995 vorgestellt diese hatte allerdings noch einige Schwächen. Seit 1998 existiert eine überarbeitete Version welche den aktuellen Stand repräsentiert.

Die IPSec Architektur [6] ist ein großes komplexes System bestehend aus mehreren einzel Teilen und ist in über einem Dutzend RFCs spezifiziert. Dieses Kapitel beschreibt die Funktionsweise und das Zusammenwirken der einzelnen Teile der IPSec Architektur.

3.1 Übersicht

IPSec setzt, wie der Name schon sagt, auf IP-Ebene (der Netzwerkebenen; Schicht 3 des OSI Modells, siehe 2.1.1) an. Hierdurch kann IPSec den gesamten IP Netzwerk-Verkehr transparent schützen. In diesem Zusammenhang bedeutet Transparent soviel wie, daß keine Veränderung an anderen Protokoll-Schichten oder an Applikationen nötig sind.

Es gibt zwei Varianten von IPSec Implementierungen:

Host In der Host-Variante ist IPSec direkt auf den jeweiligen End-Systemen vorhanden. Sie ermöglicht eine direkte (so genannte) Ende-zu-Ende Sicherung des Netzwerk-Verkehrs.

Gateway In der Gateway-Variante ist IPSec nur auf einem Zwischen-System (z.B. einem Router) vorhanden. Dieses Zwischen-System wird dazu verwendet den Netzwerk-Verkehr von nicht IPSec fähigen End-System zu schützen. Weiterhin ermöglicht diese eine gesicherte Kommunikation zwischen IPSec fähigen und nicht IPSec End-Systemen. In der Gateway-Variante geht allerdings die Ende-zu-Ende Sicherung verloren.

3.1.1 Dienste

IPSec bietet eine Reihe von Sicherungs-Diensten:

Zugriffskontrolle Durch die Zugriffskontrolle kann bestimmt werden ob und wie Netzwerkverbindungen genutzt werden können, hiermit kann zum Beispiel festgelegt werden, daß ein System nur mit bestimmten anderen Systemen kommunizieren darf beziehungsweise wie diese Kommunikation aussieht (z.B. ob Verschlüsselung eingesetzt wird oder nicht).

Schutz der Integrität von Daten Hierdurch kann gewährleistet werden, daß die Daten korrekt (d.h. unverändert) sind beziehungsweise daß sie wirklich von einer autorisierten Quelle stammen.

Schutz vor Wiedereinspielung Die Wiedereinspielung von abgefangenen Netzwerk-Paketen (Replay-Attack) ist eine bekannte Möglichkeit schlecht gesicherte Protokolle zu überlisten. Dieses kann durch IPSec verhindert werden.

Daten-Vertraulichkeit Durch Verschlüsselung der Daten kann deren Vertraulichkeit gewährleistet werden.

Vertrauchlichkeit des Datenflußverhaltens Verhinderung der Datenflußanalyse, zum Beispiel durch Verschleierung der Quelle und der Größe eines IP-Paketes.

3.1.2 Die Protokolle

Die von IPSec gebotenen Dienste werden durch den Einsatz von zwei Netzwerk-Sicherungs-Protokollen, von Prozeduren zur Verwaltung von kryptographischen Schlüsseln und durch diverse Management-Konstrukte, erbracht. Die beiden Netzwerk-Sicherung-Protokolle:

Authentication Header (AH) für Daten Authenzität

und

Encapsulated Security Payload (ESP) für Daten Verschlüsselung

decken dabei jeweils einen Teil der Dienste mit Überlappungen ab. Die Protokolle können dabei auf zwei Arten eingesetzt werden: im *Transport Mode* und im *Tunnel Mode*.

Der *Transport Mode* ist der preferierte Modus für eine Verbindung zweier *Hosts* (siehe 3.1). Hierbei wird das zusätzliche Header (AH oder ESP), wie in der Abbildung 3.1 dargestellt, direkt in das zu schützende Datagramm eingefügt.

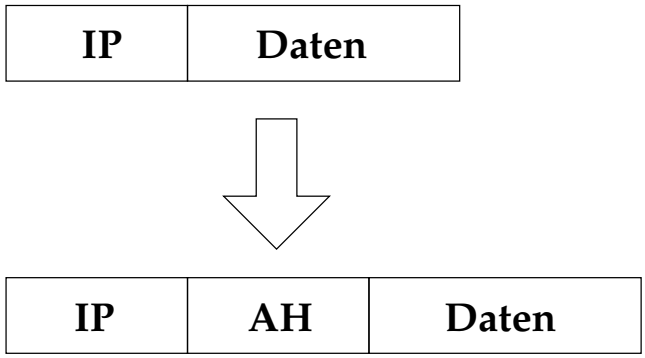

Abbildung 3.1: AH im Transport Mode

Der *Tunnel Mode* wird eingesetzt sobald einer der Kommunikationspartner ein *Gateway* System ist. Da hierbei der End-Punkt der *Security Association* (siehe nächster Abschnitt) nicht der End-Punkt der eignedlichen „Verbindung“ ist. Im *Tunnel Mode* wird das zu schützende Datagramm vor der IPSec Behandlung in ein neues Datagramm eingekapselt. Dieses neue Datagramm hat dann als IP-Zieladresse die Adresse des *Gateway* Systems. Das so neu entstandene Datagramm wird dann wie ein *Transport Mode* Datagramm behandelt.

Die Tunnelung (Einkapslung in ein neues Datagramm) ist notwendig da es mehrere Wege zwischen einem *Gateway* und einem *Host* beziehungsweise zwischen zwei *Gateways* gegeben kann

und gewährleistet sein muß, daß die geschützen Datagramme in jedem Fall das *Gateway* passieren um so das zusätzliche Header, vor dem Transport des Datagramms an das nicht IPSec fähige System, zu entfernen. Prinzipiell können auch zwei *Hosts* untereinander über einen Tunnel kommunizieren, dies bringt allerdings keine zusätzliche Sicherheit. Den Aufbau eines getunnelten Datagramms zeigt die folgende Abbildung.

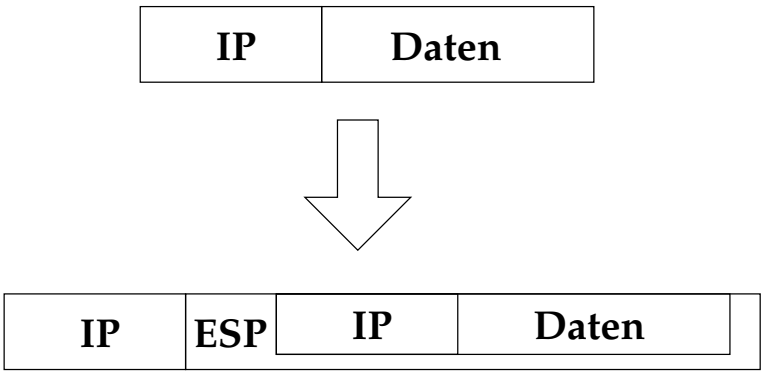

Abbildung 3.2: ESP im Tunnel Mode

Die Management-Konstrukte werden für die interne Verwaltung beziehungsweise für die Steuerung des IPSec System benutzt. Das wichtigste Konstrukt ist die *Security Association*.

3.1.3 Die Security Association

Die *Security Association* (SA) stellt den Kern der IPSec Architektur da. Sie repräsentiert eine gesicherte, unidirektionale Punkt-zu-Punkt Verbindung zwischen zwei IPSec System, das heißt eine SA gibt das Protokoll, dessen Transportart (siehe 3.1.2) und die zu verwendenden kryptographischen Algorithmen an, durch welches eine „Verbindung“ gesichert wird.

Jede SA spezifiziert genau ein Protokoll, das bedeutet: sollen mehrere Protokolle, parallel, eingesetzt werden, müssen für jedes weitere Protokoll separate SAs erstellt werden (diese Kombination von mehreren SAs wird als *Security Association Bundle* [Bündel] bezeichnet). Jede SA wird eindeutig durch ein Tripel identifiziert:

IP-Zieladresse Die IP-Adresse des anderen Endes der SA

IPSec Protokoll Das verwendete IPSec Protokoll

Security Parameter Index Ein Wert wodurch Verbindungen, mit der gleichen IP-Zieladresse und dem gleichen IPSec Protokoll, unterschieden werden können. Dieser Wert ist in beiden IPSec Headern (AH und ESP) vorhanden und wird durch den „Empfänger“ festgelegt (siehe 3.4).

Da SAs unidirektional sind, werden immer zwei SAs für eine „Verbindung“ benötigt, eine zum *Senden* und eine zum *Empfangen*. Soll Beispielsweise die Kommunikation der beiden Hosts *A* und *B* durch ESP gesichert werden, muß jeder der beiden Hosts zwei SAs (jeweils eine zum senden und eine zum empfangen) besitzen. Diese SAs müssen jeweils mit dem gleichen Algorithmus und den gleichen Schlüsseln konfiguriert sein. Wenn diese Verbindung durch AH und ESP gesichert werden soll, bräuchte man auf jeder Seite vier SAs.

Die Verwaltung des Netzwerk-Verkehrs unter der Verwendung von IPSec wird im wesentlichen durch eine Datenbank, der *Security Policy Database*, gesteuert. Sie gibt unter anderem an, welche Art von SAs für eine bestimmte Verbindung benutzt werden muß.

3.1.4 Die Security Policy Database

Die *Security Policy Database* (SPD) spezifiziert wie Netzwerk-Verkehr behandelt wird beziehungsweise welche Dienste für IP-Datagramme angeboten werden. Die SPD ist sowohl für ein- als auch für ausgehenden Netzwerk-Verkehr zuständig, das heißt sie regelt den gesamten Netzwerk-Verkehr des Systems und unterscheidet drei mögliche Behandlungsarten für Datagramme.

apply Datagramm durch IPSec sichern

bypass Datagramm unverändert passieren lassen

discard Datagramm verwerfen

Die Zuordnung von IP-Datagrammen zu einer *Security Policy* geschieht durch sogenannte Selektoren. Selektoren beschreiben Charakteristika von IP-Datagrammen. Charakteristika sind zumeist Werte von Feldern des IP-Headers beziehungsweise von Feldern andere Protokolle (z.B. TCP oder UDP). Einzelne Selektoren können durch die Besetzung mit Wildcards (Platzhaltern) ignoriert werden.

IP-Zieladresse Selektor welcher sich auf die IP-Zieladresse des Datagramms bezieht

IP-Quelldresse Selektor welcher sich auf die IP-Quelladresse des Datagramms bezieht

Transport Protokoll Selektor welcher sich auf den Typ des von IP transportierten höheren Protokolles bezieht

Ziel-Port Selektor welcher sich auf den Ziel-Port eines TCP oder UDP Datagramms bezieht

Quell-Port Selektor welcher sich auf den Quell-Port eines TCP oder UDP Datagramms bezieht

Es gibt zwei weitere Selektoren, diese beziehen sich allerdings nicht auf Informationen aus den Headern.

Name Selektor welcher sich auf den Benutzer bezieht welcher das Datagramm erzeugt hat

Data Sensity Level Selektor welcher sich auf IPSO beziehungsweise CIPSO Labels bezieht. Diese Art von Selektoren muß nur von Systemen unterstützt werden auf denen *Information Flow Security* Implementiert ist.

3.1.5 Die Security Association Database

Die *Security Association Database* (SAD) enthält die Zustands-Informationen aller aktiven SAs. Das heißt, in einem Eintrag der SAD werden alle Parameter einer aktiven SA gespeichert. Diese Parameter sind die Informationen welche für die Verarbeitung eines IP-Datagramms durch die entsprechende SA benötigt werden. Die Wichtigsten Parameter sind:

Der Security Parameter Index Zur Indetifikation der SA (siehe 3.1.3)

Das IPSec Protokoll AH oder ESP

Der Protokoll Modus Transport oder Tunnel

Die IP-Adressen Die Quell und Ziel IP-Adresse und im *Tunnel Mode* die IP-Adressen der beiden Tunnel-Enden

Die aktuelle Sequenz-Nummer Für Anti-Replay Schutz (Schutz vor Wiedereinspielung)

Anti-Replay Window Status des Anti-Replay Schutzmechanismus

Lifetime Angaben zur Lebensdauer der SA

Selektoren Die Selektoren der zu der SA gehörigen *Security Policy*

Protokoll spezifische Parameter Bei AH kommen noch Parameter wie: Algorithmentyp, Schlüssel-Länge und Schlüssel dazu. Bei ESP kommen für Verschlüsselung: der Algorithementyp, die Schlüssel-Länge, der Schlüssel und die eventuell benötigten Synchronisations Daten (siehe 3.3.2) dazu; für Authentizität werden noch einmal zusätzlich: der Algorithementyp, die Schlüssel-Länge und der Schlüssel gespeichert.

3.2 IPSec Verarbeitung

Die IPSec Verarbeitung ist für ein- und ausgehenden Netzwerk-Verkehr grundsätzlich verschieden.

3.2.1 Ausgehend

Im ersten Schritt wird die *Security Policy Database* befragt um herauszufinden welcher der drei Behandlungsarten (siehe 3.1.4) für das zu sendende IP-Datagramm selektiert ist. Wird keine passende *Security Policy* gefunden, wird das Datagramm verworfen, dies entspricht dem Grundgedanken von *Secure on Default* (keine ungewollte Kommunikation, z.B. durch vergessende Konfiguration).

Wenn die gefundene *Security Policy* besagt, daß das Datagramm verworfen oder durchgelassen werden soll, wird dies entsprechend ausgeführt.

Soll IPSec verwendet werden, wird die *Security Association Database* nach der passenden SA (beziehungsweise SA Bündel) durchsucht. Wird keine SA gefunden, wird mit Hilfe des Schlüsselverwaltungssystems eine neue SA anhand der Vorgaben der *Security Policy* erstellt. Ist die gefundene SA „abgelaufen“ wird ebenfalls probiert diese mittels des Schlüsselverwaltungssystems neu zu erstellen. Wenn keine neue SA erstellt werden konnte wird das Datagramm verworfen.

Zum Schluß wird das Datagramm anhand der Vorgaben der SA (bei mehreren SAs, in der entsprechenden Reihenfolge) verarbeitet. Die Verarbeitung der Entsprechenden Protokolle wird im nächsten Abschnitt beschrieben. Nach der erfolgreiche Verarbeitung wird das Datagramm versendet (hierbei sind eventuell weitere Schritte wie z.B. Fragmentierung notwendig, diese werden allerdings durch die IP-Schicht vorgenommen).

3.2.2 Eingehend

Der erste Schritt für eingehende IP-Datagramme besteht darin zu prüfen ob es sich bei dem Datagramm um ein IPSec-Datagramm handelt, dies ist der Fall wenn das Next-Header Feld (siehe 2.3) des Datagramms das vorhanden sein eines der beiden IPSec Protokolle (AH oder ESP) angibt.

Ist dies nicht der Fall, wird die *Security Policy Database* befragt. Wenn diese angibt, daß das Datagramm durchgelassen oder verworfen werden soll wird dies entsprechend ausgeführt. Wenn keine *Security Policy* gefunden wird, wird das Datagramm ebenfalls verworfen (siehe 3.2.1).

Ist eines der beiden IPSec Header vorhanden; wird anhand der IP-Zieladresse, dem *Security Parameter Index* und dem IPSec Protokoll der passende Eintrag in der *Security Association Database*

gesucht. Wenn kein passender Eintrag gefunden wird, wird das Datagramm verworfen. Ansonsten wird das Datagramm anhand des gefundenen Eintrags verarbeitet.

Nach der erfolgreichen Verarbeitung, wird erneut geprüft ob sich eventuell weitere IPSec-Header in dem Datagramm befinden. Ist dies der Fall wird der Vorgang solange wiederholt bis keine IPSec-Header mehr vorhanden sind, oder bis ein IPSec-Header für ein anderes System gefunden wurde (nur bei *Gateways*, siehe 3.1).

Schlägt die Verarbeitung fehl, wird das Datagramm verworfen.

Zum Schluß wird das entpackte Datagramm gegen die *Security Policy Database* geprüft und die darin spezifizieren SAs mit den tatsächlich verwendeten verglichen, stimmen diese nicht überein wird das Datagramm verworfen, ansonsten wird das Datagramm an die IP-Schicht weiter gereicht.

3.3 Protokoll Verarbeitung

Die Protokoll Verarbeitung ist grob drei geteilt in AH, ESP und einen gemeinsame Teil. Der gemeinsame Teil wird zum Schluß erklärt.

3.3.1 Authentication Header

Der *Authentication Header* (AH) [6], siehe Abbildung 3.3, wird direkt nach dem IP-Header (und vor den Daten des nächst höheren Protokolls) eingefügt. Im *Tunnel Mode* wird dem ursprüngliche Datagramm ein zusätzliches IP-Header vorangestellt, wobei das *Authentication Header* dann zwischen den beiden IP-Headern liegt.

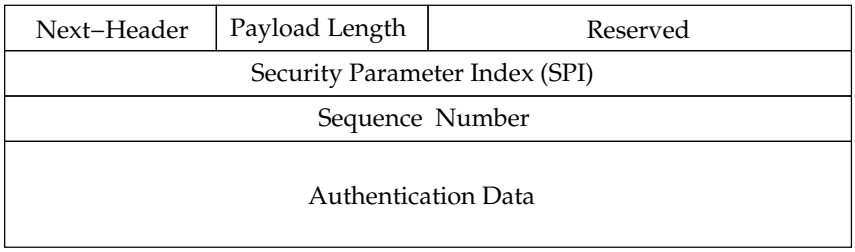

Abbildung 3.3: Das Authentication Header

Das Next-Header Feld hat die gleiche Funktion wie das entsprechende Feld des IP-Headers, es gibt die Art des nächsten Protokolls an. Beim Einfügen des *Authentication Headers* in das Datagramm, bekommt dieses den Wert des Next-Header Feldes des IP-Headers welches wiederum mit dem Wert für AH belegt wird. Im *Tunnel Mode* wird dieses Feld mit dem Wert für IP belegt.

Im Payload-Length Feld wird die Länge des AH angegeben, dies ist notwendig da die Länge der Authentifizierungsdaten variable (abhänging von dem verwendeten Algorithmus) ist. Die Länge wird in 32 Bit-Einheiten angegeben, wobei die ersten 8 Bytes nicht mit gerechnet werden.

Das Reserved Feld wird nicht verwendet und muß auf Null (0) gesetzt werden. Der *Security Parameter Index* wird wie in 3.2.2 beschrieben, vom Empfänger dazu verwendet den entsprechenden Eintrag in der SAD zu finden, der Sender muß daher dieses Feld entsprechend besetzen.

Das Sequenz-Nummern-Feld beinhaltet die von Sender für jedes Datagramm erhöhte Sequenz-Nummer, sie wird für den Anti-Replay Schutz verwendet (siehe 3.3.3). Wird der Anti-Replay Schutz nicht verwendet, ignoriert der Empfänger dieses Feld einfach.

Das Authentication Data Feld enthält den *Integrity Check Value* (ICV), eine kryptographische Signatur. Diese Signatur wird über das komplette Datagramm berechnet und bestätigt so dessen Authenzität. Die genaue Vorgehensweise, um die Signatur zu berechnen, ist vom verwendeten Verfahren abhängig. Meistens werden hierfür kryptographische Einweg-Hash-Funktionen zusammen mit dem HMAC [7] Verfahren verwendet. Das HMAC Verfahren dient dazu eine „Checksumme“ an einen Schlüssel zu binden, um so eine einfache neu Berechnung zu verhindern.

Durch die Spezifikation sind zwei Standartverfahren zur Berechnung der ICV festgelegt:

- HMAC-MD5-96, beschrieben in [8]
- HMAC-SHA-196, beschrieben in [9]

Zur Berechnung der Signatur müssen alle Felder des IP-Headers (inklusive der Optionen) und die des AH deren Wert sich während des Transports verändern könnten beziehungsweise deren Wert bei der Berechnung nicht bekannt ist (z.B. das Authentication Data Feld) auf Null (0) gesetzt werden. Felder deren Werte vorhersagbar sind, müssen für die Signaturberechung auf diese Werte gesetzt werden. Diese „Vorbelegung“ muß sowohl vom Sender als auch Empfänger vor der Berechnung geschehen.

3.3.2 Encapsulated Security Payload

Bei *Encapsulated Security Payload* (ESP) [5] werden, wie der Name schon sagt, die transportierten Daten eingekapselt. Im *Transport Mode* sind dies die Daten direkt nach dem IP-Header und im *Tunnel Mode* ist dies das komplette ursprüngliche IP-Datagramm. Wie in der Abbildung 3.4 zu sehen ist, besitzt auch ESP ein Feld für Authenzitätsdaten, die Signatur wird hierbei allerdings nur über den ESP Teil des Datagramms berechnet und ist somit „schwächer“.

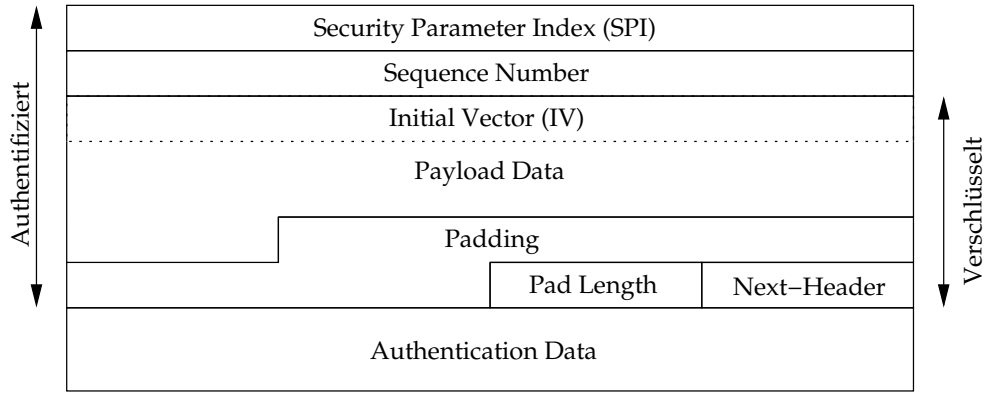

Abbildung 3.4: Die Encapsulated Security Payload

Wie in der Abbildung 3.4 zu sehen ist, werden die transportierten Daten (die Payload), das Padding, die Pad Length und das Next-Header Feld verschlüsselt. Das Padding Feld wird zum Auffüllen der Payload verwendet, die Auffüllung ist dabei aus mehreren Gründen notwendig:

einerseits wird für die Verschlüsselung meistens eine Block-Chiffre eingesetzt, welche ein Vielfaches seiner Blockgröße verlangt. Weiterhin muß gewährleistet sein, daß die beiden Felder (Pad Length und Next-Header) wie in der Abbildung 3.4 angeordnet sind (am äußeren Rand eines 32-Bit Blocks). Der Inhalt des Padding Feldes richtet sich dabei nach den Vorgaben des Verschlüsselungsalgorithmus.

Je nach Verschlüsselungsalgorithmus enthält die Payload noch einen Initial Vector (IV), dessen Größe und Position innerhalb der Payload ist allerdings abhängig vom Algorithmus.

Der *Secuirty Parameter Index*, die Sequenz-Nummer und das Next-Header Feld werden genau so wie beim *Authentication Header* behandelt.

Die Verschlüsselung und die Authenzität können bei ESP zusammen oder getrennt benutzt werden.

3.3.3 Allgemeine Verarbeitungsschritte

Bei Versenden wird zuerst das neue Datagramm zusammengebaut, das heißt die einzelnen Felder der Header werden initialisiert beziehungsweise neu besetzt. Dieser Schritt beinhaltet auch das hochzählen der Sequenz-Nummer vor dem einfügen der selbigen. Wurde die SA automatisch erstellt darf die Sequenz-Nummer nicht überlaufen und muß vor dem überlauf durch eine neue ersetzt werden (siehe 3.4). Nach der Berechnung der Signaturen werden diese entsprechend eingefügt. Bei ESP muß immer zuerst verschlüsselt und dann Signiert werden.

Beim Empfangen wird bei aktivem Anti-Replay Schutz zuerst die Sequenz-Nummer geprüft, zu diesem Zwecke wird die in dem jeweiligen Header enthaltene Sequenz-Nummer gegen die Sequenz-Nummer aus dem dazugehörigen *Security Association Database* Eintrag geprüft. Ansonsten wird zuerst die Authentizität geprüft und dann entschlüsselt. Nach der erfolgreichen Verarbeitung werden die entsprechenden Header entfernt.

Eingehende Datagramme können aus mehreren Gründen verworfen werden:

- wenn sie als Replay erkannt werden
- falscher Authentzitäts Prüfwert (ICV)
- ungültige Länge des Datagramms, nach Entschlüsselung (falscher Schlüssel)

Anti-Replay Schutz

Der Anti-Replay Schutz funktioniert nach einem einfachen Prinzip. Beim Einrichten der SAs wird die Sequenz-Nummer beim Sender und Empfänger auf Null (0) gestezt. Zudem hält der Empfänger ein Offset-Fenster, dessen oberes Ende an der letzten höchst empfangenen und akzeptierten Sequenz-Nummer definiert ist.

Der Sender erhöht nun die Sequenz-Nummer für jedes Datagramm um eins (1) und der Empfänger prüft jede neu empfangene Sequenz-Nummer ob sie höher, als die im Offset-Fenster höchst vermerkte Sequenz-Nummer, ist. Ist dies der Fall wird die empfangene Sequenz-Nummer, nach erfolgreicher Authentizitätprüfung als neue höchste Sequenz-Nummer vermerkt und das Offset-Fenster verschiebt sich entsprechend nach oben. Ist die empfangene Sequenz-Nummer kleiner als die höchste, im Offset-Fenster, vermerkte Nummer, wird geprüft ob sie schon im Offset-Fenster vorhanden ist, oder ausserhalb der untersten Grenze des Offset-Fensters liegt. Tritt einer dieser Fälle zu, wird das Datagramm verworfen. Wenn dies nicht zutrifft wird nach erfolgreicher Authentizitätsprüfung die Sequenz-Nummer in dem Offset-Fenster vermerkt.

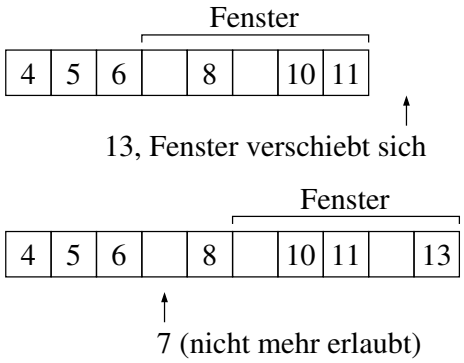

Abbildung 3.5: Anti-Replay Fenster

Die Abbildung zeigt ein Offset-Fenster der Größe fünf (5). Das obere Fenster würde die Sequenz-Nummern 7, 9 und alle Nummern größer als 11 akzeptieren. Beim Eintreffen der Sequenz-Nummer 13, wird diese akzeptiert und das Fenster entsprechend verschoben, jetzt werden nur noch 9, 12 und größer 13 akzeptiert. Wenn jetzt die Sequenz-Nummer 7 eintrifft wird diese als *Replay* erkannt und abgewiesen.

Implementierungen müssen eine minimale Fenstergröße von 32 Datagrammen unterstützen, aber im Normalfall ist eine Größe von 64 Datagrammen wohl am sinnvollsten.

3.4 Schlüssel-Verwaltung

Die IPSec Architektur basiert auf Kryptographie, was bedeutet das kryptographische Schlüssel erzeugt und verteilt werden müssen.

Generell stehen hierfür zwei Möglichkeiten zur Verfügung:

- Manuelle Schlüssellverwaltung
- Automatische Schlüsselverwaltung

3.4.1 Manuelle Verwaltung

Bei der manuellen Schlüsselverwaltung richtet ein Benutzer oder Administrator die entsprechenden *Security Policys* mit den dazugehörigen SAs ein. Hierbei wird unter anderem der *Security Parameter Index* (SPI) durch den Benutzer manuell vergeben. Der Benutzer muß folgende Paramter festlegen:

- *Security Policys*, Selektoren, etc.
- SAs
  - SPI
  - Protokoll
  - Transportart
  - Algorithmen
  - Schlüssel

- Kombination der SAs

Für manuell eingerichtete „Verbindungen“ steht kein Anti-Replay Schutz (siehe 3.3.3) zur Verfügung. Das heißt der Empfänger ignoriert die Sequenz-Nummer und der Sender „läßt“ das überlaufen der Sequenz-Nummer zu.

3.4.2 Automatische Verwaltung

Bei automatisch Verwalteten SAs, werden die *Security Policys* mit den dazugehörigen SA-Gerüsten vom Benutzer eingerichtet.

- *Security Policys*
- SA-Gerüste
  - Protokoll
  - Transportart
  - Algorithmen
  - Lebens-Zeit
- Kombination der SAs

Automatische SAs haben anders als manuelle SAs nur ein begrenzte Lebenszeit. Die Lebenszeit kann dabei in Sekunden oder in Bytes (Anzahl der durch die SA übertragene Bytes) angegeben werden. Ist die Lebenzeit abgelaufen, muß die SA neu erstellt werden.

Schlüsselaushandlung ist eine komplexe Angelegenheit, hierbei spielen viele Faktoren eine Rolle. Schlüsselaushandlung besteht meistens aus zwei Schritten, als erstes verifizieren die beiden Partner sich gegenseitig und tauschen nach erfolgreicher Verifizierung dann ihre Schlüssel.

Die Schlüsselaushandlung wird meistens in Form einer Applikation (ausserhalb des Betriebssystem-Kerns) implementiert.

Das Standart Verfahren für IPSec ist IKE [4]. Auf IKE oder andere Schlüsselaushandlungs-Protokolle wird hier nicht weiter eingegangen.

KAPITEL

VIER

Analyse

Nach der Betrachtung der IPSec Architektur an sich, wird nun kurz auf das Problem ICMP und IPSec eingegangen. Weiter wird kurz erläutert welche Teile der Architektur näher betrachtet werden müssen. Zum Schluß werden noch die verschiedenen Implementierungsmöglichkeiten erläutert.

4.1 IPSec und ICMP

Bei IPSec gibt es im Zusammenhang mit ICMP-Datagrammen einige Probleme beziehungsweise Besonderheiten zu beachten.

4.1.1 ICMP Nachrichtenarten

Generell sind ICMP-Nachrichten in zwei Kategorien eingeteilt: *Anfragen* und *Fehler*

Da *Anfragen* Ende-zu-Ende Nachrichten sind, können diese direkt (wie jede andere Art von IP-Datagramm) über die *Security Policy Database* verwaltet werden und benötigen daher keiner speziellen Behandlung.

Ende-zu-Ende *Fehler* benötigen ebenfalls keiner besonderen Betrachtung, da sie wie *Anfragen* behandelt werden können.

4.1.2 Router-zu-Ende Nachrichten

Die hier als Router-zu-Ende Nachrichten bezeichneten ICMP Fehler sind Nachrichten, die von Routern zu Hosts gesendet werden. Router sind Systeme welche Netzwerk-Verkehr zwischen einzelnen Enden (Hosts) dynamisch vermitteln. Das Problem bei solchen Nachrichten ist, daß es im Normalfall keine *Security Policys* für diese gibt, das liegt daran, daß eine „Verbindung“ nicht immer von dem gleichen Router „bedient“ wird. Hierdurch sind Router-zu-Ende Nachrichten generell „unsicher“.

Unsichere ICMP Nachrichten können zu diversen sicherheitskritischen Problemen führen, daher ist es logisch alle „unsicheren“ Nachrichten generell zu verwerfen. Die generelle Verwerfung dieser Nachrichten würde allerdings den normalen Netz-Betrieb stark beeinträchtigen. Daher müssen Nachrichten, welche wichtigste Funktionen erbringen, speziell behandelt werden. Die wohl wichtigste ist die *Path MTU Discovery*.

4.1.3 Path MTU Discovery

Die *Maximum Transmission Unit* (MTU) ist die größte Paket-Länge, die über ein physisches Netz verschickt werden kann (z.B. 1500 Bytes bei Ethernet). Die MTU eines Pfades (Paths), ist die kleinste MTU aller Strecken dieses Pfades. Die *Path MTU* ist somit die größte Paket-Länge bei den Datagrammen, die entlang dieses Pfades gesendet werden, nicht in Fragmente zerlegt werden müssen. Die MTU ist von Interesse, da Datenströme mit langen Paketen effizienter vermittelt werden können (da bei gleicher Datenrate die Paketrate niedriger ist als bei kurzen Paketen) und die Router nicht mit der Fragmentierung von Paketen belastet werden.

Durch die *Path MTU Discovery* [10] kann ein System diese Größe feststellen. Dazu setzt ein System für ausgehende IP-Datagramme das Don't Fragment (DF) Bit und erzwingt so eine Rückmeldung von Routern für welche dieses Datagramm zu groß ist. Diese Router verwerfen das Datagramm und senden eine ICMP Nachricht (Fragmentierung erforderlich, aber unmöglich). Diese ICMP Nachricht enthält auch die größte Paket-Länge.

Da IPSec diese Paket-Länge im negativen Sinn beeinflußt (durch einfügen von zusätzlichen Headern), muß sich eine IPSec Implementierung an der *Path MTU Discovery* „beteiligen". Diese Beteiligung besteht darin, die mitgeteilte Größe vor der Weiterleitung der Nachricht, entsprechend den benutzten IPSec Protokollen und Algorithmen, anzupassen, das heißt zu verkleinern.

4.2 Systemkomponenten

Vor dem Design ist es nötig festzulegen was getan werden muß. Hierfür wurde die IPSec Architektur in drei Komponente zerlegt.

- Protokoll-Treiber
- *Security Policy Database*
- *Security Assocation Database*

4.2.1 Der Protokoll-Treiber

Die IPSec Protokolle sind Teil der Netzwerk-Schicht. Je nach Implementierungsart kommunizieren sie direkt mit den Protokollen der Transport-Schicht oder mit anderen Protokollen der Netzwerk-Schicht.

Der Protokoll-Treiber muß eine Reihe von Aufgaben effizient ausführen, Implementierungsabhängig müssen eventuell Funktionen andere Schichten zusätzlich implementiert werden. Die Zusammenfassung der Aufgaben des Protokoll-Treibers lautet:

Header-Verarbeitung Prüfung und Manipulation von IP-Headern

IP-Tunnel Verarbeitung Ein- und Entkapselung von IP-Datagrammen

AH und ESP Verarbeitung Verarbeitung, Erstellung und Prüfung der AH und ESP Header

Versand von IP-Paketen Weiterleitung von IP-Paketen an höhere beziehungsweise niedrigere Schichten

ICMP-Verarbeitung Verarbeitung und Erzeugung von ICMP Fehler Nachrichten

IP-Fragmentierung Fragmentierung und Re-Assemblierung von IP-Datagrammen

4.2.2 Die Security Policy Database

Die *Security Policy Database* (SPD) regelt das gesamte Verhalten des IPSec Systems, das heißt jedes IP-Paket wird gegen sie geprüft. Hieraus ergibt sich das die SPD so angelegt werden muß, das ein Schneller Zugriff (Suchen/Lesen) auf sie möglich ist. Diese optimale (schnelle) Zugriff muß unter der Berücksichtigung der totalen Ordnung der einzelnen Einträge erfolgen.

4.2.3 Die Security Assocation Database

In der *Security Association Database* (SAD) werden alle aktiven *Security Associations* (SAs) gehalten. Auf die SAD wird, wie auch auf die SPD, für jedes IP-Paket zugegriffen. Daher muß auch auf die SAD effizient zugegriffen werden können. Die einzelnen Einträge müssen hierbei allerdings keiner Ordnung unterliegen, da sie jeweils über ein Tripel (siehe 3.1.3) identifiziert werden können.

4.3 Implementierungsmöglichkeiten

In [6] sind drei Möglichkeiten für eine IPSec Implementierung angegeben, diese werden hier kurz erläutert.

4.3.1 Integriert in den IP-Stacks des Betriebssystems

Die Integration in den IP-Stack des Betriebssystems ist die wohl offensichtlichste Implementierungsmöglichkeit. Sie bringt viele Vorteile mit sich. Die Effizienz einer IPSec Host Implementierung kann durch diese Implementierungsart entscheidend verbessert werden. Einige optionale Funktionen von IPSec Host Implementierungen lassen sich ausschließlich durch eine Integration in den IP-Stack des Betriebssystems bewerkstelligen.

4.3.2 Bump-in-the-Stack

Bei der *Bump-in-the-Stack* (BITS) Implementierung wird die IPSec Funktionalität als separates Modul entwickelt. Dieses Modul wird zwischen dem IP-Stack des Betriebssystems und dem Netzwerk-Adapter-Treiber plaziert. Hierbei wird es nötig einige Funktionen eines IP-Stacks zusätzlich zu implementieren, wie zum Beispiel die Re-Assemblierung von IP-Fragmenten zu IP-Datagrammen. Die *Bump-in-the-Stack* Implementierung wird meistens bei einer nachträglichen Implementierung eines IPSec Stacks in ein bereits vorhandenes Betriebssystem herangezogen, weil hierfür kein Zugriff auf die Quellen des Wirts-Betriebssystems notwendig ist. Diese Implementierungsart wird hauptsächlich bei IPSec Host Implementierungen verwendet.

4.3.3 Bump-in-the-Wire

Die *Bump-in-the-Wire* (BITW) Implementierung ist grundlegend verschieden zu den beiden erst genannten Implementierungsarten. Hierbei handelt es sich um eine vom Betriebssystem getrennte Implementierung, meistens als ein Stück externe Hardware. Diese externe Hardware verfügt normalerweise über eine eigene IP-Adresse. Eine *Bump-in-the-Wire* Implementierung kann als IPSec Host Implementierung fungieren und IPSec Dienste für genau ein System bieten. Hierbei ist das Verhalten ähnlich zu einer *Bump-in-the-Stack* Implementierung. Wenn IPSec Dienste für mehrere Systeme geleistet werden muß, muß sich die *Bump-in-the-Wire* Implementierung wie eine Gateway Implementierung verhalten. Diese Implementierungsart ist unabhängig von der Art der eingesetzten Betriebssysteme der Host System für die IPSec Dienste geleistet werden.

4.3.4 Vergleich

Im direkten Vergleich scheint es nur logisch den IPSec-Stack in den IP-Stack des Betriebssystems zu integrieren, da hierbei die Einbindung in die Socket API möglich ist. Durch die Einbindung in die Socket API können Benutzer[1] bezogene Selektoren (siehe 3.1.4) implementiert werden. Weiterhin bietet die Einbindung in die Socket API, wie im obigen Abschnitt bereits erwähnt, Effizienz Vorteile für Host Implementierungen. Eine direkte Implementierung ist allerdings vom Willen des Betriebssystem-Herstellers abhängig (sofern er nicht selber die IPSec Implementierung vornimmt), da hierfür der Quellcode des Betriebssystems beziehungsweise der des IP-Stacks und einiger Systembibliotheken vorliegen muß. Die *Bump-in-the-Stack* (BITS) Implementierung zielt auf eine nachträgliche IPSec Implementierung ab. Ein Teil der Funktionalität wie zum Beispiel Benutzer bezogene Selektoren können damit nur sehr schlecht und teilweise gar nicht implementiert werden. Sie eignet sich aber gut für eine Portable IPSec Implementierung, da die IPSec Funktionalität als ein externes Modul implementiert und getestet werden kann. Die Schnittstelle zu den Wirts-Betriebssystemen ist besonders klein was der Portabilität sehr zu gute kommt. Gravierende Nachteile gegenüber der direkten Implementierung sind nicht vorhanden, obwohl die BITS Implementierung hauptsächlich auf Host Implementierungen abzielt ist die Gateway Funktionalität ohne größeren zusätzlichen Aufwand realisierbar. Die *Bump-in-the-Wire* (BITW) Lösung zielt auf eine Hardware basierte Implementierung ab. IPSec BITW Implementierungen sind meist kleine Betriebssysteme, wie sie auch bei Routern zum Einsatz kommen, die auf die Bearbeitung von Netzwerk-Verkehr optimiert sind. Bei solchen Lösungen werden oft Teile, zum Beispiel kryptographische Algorithmen, in Hardware realisiert. Der IPSec Funktionsumfang ist vergleichbar mit dem von *Bump-in-the-Stack* Implementierungen mit Ausnahme daß, Benutzer basierte Selektoren nicht implementiert werden können. Der Entwicklungsaufwand ist allerdings um einiges höher als bei allen anderen Lösungen.

[1] In Form eines Benutzer-Kontos eines Computersystems

KAPITEL
FÜNF

Design

5.1 Entwurfs Übersicht

Die beste Möglichkeit für die Entwicklung einer portablen IPSec Implementierung bietet der *Bump-in-the-Stack* (BITS) Ansatz. Der BITS Ansatz setzt an der Stelle im Betriebssystem beziehungsweise an der Stelle im Netzwerk-Stack an ‚die prinzipiell so überall vorhanden ist. An der Schnittstelle zwischen der IP und der Netzwerk-Schicht.

5.1.1 Das Konzept

Das generelle Funktionsprinzip des Schichten basierten Aufbaus (im TCP/IP Umfeld, siehe 2.1.2) ist, daß jede Implementierung einer Schicht zwei Funktionen hat: eine zum *Senden* und eine zum *Empfangen*.

Im Falle der IP-Schicht, würde die TCP-Schicht (welche höher liegt) die *Senden* Funktion der IP-Schicht „aufrufen“ und ihr ein TCP-Paket zum Versenden übergeben. Die IP-Schicht fügt nun das IP-Header zu den TCP-Daten hinzu und ruft die *Senden* Funktion der Netzwerk-Schicht auf. Die Netzwerk-Schicht versendet dann zum Schluß das Datenpaket über das physische Netz.

Im Umkehr-Fall, ruft die Netzwerk-Schicht die *Empfangen* Funktion der IP-Schicht auf, diese ruft nun nach dem Entfernen des IP-Headers, je nach Inhalt die *Empfangen* Funktion der TCP oder UDP Schicht auf, ICMP Pakete werden innerhalb der IP-Schicht behandelt.

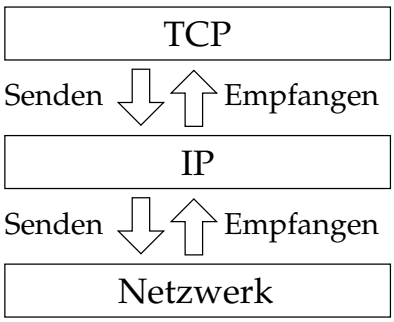

Abbildung 5.1: Schichten Aufbau für TCP/IP

Die *Bump-in-the-Stack* Implementierung setzt genau an dieser Schnittstelle an. Sie hat ebenfalls, wie die IP und die Netzwerk-Schicht eine Funktion zum *Senden* und eine Funktion zum *Empfangen*. Das Verhalten der IPSec-Schicht ist dabei genau das Gleiche wie das der IP-Schicht.

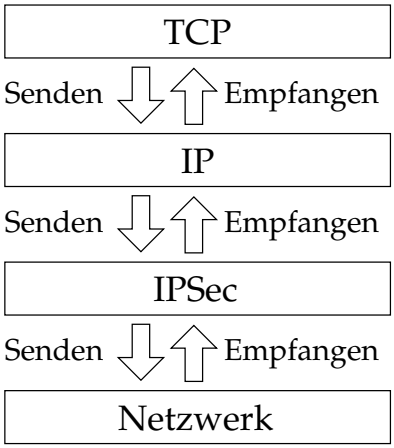

Abbildung 5.2: Aufbau inklusiv IPSec

Die *Empfangen* Funktion der IPSec-Schicht wird nun anstatt der IP-Schicht von der Netzwerk-Schicht für eintreffende IP-Pakete aufgerufen. Die IPSec-Schicht verarbeitet die eintreffenden IP-Pakete entsprechend der gesetzten Regeln, das heißt sie entfernt unter anderem die zusätzlichen IPSec-Header aus den IP-Paketen. Nach der Verarbeitung ruft die IPSec-Schicht die *Empfangen* Funktion der IP-Schicht auf, ab hier geht dann die „normale“ Verarbeitung weiter. Die IP-Schicht ruft jetzt natürlich auch die *Senden* Funktion der IPSec-Schicht anstatt die der Netzwerk-Schicht auf.

5.1.2 Portierbarkeit

Da die Umsetzung (Implementierung) der einzelnen Schichten von Betriebssystem zu Betriebssystem unterschiedlich ist, ist es nötig eine Abstraktion Schicht innerhalb des IPSec Moduls zu haben um so eine Anpassung an das jeweilige Ziel-Betriebssystem zu vereinfachen. Diese Abstraktion Schicht wird, wie in der Abbildung 5.3 zu sehen, außen um das eigentliche IPSec Modul herum gebaut.

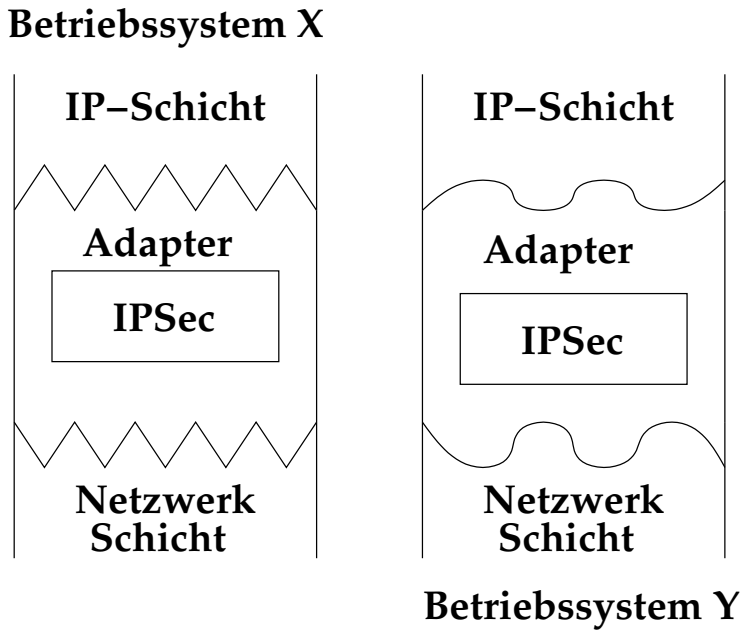

Abbildung 5.3: Der Adapter

Diese äußere Abstraktions-Schicht wird hier als *Adapter* bezeichnet. Auf die Entwicklung eines Betriebssystem spezifischen *Adapters* wird hier und im Laufe der Arbeit nicht weiter eingegangen, da dies den Rahmen der Arbeit sprengen würde beziehungsweise die Entwicklung eines solchen *Adapters* nicht Teil der Aufgabenstellung war. Der *Adapter* als solches wird allerdings mehrfach im weiteren Verlauf des Designs referenziert.

5.2 System Design

Wie in der Konzeptionierung beschrieben wird das IPSec System in Form einer Protokoll-Schicht implementiert. Daraus ergibt sich ein relativ großes Modul mit einer relativ kleinen Schnittstelle bestehend aus einer Funktion zum *Senden* und einer zum *Empfangen* von Datagrammen. Auf den Aufbau der Schnittstelle selbst wird in der Implementierung noch weiter eingegangen, im jetzt folgenden Abschnitt wird der interne Aufbau und die Funktionsweise des IPSec Moduls erläutert.

Die beschriebenen IPSec Verarbeitungsschritte dienen zusammen mit den Vorgaben über den Aufbau beziehungsweise die Funktionalität der IPSec Datenbanken (siehe 3.2) als Basis für den internen Aufbau des IPSec Moduls (in Folge nur noch als *Modul* bezeichnet).

Da das *Modul* zwischen der IP-Schicht und der Netzwerk-Schicht sitzt ist es nötig auf die IP-Fragmentierung (siehe 2.3) einzugehen. Dies geschieht gleich zu Beginn, da dies im weiteren Verlauf des Designs keine Rolle mehr spielt und somit nur kurz erläutert werden muß.

5.2.1 IP-Fragmentierung

Da das *Modul* die Funktion einer IPSec Host Implementierung erfüllen soll, was den hauptsächlichen Einsatz von Transport Mode SAs zu Folge hat, wurde entschieden daß, das *Modul* für die Verarbeitung von IP-Datagrammen ausgelegt wird.

Das heißt, das eigentliche *Modul* kann nur komplette IP-Datagramme verarbeiten und IP-Fragmente müssen vor der eigentlichen IPSec Verarbeitung beziehungsweise bevor sie an eine der beiden Einstiegsfunktionen (*Senden*, *Empfangen*) übergeben werden, in ein komplettes IP-Datagramm zusammen gesetzt werden. Dieser Zusammensetzungs Prozeß wird aus verschiedenen Gründen in den *Adapter*, welcher die Anpassung an ein spezifisches Betriebssystem vornimmt (siehe 5.1.2), verlagert. Hierauf wird später noch weiter ein gegangen.

Generell kann IPSec auch mit IP-Fragmenten umgehen (im *Tunnel Mode*) da aber ein Minimum an Informationen aus den höheren Protokollen (z.B. die TCP oder UDP Header), zur Selektion einer *Security Policy*, benötigt werden, ist es generell sinnvoll IP-Fragmente vor der IPSec Verarbeitung zu einem Datagramm zusammenzusetzen. Weiterhin müssen fragmentierte IPSec Datagramme (für eingehenden Netzwerk-Verkehr) sowieso vor der Verarbeitung zusammengesetzt werden. Auch die eventuell nötige Fragmentierung (siehe 2.3) vor der Versendung des Datagramms über das Netz, geschieht außerhalb des eigentlichen *Moduls*, in dem *Adapter*. Der sich hieraus ergebene Aufbau des *Adapters* wird durch die Abbildung 5.4 sichtbar dargestellt.

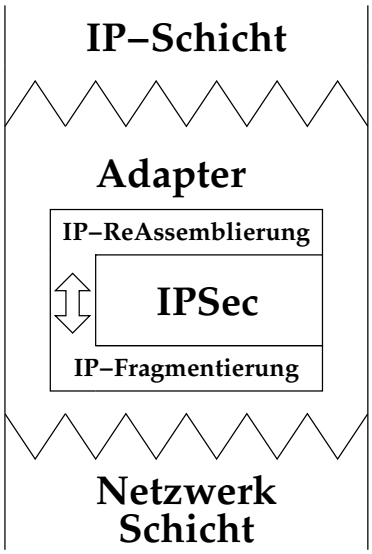

Abbildung 5.4: Der Adapter mit Fragmentierung / ReAssemblierung

Die angesprochene Auslagerung der Komponente, welche für die Zusammensetzung von IP-Fragmenten zu IP-Datagrammen beziehungsweise für die Fragmentierung (Zerlegung in Fragmente) zuständig ist, geschieht aus mehreren Gründen. Diese Gründe werden hier kurz anhand von Stichpunkten erläutert.

logische Trennung Da die IP-Fragmentierung nicht mit der eigentlichen IPSec Verarbeitung zusammen hängt, wird diese von ihr logisch getrennt.

Portierbarkeit/Effizienz Da die einzelnen Betriebssysteme, bezüglich ihres Verhaltens was die Fragmentierung von IP-Datagrammen angeht, Eigenheiten besitzen ist es logisch diesen Teil in den *Adapter* zu verlagern um so die Portabilität zu fördern.

5.3 Datenbanken

5.3.1 Die Security Policy Database

In der *Security Policy Database* (SPD) ist spezifiziert wie eingehender beziehungsweise ausgehender Netzwerk-Verkehr behandelt wird. Da die Selektoren (siehe 3.1.4) jeweils von der Richtung (eingehend beziehungsweise ausgehend) des zu behandelnden IP-Datagramms abhängig sind, ist es nötig (und auch in [6] spezifiziert) zwei *Security Policy Databases* (eine für eingehenden und eine für ausgehenden Netzwerk-Verkehr) zu halten.

Der Aufbau der SPD ist recht einfach und benötigt daher nur eine kurze Beschreibung. Da das Verhalten, die Reihenfolge in der die einzelnen Politiken (Datenbank Einträge) mit dem zu prüfenden IP-Datagramm verglichen werden, immer vorhersagbar sein muß und weil die SPD keiner dauernden Veränderung ausgesetzt ist, wird die SPD in Form einer einfachen Liste verwaltet.

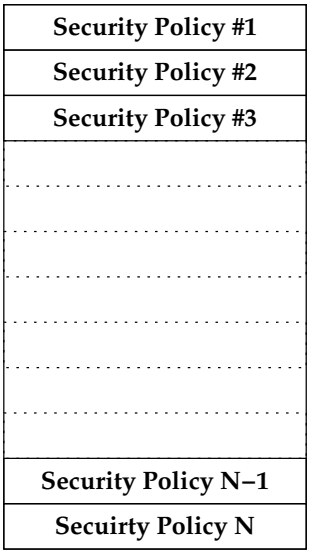

Abbildung 5.5: Security Policy Database

5.3.2 Die Security Association Database

Die *Security Association Database* (SAD) beinhaltet alle aktiven *Security Associations* (SAs) des Systems. Da die einzelnen SAs beziehungsweise SA Bündel im Zusammenhang mit einer *Security Policy* (einem Eintrag in einer der beiden SPDs) existieren und weil jede SA anhand des Tripels (*Security Parameter Index* (SPI), IP-Zieladresse und IPSec Protokoll) eindeutig identifiziert werden

kann, muß die SAD **nicht** wie die SPD in einer Liste gehalten werden.

Bei dem Design der SAD müssen die verschiedenen IPSec Verarbeitungsschritte und ihre Reihenfolge (siehe 3.2) in Zusammenhang mit der Richtung des Netzwerk-Verkehrs betrachtet werden; eingehende IP-Datagramme (wenn sie eines der beiden IPSec Protokolle transportieren) werden anhand des Tripels (SPI, IP-Zieladresse und IPSec Protokoll) einer SA zugeordnet, ausgehender IP Netzwerk-Verkehr hingegen muß erst gegen die SPD geprüft werden um festzustellen ob IPSec Verarbeitung überhaupt stattfindet. Anhand dieser Tatsachen ist es sinnvoll die SAD wiederum in zwei Teile zu teilen.

Ausgehend

Die SAD für ausgehenden Netzwerk-Verkehr wird direkt in die SPD (für ausgehenden Netzwerk-Verkehr) integriert. Hierbei enthält jeder Eintrag der SPD eine Liste mit SAs welche direkt zu dieser gehören.

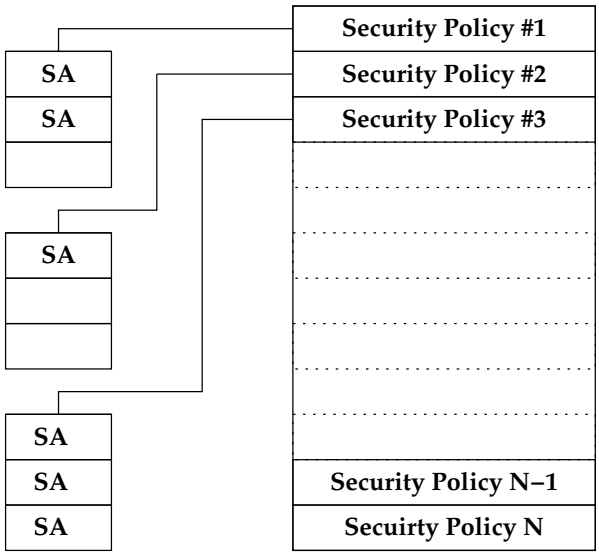

Abbildung 5.6: Security Association Database für Ausgehenden Netzwerk-Verkehr

Eingehend

Die SAD für eingehenden Netzwerk-Verkehr ist, anders als die SAD für ausgehenden Netzwerk-Verkehr, nicht aufgeteilt da sie direkt befragt wird wenn ein IP-Datagramm, welches eines der IPSec Protokolle (AH oder ESP) transportiert, „empfangen“ wird (siehe 3.2).

Um die Zugriffszeiten auf die SAD möglichst gering zu halten ist es sinnvoll die SAD in einer effizienten Datenstruktur abzulegen. Diese Überlegung wird von der Tatsache, daß eine SA eindeutig anhand des bereits mehrfach genannten Tripels identifiziert wird, unterstützt.

Die Überlegung hierzu war zum einen der Einsatz einer Hashtabelle bei der der Index aus dem Tripel (SPI, IP-Zieladresse und IPSec Protokoll) berechnet wird. Da aber die Eigenschaften von

Hashtabellen, wie beispielsweise die feste Größe einer Tabelle beziehungsweise der enorme Aufwand welche zur Vergrößerung einer Tabelle betrieben werden muß, nicht hinnehmbar ist, mußte eine andere Möglichkeit gesucht werden.

Diese andere Möglichkeit stellte die Verwendung eines Binärbaums dar, wobei jede SA einen Knoten bildet. Binärbäume haben gegenüber von Hashtabellen nicht die Eigenschaft der festen Größe beziehungsweise die Anzahl der zu erwartenden Elemente (welche der Baum verwalten muß) spielt keine Rolle. Als Vergleichswert für die Anordnung der einzelnen SAs innerhalb des Baumes wird das SA Tripel genutzt.

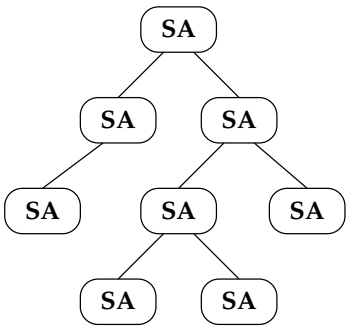

Abbildung 5.7: Security Association Database für Eingehenden Netzwerk-Verkehr

5.4 ICMP Behandlung

ICMP ist zwingend notwendig für die Funktion von IP, daher ist es wichtig ICMP Nachrichten nicht generell zu verwerfen (siehe 4.1.2).

Wie in der Analyse beschrieben wurde, stellen gesicherte ICMP-Nachrichten, das heißt Nachrichten welche durch ESP oder AH gesichert sind, kein Problem dar, sie werden nach der IPSec Verarbeitung einfach an den Host weitergeleitet.

5.4.1 Ungesicherte ICMP-Nachrichten

Für die Behandlung von unsicheren ICMP-Nachrichten wurden zwei zusätzliche Selektoren eingeführt. Diese funktionieren ähnlich wie die Selektoren für TCP und UDP Ports, das heißt sie werden nur geprüft wenn das entsprechende höhere Protokoll (ICMP) in dem IP-Datagramm transportiert wird.

Hierdurch läßt sich die Behandlung von ungesicherten ICMP-Nachrichten sehr elegant durch die *Security Policy Database* regeln.

5.4.2 Unterstützung der Path MTU Discovery

Die *Path MTU* wird wie durch die Spezifikation [6] angegeben auf *Security Assocation* (SA) Ebene verwaltet. In jedem Eintrag der *Securit Association Database* werden drei Werte zur Unterstützung der *Path MTU Discovery* gespeichert:

- Anzahl Bytes die durch diese SA von der *Path MTU* abgezogen werden müssen
- die aktuelle *Path MTU*

- eine Lebenszeit für die aktuelle *Path MTU*

Ausgehende Datagramme werden vor der SA Verarbeitung geprüft ob sie kleiner oder gleich, der gespeicherten *Path MTU* der SA sind. Ist dies nicht der Fall wird das Datagramm verworfen und eine entsprechende ICMP Nachricht an den Host versendet. Die Lebenszeit wird dazu verwendet von Zeit zu Zeit zu große Datagramme passieren zu lassen, um so die eventuelle Veränderung des Pfades und so die Veränderung der *Path MTU* zu bemerken.

KAPITEL
SECHS

Implementierung

6.1 System-Entwicklung

Bei der Entwicklung auf Betriebssystem-Ebene beziehungsweise beim Arbeiten im Kernel muß man sich über dessen Funktionsweise beziehungsweise über die allgemeine Funktionsweise von Betriebssystem-Kernen im klaren sein. In diesem Fall ist nur die Funktionsweise des unteren Teiles des Netzwerk-Bereiches interessant.

6.1.1 Protokoll-Stacks

Wenn man von den vier Schichten des TCP/IP Modells (2.1.2) ausgeht, ist die unterste ein Abstraktions-Schicht um die IP-Schicht an jede Art von Netzwerk-Adapter zu koppeln. Die IP-Schicht „registriert“ sich bei der Abstraktions-Schicht um ihr mitzuteilen, daß sie für IP-Daten zuständig ist.

Bei der Registrierung werden sogenannte Callback-Funktionen ausgetauscht. Über diese Callback-Funktionen kommunizieren die beiden Schichten miteinander. Generell funktioniert dies genau so wie 5.1.1 beschrieben.

Das Wichtigste an diesem Aufbau ist, daß der gesamte Ablauf Asynchron ist, das bedeutet alle Funktionen können mehrfach parallel ausgeführt werden. Durch diese Parallelität in den Abläufen müssen alle Datenstrukturen die während der Verarbeitung benutzt beziehungsweise verändert werden, synchronisiert (blockiert) werden um zu verhindern, daß die Datenstrukturen durch paralleles schreiben zerstört werden. Dieses Blockieren wird durch sogenannte *Spinlocks* bewerkstelligt.

6.1.2 Spinlocks

Spinlocks dienen dazu kurzzeitig Kritische-Bereiche eines Programmes zu blockieren, so daß dieser Bereich des Programms nicht parallel ausgeführt wird.

Bei *Spinlocks* wird anders als bei Semaphoren, aktiv auf die Freigabe gewartet (busy waiting), dies hat den Vorteil daß der wartende Kernel-Prozeß nicht unterbrochen wird. Ein *Spinlock* wird durch eine *lock* Funktion blockiert und durch eine *unlock* Funktion freigegeben. Ist der *Spinlock* bei dem Aufruf von *lock* schon blockiert, wird solange gewartet bis der *Spinlock* durch *unlock* freigegeben wird.

Semaphoren funktionieren nach einem anderen Prinzip, hier wird der „wartende“ Prozeß solange unterbrochen bis der Kritische-Bereich wieder „frei“ ist. Semaphoren sind allerdings zu aufwendig für kurze Blockade-Zeiten und werden daher **hier** nicht benutzt.

6.2 Implementierung des IPSec Moduls

Die Schnittstelle des Moduls wurde prinzipiell so wie im Design (5.1.1) beschrieben implementiert, das bedeutet es gibt zwei Funktionen: eine zum *Senden* und eine zum *Empfangen* von Datagrammen.

```
ipsec_result ipsec_send    ( ip_datagram dgram_old,
                             ip_datagram dgram_new )

ipsec_result ipsec_receive ( ip_datagram dgram_old,
                             ip_datagram dgram_new )
```

Die Besonderheit der Funktionen ist, daß das Eingabe-Datagramm nicht verändert wird. Statt dessen wird für alle Aktionen, bei denen das Datagramm verändert werden würde, ein neues Datagramm erstellt. Diese Trennung mußte aus Portabilitätsgründen vorgenommen werden; die Speicherverwaltung im Betriebssystem-Kern ist teilweise sehr spezifisch so daß reservierter Speicher nicht einfach größer oder kleiner gemacht werden kann. Durch diese Trennung von Eingabe und Ausgabe Datagrammen wurden fünf verschiedene Rückgabewerte für die Schnittstellen definiert.

IPSEC_BLOCK	Datagramm verwerfen
IPSEC_OLD_TO_NET	Eingabe Datagramm ins Netz senden
IPSEC_OLD_TO_HOST	Eingabe Datagramm an das System weiterleiten
IPSEC_NEW_TO_NET	Ausgabe Datagramm ins Netz senden
IPSEC_NEW_TO_HOST	Ausgabe Datagramm an das System weiterleiten

Tabelle 6.1: Mögliche Rückgabewerte der Schnittstellen

Die Rückgabewerte geben dabei an, wie mit dem „alten“ beziehungsweise mit dem „neu erstellten“ Datagramm verfahren werden muß. Die verschiedenen Aktionen geben jeweils an welches Datagramm weitergeleitet werden soll beziehungsweise ob es verworfen werden muß. Die Flußrichtung bei der Weiterleitung ist dabei sehr wichtig.

Im Normalfall wird das durch ipsec_send() erzeugte Datagramm ins Netz gesendet. Da aber auch ein ICMP-Datagramm erzeugt werden kann, welches nicht ins Netz sondern zurück an den Host gesendet werden muß, muß die *Adapter* Implementierung dies entsprechend ausführen können, das heißt sie muß für beide Schnittstellen das neue Datagramm jeweils in beide Richtungen senden können.

6.2.1 Die Security Policy Database

Die *Security Policy Database* wurde in Form zweier Listen (jeweils eine für eine Richtung; eingehend und ausgehend) implementiert. Für die *Security Policys* wurden folgende Selektoren (siehe 3.1.4) implementiert:

- IP-Zieladresse mit Maske
- IP-Quelladresse mit Maske
- Transport-Protokoll

- Ziel-Port (für TCP und UDP)
- Quell-Port (für TCP und UDP)

Durch die Maske können einzelne IP-Adressen und Bereiche angegeben werden.

Für die Behandlung von ICMP-Nachrichten wurden zwei weiter Selektoren implementiert. Diese werden allerdings nur für eingehenden Netzwerk-Verkehr geprüft (siehe 5.4).

ICMP-Code Der Code gibt die Art der ICMP-Nachricht an

ICMP-Type Der Type spezifiziert die Subklasse der Nachricht

Für die Verwaltung der SAD Einträge (für ausgehenden Netzwerk-Verkehr) enthält jede *Security Policy* eine statische Liste für vier Einträge (SAs). Im Normalfall wird meistens eine bis zwei SAs für eine Verbindung genutzt, daher sollte die Beschränkung auf vier SAs je *Security Policy* kein Problem darstellen.

6.2.2 Die Security Association Database

Für die *Security Association Database* (für eingehenden Netzwerk-Verkehr) wird, wie im Design (5.3.2) beschrieben, ein Binärbaum benutzt.

Für den Binärbaum wurde ein *SplayTree* [16] implementiert. Diese Art von Binärbaum unterscheidet sich vom Aufbau her nicht von einem normalen Binärbaum. Der *SplayTree* definiert allerdings einige Optimierungs-Schritte, welche jeweils beim Zugriff auf den Baum (Suchen, Einfügen, Löschen) durchgeführt werden. Durch diese Optimierung werden alle oft benutzten Elemente (SAs) nahe an die Wurzel gebracht. Hieraus ergibt sich ein schnellerer Zugriff für oft benutzte Elemente.

6.2.3 Ausgehender Netzwerk-Verkehr

Für jedes ausgehende IP-Datagramm wird ipsec_send() durch die *Adapter* Implementierung aufgerufen und somit die IPSec Verarbeitung in Gang gesetzt. Die Verarbeitung findet ab hier prinzipiell, wie in 3.2 beschrieben, statt. Zuerst wird die passende *Security Policy* ermittelt. Wird keine passende *Security Policy* gefunden, wird dem *Adapter* mitgeteilt daß das Datagramm verworfen werden muß.

Gibt die gefundene *Security Policy* an daß das Datagramm durchgelassen werden soll, wird dies dem *Adapter* durch IPSEC_OLD_TO_NET mitgeteilt. Soll das Datagramm verworfen werden, wird IPSEC_BLOCK zurückgegeben.

Wenn die *Security Policy* die Verwendung von IPSec angibt, werden alle SAs der Reihenfolge nach verarbeitet, dabei wird vor der Verarbeitung jeder einzelnen SAs die entsprechenden Schritte für die Unterstützung der *Path MTU Discovery* durchlaufen.

Hierbei wird zunächst geprüft ob das DF-Bit (siehe 4.1.3) gesetzt ist. Ist dies der Fall, wird die Größe des Datagramms mit der gespeicherten *Path MTU* verglichen. Ist das Datagramm kleiner oder gleich der *Path MTU* wird es anhand der SA Verarbeitet. Ist das Datagramm größer wird die Lebenszeit der *Path MTU* geprüft, ist diese aktuell muß das Datagramm verworfen werden und eine entsprechende *Path MTU* Nachricht an den Host gesendet werden. Die *Path MTU* ICMP Nachricht wird als neues Datagramm zurückgegeben und der *Adapter* wird angewiesen dieses neue Datagramm an den Host weiterzuleiten. Ist die Lebenszeit abgelaufen, wird das zu große Datagramm nicht verworfen, hierdurch wird die *Path MTU Discovery* initiiert.

Tritt bei SA Verarbeitung ein Fehler auf wird das Datagramm verworfen, dies wird dem *Adapter* durch IPSEC_BLOCK mitgeteilt. Ansonsten wird dem *Adapter* durch IPSEC_NEW_TO_NET mitgeteilt, daß das neue Datagramm über das Netz zu versenden ist.

6.2.4 Eingehender Netzwerk-Verkehr

Die *Adapter* Implementierung ruft für jedes eingehende IP-Datagramm die Funktion ipsec_receive() auf. Hier wird als erstes der transportierte Protokolltyp (das Next-Header, siehe 2.3) geprüft, ist dieser **AH** oder **ESP** wird die passende SA gesucht.

Zum Auffinden der passenden SA wird das Tripel (siehe 3.1.3) aus dem Datagramm geholt, das heißt die IP-Zieladresse und der Protokolltyp wird aus dem IP-Header extrahiert; der *Security Parameter Index* (SPI) wird aus dem entsprechenden IPSec-Header (AH oder ESP) extrahiert.

Das Datagramm wird dann anhand der gefundenen SA Verarbeitet. Nach der erfolgreichen Verarbeitung wird erneut der transportierte Protokolltyp geprüft, handelt es sich wiederum um ein IPSec-Header, beginnt der Ablauf erneut.

Wird keine passende SA gefunden oder schlägt die Verarbeitung fehl, wird IPSEC_BLOCK an den *Adapter* zurückgegeben der dann das Datagramm verwirft.

Vor der Weiterleitung des verarbeiteten Datagramms wird geprüft ob es sich um eine ICMP *Path MTU Discovery* Nachricht handelt. Ist dies der Fall wird die Nachricht dem entsprechenden *Security Association Database* Eintrag zugeordnet und weiterverarbeitet. Dieser Prozeß wird im nächsten Abschnitt beschrieben.

Prinzipiell müßte das Datagramm nach der SA Verarbeitung gegen die *Security Policy Database* geprüft werden, um die verwendeten SAs mit den spezifizierten SAs zu vergleichen. Dieser Prozeß ist allerdings sehr aufwendig, da hierfür unter anderem während der SA Verarbeitung eine Liste der verwendeten SAs erstellt werden müßte. Daher wird nur geprüft ob es für das Datagramm eine *Security Policy* gibt welche IPSec-Verarbeitung vorschreibt. Diese Optimierung des abschließenden Vergleiches bringt im Regelfall keine Probleme mit sich, da der Sender im Normalfall vertrauenswürdig ist.

Wird zu Beginn kein IPSec-Header gefunden wird gleich die *Security Policy Database* befragt und das Datagramm wird verworfen oder durchgelassen. Für ICMP *Path MTU Discovery* Nachrichten wird wie im folgenden Abschnitt verfahren.

6.2.5 Path MTU Discovery Verarbeitung

Die *Path MTU Discovery* Nachricht [10] enthält das IP-Header und die ersten 64-Bits des Datagramms, durch welche diese ausgelöst wurde. Durch den *Security Parameter Index* (SPI), welcher jeweils in den ersten 64-Bits von AH oder ESP enthalten ist, kann der *Security Association Database* Eintrag einfach gefunden werden. Die enthaltene *Path MTU* wird in dem gefundenen Eintrag abgelegt.

6.3 Protokoll-Verarbeitung

Die unterste Ebene des IPSec Moduls besteht aus der eigentlichen Protokoll-Verarbeitung. Da beide Protokolle viele gemeinsame Verarbeitungsschritte haben, wurde die Protokoll-Verarbeitung für AH und ESP zusammengelegt um redundanten Code zu vermeiden.

Die Verarbeitung ist wiederum in eingehend und ausgehend unterteilt.

```
ipsec_result ipsec_sa_in  ( ipsec_sa    sa,
                            ip_datagram dgram_in,
                            ip_datagram dgram_out )

ipsec_result ipsec_sa_out ( ipsec_sa    sa,
                            ip_datagram dgram_in,
                            ip_datagram dgram_out )
```

Der jeweiligen Funktion wird das zu bearbeitende Datagramm und der entsprechende *Security Association Database* Eintrag übergeben.

6.3.1 Ausgehend

Im ersten Schritt wird die Größe des neuen Datagramms berechnet, hierbei spielen sämtliche Parameter der SA eine Rolle (z.B. ein zusätzliches IP-Header im *Tunnel Mode*, Länge des Padding oder die Größe der Authentifizierungs Daten). Im zweiten Schritt wird das neue Datagramm zusammen gesetzt, dies schließt unter anderem die Aufaddierung der Sequenz-Nummer und das Kopieren der Daten des alten Datagramms mit ein. Nach dem das neue Datagramm zusammen gebaut ist muß die IP-Header Checksumme (siehe 2.3) neu Berechnet werden.

Zum Schluß folgen die Protokoll spezifischen Schritte, generell heißt das es wird erst verschlüsselt und dann authentifiziert.

6.3.2 Eingehend

Gibt die SA die Verwendung des Anti-Replay Schutzes an, wird im ersten Schritt die Sequenz-Nummer extrahiert. Für die Prüfung der Sequenz-Nummer wird das Beispiel Verfahren aus [6] in einer leicht veränderten Form benutzt. Wird das Datagramm als Replay erkannt, wird die Verarbeitung mit einem entsprechenden Rückgabewert beendet.

Ist die Sequenz-Nummer innerhalb des erlaubten Bereiches, folgen entsprechend den Protokollen erst die Authentifizierung und dann die Entschlüsselung der Daten. Nach erfolgreicher Bearbeitung wird die Sequenz-Nummer in die SA Struktur übernommen.

6.3.3 Die Authentication Header Implementierung

Bei der *Authentication Header* (AH) Implementierung gibt es ein ziemlich unangenehmes Problem, nämlich die zur Berechnung der Signatur erforderliche Vorbelegung der einzelnen Header Felder (siehe 3.3.1). Diese Problem ließ sich leider nur durch das vollständig kopieren des Datagramms lösen. Das bedeutet: zur Berechnung der Signatur des AH wird das komplette Datagramm erst kopiert, dann werden alle Felder entweder mit dem vorhersagbaren Wert gefüllt oder entsprechend mit Null (0) überschrieben. Die Signatur wird dann zum Schluß über die Kopie berechnet und in das eigentliche Datagramm kopiert.

Diese Vorbelegung ist auf Empfänger und Sender Seite jeweils leicht verschieden.

Für die Signaturberechnung wurde wie es der Standard verlangt HMAC-MD5-96 [8] und HMAC-SHA-196 [9] implementiert, weiterhin wurde noch ein NULL-Authentication-Funktion (eine Dummy-Funktion zum Testen) implementiert.

6.3.4 Die Encapsulated Security Payload Implementierung

Die *Encapsulated Security Payload* (ESP) verlief prinzipiell Problemlos, allerdings wurde auf die Prüfung des Inhaltes des Padding-Feldes (beim Empfang) verzichtet. Die Prüfung ist relativ sinnlos, da sie erst nach der Signaturprüfung beziehungsweise nach der Entschlüsselung passiert und somit keinen wirklichen zusätzlichen Schutz bietet.

Zur Verschlüsselung wurde AES-CBC [15] und ein NULL-Verschlüsselung (zum Testen) implementiert, für die Signaturberechnung (Authentizität) wurden wie für AH, HMAC-MD5-96 und HMAC-SHA-196 verwendet.

KAPITEL

SIEBEN

Fazit

IPSec stellt momentan die wohl beste und umfangreichste Sicherungsmöglichkeit für IP-Netzwerke dar. Durch IPSec kann praktisch jede Art von Sicherung realisiert werden, angefangen bei der einfach Daten Authentisierung bis hin zur Erstellung von verschlüsselten VPNs (Virtuelles Privates Netzwerk). Durch den Aufbau beziehungsweise durch den Ansatz auf IP-Ebene, kann IPSec ohne Probleme in einem nicht IPSec Umfeld für die Sicherung eingesetzt werden. Weiterhin ist IPSec für Anwendungen und Benutzer völlig transparent.

Leider ist die IPSec Architektur beziehungsweise das Thema IPSec sehr umfangreich und komplex. Allein die Tatsache das die IPSec Spezifikation in über einem Dutzend RFCs aufgeteilt ist, macht dieses sehr deutlich. Durch die Komplexität von IPSec ist eine Implementierung nicht besonders einfach, gerade bei meiner Arbeit bin ich immer wieder über Details gestolpert die ich so „vorher“ noch nicht entdeckt hatte.

7.1 Gedanken zu IPSec

Während meiner Arbeit an IPSec bin ich auf einige Dinge gestoßen die mir nicht sonderlich gut gefallen. Ein Punkt ist, daß *Authentication Header*, dieses ist durch seinen Aufbau sehr rechenintensiv (Datagramme müssen zur Berechnung der Signatur kopiert werden da bestimmte Felder für die Berechnungen einen anderen Wert haben müssen). Meiner Meinung nach leistet das *Authentication Header* nicht genug im Gegensatz zu dem von ihm verschlungenen Rechenaufwand. Ein anderer Punkt ist, daß die Authentizitäts Funktionalität der *Encapsulated Security Payload* (ESP) optional ist, das heißt es währe theoretisch möglich, Verschlüsselung ohne Authentizität zu benutzen. Ein solches Setup würde einen Großteil der Sicherungsbemühungen untergraben, da Verschlüsselte Datagramme einfach verändert werden könnten. Weiterhin wird die Authentizitäts Funktionalität für den Anti-Replay Mechanismus benötigt, was wiederum dafür spricht Authenzität nicht optional zu machen.

7.1.1 Andere Gedanken zu IPSec

In seiner Evaluierung [3] der IPSec Architektur, empfiehlt der bekannte Kryptoanalytiker Bruce Schneier das Entfernen des *Transport Modes* um so ein Teil der Komplexität der Architektur zu elimieren. Außerdem ist er der Meinung das ESP im *Tunnel Mode* die ohnehin am häufigsten benutzte Variante ist.

Dieser Meinung kann ich mich nicht anschließen. Meiner Ansicht nach währe die Entfernung des *Transport Modes* sehr kurzsichtig da es in Zukunft immer wichtiger sein wird, Verbindungen von Anfang bis Ende komplett zu sichern, hierfür wurde der *Transport Mode* entwickelt. Würde

es nur den *Tunnel Mode* geben, müßte dieser auch für einfache Verbindungen benutzt werden. Somit würde der durch IPSec erlittene Effizienzverlust noch unnötig erhöht.

7.2 Ausblick

Da IPSec ein essentieller Bestandteil von IPv6 ist, wird IPSec in den kommenden Jahren wohl noch um einiges an Bedeutung dazu gewinnen. Weiterhin steigt auch das Sicherheitsbedürfnis auch stetig an, dies sollte IPSec ebenfalls einen Schub geben.

Da mich das Thema IPSec sehr begeistert hat, werde ich mich wohl noch längere Zeit damit beschäftigen.

LITERATURVERZEICHNIS

[1] A8, F. I. Coseda : Comprehensive security for distributed architectures http://www.igd.fhg.de/igd-a8/projects/coseda/index.html.

[2] DAY, J. D., AND ZIMMERMAN, H. The OSI reference model. In *Proc. IEEE (USA)* (1983), pp. 1334–1340. 5 Refs. Treatment GENERAL OR REVIEW.Published as Proc. IEEE (USA), volume 71, number 12.

[3] FERGUSON, N., AND SCHNEIER, B. A cryptographic evaluation of IPsec. Technical report, Counterpane Internet Security, 3031 Tisch Way, Suite 100PE, San Jose, CA 95128, USA, 2000.

[4] HARKINS, D., AND CARREL, D. RFC 2409: The Internet Key Exchange (IKE), Nov. 1998. Status: PROPOSED STANDARD.

[5] KENT, S., AND ATKINSON, R. IP encapsulating security payload (ESP). Internet Request for Comment RFC 2406, Internet Engineering Task Force, Nov. 1998.

[6] KENT, S., AND ATKINSON, R. Security architecture for the internet protocol. *RFC 2401* (Nov. 1998).

[7] KRAWCZYK, H., BELLARE, M., AND CANETTI, R. RFC 2104: HMAC: Keyed-hashing for message authentication, Feb. 1997. Status: INFORMATIONAL.

[8] MADSON, C., AND GLENN, R. RFC 2403: The use of HMAC-MD5-96 within ESP and AH, Nov. 1998. Status: PROPOSED STANDARD.

[9] MADSON, C., AND GLENN, R. RFC 2404: The use of HMAC-SHA-1-96 within ESP and AH, Nov. 1998. Status: PROPOSED STANDARD.

[10] MOGUL, J. C., AND DEERING, S. E. RFC 1191: Path MTU discovery, Nov. 1990. Obsoletes RFC1063. Status: DRAFT STANDARD.

[11] POSTEL, J. RFC 768: User datagram protocol, Aug. 1980. Status: STANDARD. See also STD0006 [**?**].

[12] POSTEL, J. RFC 791: Internet Protocol, Sept. 1981. Obsoletes RFC0760. Status: STANDARD.

[13] POSTEL, J. RFC 792: Internet Control Message Protocol, Sept. 1981. Obsoletes RFC0777 [**?**]. Updated by RFC0950 [**?**]. See also STD0005 [**?**]. Status: STANDARD.

[14] POSTEL, J. RFC 793: Transmission control protocol, Sept. 1981.

[15] S. FRANKEL, S. KELLY, R. G. The aes cipher algorithm and its use with ipsec.

[16] SLEATOR, D. D., AND TARJAN, R. E. Self-Adjusting Binary Trees. *Proceedings of the ACM SIGCAT Symposium on Theory of Coumputing* (1983), 235–245.