# Flexible Datentunnelung per ICMP

Matthias Hammerle

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2011

# Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 22. Juni 2011

Matthias Hammerle

# Inhaltsverzeichnis

**Erklärung** **iii**

**Vorwort** **vi**

**Kurzfassung** **vii**

**Abstract** **viii**

**1 Einleitung** **1**
1.1 Kommunikation in Computernetzwerken . . . . . 3
1.1.1 Offene Kommunikation . . . . . 5
1.1.2 Verdeckte Kommunikation . . . . . 6

**2 Kommunikationsprotokolle** **10**
2.1 Internet Protocol . . . . . 12
2.1.1 IPv4 . . . . . 12
2.1.2 IPv6 . . . . . 14
2.2 Internet Control Message Protocol . . . . . 16
2.2.1 Format . . . . . 17

**3 Flexible Datentunnelung** **23**
3.1 Motivation . . . . . 23
3.2 Flexibilität . . . . . 25
3.3 Kommunikationsschemata . . . . . 26
3.4 SOCKS . . . . . 30
3.5 Tunnelung . . . . . 34
3.5.1 Anmeldung . . . . . 37
3.5.2 Verbindungsanfrage . . . . . 43
3.5.3 Datentransfer . . . . . 46
3.5.4 Abmeldung . . . . . 48
3.6 Demultiplexing . . . . . 49

**4 Alternative Ansätze** **51**
4.1 Datentunnel per ICMP . . . . . 51

4.1.1 TCP over ICMP . . . . . 51
4.1.2 IP over ICMP . . . . . 52
4.1.3 Kernel module . . . . . 53
4.2 Datentunnel über andere Protokolle . . . . . 54
4.2.1 TCP . . . . . 54
4.2.2 HTTP . . . . . 54
4.2.3 DNS . . . . . 55

**5 Diskussion** **56**
5.1 Entdeck- und Abwehrbarkeit . . . . . 56
5.2 Verschlüsselung . . . . . 59
5.3 Probleme . . . . . 60

**6 Fazit** **61**

**Literaturverzeichnis** **62**

# Vorwort

Ich bedanke mich in erster Linie für die stets kompetente Betreuung durch Mag. Volker Christian. Außerdem ist es mir ein Anliegen, mich für die fruchtvollen Fachdiskussionen mit Dipl.-Ing. flummy zu bedanken, die mir nicht selten neue Perspektiven zu diversen IT-bezogenen Thematiken eröffnet haben. Carmen Pemwieser sei an dieser Stelle ebenfalls für die stundenlangen Chats und Telefonate als abendliche Unterhaltung und mentale Unterstützung herzlich gedankt.

# Kurzfassung

Die Diplomarbeit beschreibt im Kern die Umsetzung eines verdeckten Kanals („covert channel“) anhand speziell präparierter ICMP Pakete.

Dieser soll in Form eines Mehrbenutzersystems über eine Proxy-Schnittstelle flexible Datentunnelung ermöglichen. Einleitend wird diesbezüglich das Konzept der steganographischen Informationsübertragung im Rahmen gängiger Kommunikationsprotokolle beschrieben und dafür insbesondere das IP und ICMP untersucht. Im Fokus ist schließlich das Design eines Transportprotokolls zur zuverlässigen Tunnelung von Nutzdaten. Dabei wird unter anderem eine Möglichkeit zur Unterstützung von Benutzerauthentifizierung und ein Verfahren zur Handhabung mehrerer, gleichzeitiger Verbindungsanfragen gezeigt.

Abschließend werden ähnliche und alternative Ansätze verglichen, sowie die Entdeckbarkeit und Abwehrbarkeit des vorgestellten Systems kritisch betrachtet.

# Abstract

This thesis describes the creation of so-called “covert channels” by crafting ICMP packets.

Such a channel is supposed to allow flexible tunneling of data by providing a multi-user system with a proxy interface. Introductorily, the concept of steganographic transmission of information is explained in the scope of current communication protocols, especially IP and ICMP. The main interest lies eventually on the design of a transport protocol capable to reliably tunnel data. In this regard, a possible way to support user authentication and multiple concurrent connection requests is shown.

To complete the picture, similar and alternative tunneling approaches are compared. Furthermore, issues like detectability and defence against covert channels of this type are reviewed.

# Kapitel 1

# Einleitung

Diese Arbeit steht ganz im Zeichen der *Kommunikation*. Das Übertragen einer Informationseinheit zwischen Akteuren geschieht von einem Sender über ein Medium unter Störeinflüssen hin zu einem Empfänger (Abbildung 1.1). Claude Shannon beschreibt in [47] dieses fundamentale Schema eines einfachen Kommunikationssystems.

Das Modell lässt sich überall dort anwenden, wo Informationen in unterschiedlichsten Ausprägungen fließen. Bei der zwischenmenschlichen Interaktion sind beispielsweise Wörter, Sätze, Betonungen und die Körpersprache jeweils Informationseinheiten, die von einer Person als Informationsquelle akustisch, visuell oder haptisch kodiert werden. Als Übertragungsmedium dient der Äther, also die Luft zur Ausbreitung von Schallwellen und elektromagnetische Strahlung für optische Reize. Auch Berührungen werden über ein Medium propagiert. Elektrische Signale leiten haptische Eindrücke über die Haut und schließlich weiter über Nervenbahnen zum Gehirn.

Die kodiert versendeten Informationseinheiten unterliegen schließlich auf dem Weg zum Empfänger – dem Gesprächspartner – Störeinflüssen, die vom Sender unabhängig durch die Umwelt verursacht werden. Ein vorbeifahrendes Automobil als Geräuschquelle oder Dunkelheit als Sichtbeeinträchtigung führt zu einem Qualitätsverlust der Information und damit möglicherweise zur Veränderung ihrer ursprünglichen Semantik.

Kommt die gestörte, qualitativ degenerierte Information beim Empfänger an, muss diese dekodiert und interpretiert werden, damit der Kommunikationspartner ihre Bedeutung verarbeiten kann. Die Möglichkeit der Dekodierung hängt von der Intensität des Störeinflusses ab. An dieser Stelle greifen Dekodierung und Interpretierung ineinander, um mithilfe menschlicher Erfahrung, Kontextwissen und Auffassungsgabe eine als sinnvoll erscheinende Informationsrekonstruktion zu erreichen.

Das grundlegende Problem und gleichzeitig die Herausforderung beim Kommunizieren einer Informationseinheit an eine Person, liegt an genau dieser Schnittstelle. Die selektive und generell subjektive Wahrnehmung, also

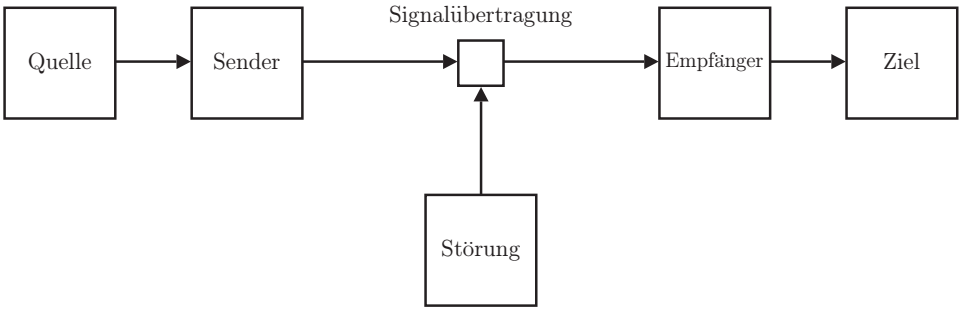



**Abbildung 1.1:** Das Modell eines Kommunikationssystems nach Shannon.

psychische Einflussfaktoren sind als Merkmale des Menschen als individuelles Wesen dafür verantwortlich, dass die Interpretation von Information bei Sender und Empfänger nie auf exakt derselben Art und Weise geschieht. Als anschauliches Beispiel seien die Bedeutungen des Kopfschüttelns in Indien erwähnt [56, S. 98]. Schüttelt der Europäer ablehnend den Kopf, kann der Inder dies fälschlicherweise als Zustimmung auffassen und würde damit die ursprüngliche Semantik negieren. Unterschiedliche kulturelle Hintergründe von Kommunikationspartnern können also zu gravierenden Missverständnissen führen. Das Perfide an solchen Situationen ist, dass man selbst dann betroffen sein kann, wenn man mit dem Hintergedanken der Konfliktvermeidung versucht, jede (überflüssige) Kommunikation zu vermeiden. Von Paul Watzlawick stammt diesbezüglich das Axiom, dass man nicht *nicht* kommunizieren könne [57]. Eben aufgrund der ständigen und unweigerlichen Körpersprache, des Subtextes, der Worte „zwischen den Zeilen“ und selbst der Bedeutung von Stille in gewissen Situationen, ist Kommunikation praktisch unvermeidbar und allgegenwärtig.

Zwischenmenschlicher Informationsaustausch kann allerdings nicht nur unmittelbar, sondern auch indirekt über elektronische Hilfsmittel erfolgen. In einem technischen Kontext kann eine Gerätschaft den Menschen als Kommunikationspartner auch vollständig ersetzen. Der Vorgang des Abrufens einer Website geschieht beispielsweise in einem mehrstufigen Prozess, bei dem der Surfer als Mensch mit einem Browser interagiert, der wiederum über Programmschnittstellen mit Netzwerk- und Datenübertragungsdiensten des Betriebssystems kommuniziert. Die in dieser Arbeit behandelte technische Komponente operiert genau an solch einer Schnittstelle zwischen Computerprogrammen. Der Mensch als Benutzer konfiguriert zwar die gewünschte Betriebsweise der Softwarekomponente und aktiviert sie bei Bedarf, interagiert aber ansonsten nicht unmittelbar damit. Stattdessen bedient er übergeordnete Programme wie Browser, Instant Messaging und ähnliche Anwendungen, welche schließlich auf die Komponente zugreifen.

## 1.1 Kommunikation in Computernetzwerken

Abgesehen von den internen Datenverarbeitungsprozessen in einem Einzelrechner, spielt auch der Informationsaustausch zwischen unabhängig arbeitenden, geographisch verteilten Geräten eine wichtige Rolle. Vor etwa fünfzig Jahren begann diesbezüglich eine längst globale informationstechnologische Revolution, die sich gemäß [53] von einer zündenden Idee in den 60er Jahren bis zum heutigen *Internet* folgendermaßen entwickelte:

Joseph Licklider hatte 1962 die Vision eines weltumspannenden Netzes von untereinander verbundenen Computern und brachte den Stein ins Rollen, indem er die DARPA[1] von dieser zukunftsweisenden Idee überzeugte. Das ARPANET[2] entwickelte sich schließlich aus einem Experiment im Jahr 1965, bei dem das erste Mal zwei Rechenanlagen über eine Telefonverbindung miteinander kommunizierten. Die Leitungsvermittlung des bestehenden Telefonsystems stellte sich dabei als unbrauchbar heraus und damit wurde das von Leonard Kleinrock am MIT schon 1961 vorgeschlagene Konzept der paketorientierten Datenvermittlung übernommen. Ende 1969 bestand das ARPANET aus nur vier Netzwerkknoten, wuchs dann aber rasant bis zum weltumspannenden Netz – dem *Internet* – heran.

Das Federal Networking Council definierte 1995 den Begriff in [52] auf diese Weise:

> "Internet" refers to the global information system that –
>
> (i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons;
>
> (ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and
>
> (iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein.

Der „global einzigartige Adressraum" wird bis dato durch das Internet Protokoll der Version 4 („IPv4") abgebildet. Die Adressierung erfolgt dabei anhand einer 32 Bit breiten Nummer, der *IP Adresse*. Damit können theoretisch maximal $2^{32}$, also etwa vier Milliarden Adressen delegiert werden. Das Internet überschritt laut der Statistik des Internet Systems Concortiums 1996 zehn Millionen angeschlossene Teilnehmer. 15 Jahre später liegt die Zahl nun bei geschätzten 800 Millionen und wird in den kommenden Jahren die Milliardengrenze durchbrechen [19].

[1] **D**efense **A**dvanced **R**esearch **P**rojects **A**gency
[2] **A**dvanced **R**esearch **P**rojects **A**gency **Net**work

Aufgrund vieler reservierter, früher großzügig an Unternehmen vergebener und anderwertig verbrauchter Adressbereiche [18], sind in der Praxis längst nicht Adressen für vier Milliarden Teilnehmer verfügbar. Am 1. Februar 2011 wurde von der IANA[3], der Koordinationsstelle für Adressierungssysteme des Internets, bereits die letzten noch verbleibenden Netzblöcke des verfügbaren IPv4 Adressraums an die fünf sog. „Regional Internet Registries" zur weiteren Delegierung vergeben [43].

Jeder Computer, der im Internet als Kommunikationsteilnehmer erreichbar sein soll, benötigt eine IP Adresse als eindeutige Kennung. Netzwerktechnische Hilfsmittel wie NAT[4] weichen diese Anforderung zwar etwas auf und führen dazu, dass nicht jedem Teilnehmer unmittelbar eine global eindeutige Adresse aus dem IPv4 Adressraum zugewiesen werden muss, eliminieren das Problem der Adressverknappung bei zunehmender Teilnehmeranzahl im Internet aber nicht. Auf die Problematik mit NAT in Zusammenhang mit Datentunnelung wird im Abschnitt 3.5 des Kapitels 3 näher eingegangen.

Als Ersatz und Lösung der Adressknappheit gilt IPv6, Nachfolger des Internet Protokolls der Version 4. Der IPv6 Adressraum ist 128 Bit breit und umfasst daher $2^{128}$ Adressen. Aufgrund des Schichtenmodells der Netzwerkprotokolle (siehe Kapitel 2) ist die Version des zugrunde liegenden IP Protokolls während eines Kommunikationsvorgangs für die betreffende Anwendung in der Regel kaum relevant. Lediglich beim Aufbau einer Verbindung und damit der Notwendigkeit der Angabe des Verbindungsziels, kommt eine Anwendung und der sie bedienende Benutzer mit unterschiedlich breiten Adressen in Berührung.

Mit IPv6 entfällt die Notwendigkeit von NAT und somit ist wieder die ursprüngliche Idee von der unmittelbaren Adressierbarkeit aller Teilnehmer gegeben. Dies hat in der Praxis allerdings auch Nachteile, die im Abschnitt 3.1 angedeutet werden. Das beschriebene Kommunikationsverfahren geht von einem vorherrschenden IPv4 Netz aus, in dem die NAT Bürde gegeben ist und besondere Anstrengungen unternommen werden müssen, um trotzdem hinreichend kommunizieren zu können.

Die Arbeit ist, soweit möglich, IP versionsneutral oder betrachtet aus Implementierungssicht die Handhabung beider Versionen separat. Wird das DNS[5] verwendet, entfällt das unmittelbare Hantieren mit numerischen IP Adressen aus Benutzer- und unter Umständen auch Anwendungssicht komplett. Das DNS wird im Kapitel 4 nochmals näher erwähnt, wenn es um alternative Tunnelungsansätze geht.

Weitere, auf das Internet Protokoll aufbauende Schichten, werden im Kapitel 2 vorgestellt. Sie werden im Rahmen dieser Arbeit insbesondere auf Möglichkeiten zur Datentunnelung untersucht.

[3] **I**nternet **A**ssigned **N**umbers **A**uthority
[4] **N**etwork **A**ddress **T**ranslation
[5] **D**omain **N**ame **S**ystem

### 1.1.1 Offene Kommunikation

Sprechen zwei Menschen miteinander, tauschen sie Wörter und Sätze unter Beachtung grammatikalischer Regeln aus. Durch das Betonen einzelner Wörter oder Buchstaben wird die Bedeutung des Gesagten beeinflusst. Auch Pausen zwischen Sätzen oder Satzteilen können die Semantik zusätzlich verändern. Wird das Gespräch von Dritten mitangehört, so wird diesen im Normalfall dieselbe Information offenbart, die unter den beiden Gesprächspartnern ausgetauscht wird. Der bereits thematisierte Störeinfluss (Abbildung 1.1) aufgrund der menschlichen Wahrnehmung und Umwelteinflüssen, sei an dieser Stelle wegabstrahiert.

In dieser Situation findet eine offene Kommunikation in Form eines Zwiegesprächs statt, das von Dritten mitangehört wird. In dieser Form können die Gesprächspartner zwar grundsätzlich Informationen austauschen aber immer unter der Prämisse, dass das gleichzeitige Mitinformieren nicht unmittelbar Beteiligter kein Problem darstellt. Bei einem Gespräch zweier Freunde in der U-Bahn beispielsweise, wäre dies für gewöhnlich der Fall. Dritte sind hier unbekannte, anonyme Personen, die passiv, unbewusst und potentiell desinteressiert das Gehörte aufnehmen.

Konstruiert man nun aber eine Szene aus einem Agentenfilm, in dem der Protagonist ein Funkgerät bedient und seinem Kollegen mitteilt, dass die überwachte Person das Haus verlässt, dann tut der Agent gut daran, von der feindlichen Überwachung der Funkfrequenz auszugehen und entsprechend Vorsicht walten zu lassen. Dritte sind in diesem Fall bewusst mitlauschende Personen, deren Informierung vermieden werden soll. Der Agent improvisiert oder vereinbart deshalb mit seinem Kollegen vorab eine Art Bildsprache, die nur mit entsprechendem Kontextwissen interpretiert werden kann. Die Statusmeldung über die Aktivität der überwachten Person könnte also lauten: *„Der Vogel hat das Nest verlassen“*. Die Kommunikation erfolgt damit grundsätzlich weiterhin offen, aber die Semantik des Gesagten ist verschleiert.

Unter gefährlichen Umständen würde der Agent wohl zu technisch raffinierteren Verschleierungsmethoden greifen. Die Benutzung einer im Funkgerät eingebauten Verschlüsselung beispielsweise, würde dazu führen, dass der Inhalt des Gesprächs von Dritten ohne Dechiffriermöglichkeiten gar nicht mehr wahrgenommen werden kann. Dennoch wird in einem gewissen Sinne offen kommuniziert, weil der Kommunikationsvorgang an sich – aufgrund der Kryptierung in Form von unverständlichem Rauschen – von Außenstehenden dokumentiert werden kann.

Umgemünzt auf den Informationsaustausch in Computernetzwerken, entspräche das Abrufen einer Website einen Link wie http://www.example.com dem ersten Szenario, also das von möglicherweise mitlauschenden Dritten nachvollziehbare Übertragen von Daten. Mit welchen technischen Hilfsmitteln solch ein Abhörvorgang bewerkstelligt werden kann, wird im Kontext der

Analyse des dort beschriebenen Datentunnelungskonzeptes im Kapitel 2 gezeigt. Als Analogie für die aktivierte Verschlüsselung im Funkgerät, drängt sich SSL/TLS[6] auf, eine Verschlüsselungsschicht für das gesicherte Übertragen von Websites. Offene Kommunikation im Allgemeinen und Datenverschlüsselung im Besonderen, sind jedoch im Hinblick auf das Kernthema dieser Arbeit periphäre Elemente und werden deshalb inhaltlich nur gestreift.

Neben offener Kommunikation – sei sie verschlüsselt oder nicht – ist bezüglich des Datentunnelungskonzeptes primär die *verdeckte* Kommunikation interessant.

### 1.1.2 Verdeckte Kommunikation

Während der Vorgang der Informationsübermittlung zwischen zwei Akteuren bei offener Kommunikation von Dritten dokumentiert werden kann, soll genau dies bei verdeckter Kommunikation verhindert werden. Es geht in diesem Zusammenhang aber wohlgemerkt nicht um das grundsätzliche Verbergen sämtlicher Anstrengungen des Informationsaustausches. Das Szenario eines Spions, der über tote Briefkästen mit einem Mittelsmann korrespondiert, wäre beispielsweise keine gute Analogie zur Interpretation des Begriffs der verdeckten Kommunikation, die hier von Belang ist. Vielmehr geht es um einen legitimen und offenen Kommunikationskanal (nach der Interpretation aus Abschnitt 1.1.1), der als Träger für *scheinbar* unbedeutende Information genutzt wird. Es liegt also eine Phantomkommunikation vor, in die für Dritte eine nicht wahrnehmbare Zusatzinformation eingebettet ist.

Dabei handelt es sich um ein „steganographisches“ System. Der Begriff der Steganographie stammt aus dem Griechischen und bedeutet „verdecktes Schreiben“. Er wurde von Johannes Trithemius geprägt, dessen Werk [55] zu Beginn des 17. Jahrhunderts erschien. In Abgrenzung zur Kryptographie, deren Ziel die (offensichtliche) Chiffrierung einer Nachricht ist, geht es bei der Steganographie um das Verstecken einer Nachricht [21]. Beides kann freilich kombiniert werden, sodass zuerst die Verschlüsselung der zu übertragenden Information vorgenommen und danach der Chiffretext in einem Wirtsmedium versteckt wird.

Als anschauliches Beispiel für das Konzept der Steganographie in einem technischen Kontext, verdeutlicht Abbildung 1.2 das Verstecken eines Bildes innerhalb eines Trägerbildes. Dazu kann die Software *OpenStego*[7] verwendet werden. Die zwei zu verrechnenden Grafiken liegen in einer Farbtiefe von jeweils 24 Bit pro Pixel vor. Der Algorithmus zum Zusammenfügen der Bilder manipuliert dabei die niederwertigsten Bits jedes Farbkanals eines Pixels des Trägerbildes. Werden von einem $1024 \times 768$ Pixel großen RGB Bild mit 24 Bit Farbtiefe die drei niederwertigsten Bits (also insgesamt 9 Bits pro Pixel) verwendet, ist im Bild Platz für 884.736 Bytes an beliebigen Binär-

[6] **S**ecure **S**ocket **L**ayer/**T**ransport **L**ayer **S**ecurity [9]

[7] http://openstego.sourceforge.net

(a) (b) skaliert (c)

**Abbildung 1.2:** In das $1852 \times 1489$ Pixel große Landschaftsbild (a) wurde das $903 \times 613$ Pixel große Wurstbild (b) versteckt. Das Ergebnisbild (c) ist mit freiem Auge nicht von (a) zu unterscheiden, obwohl es (b) vollständig enthält.

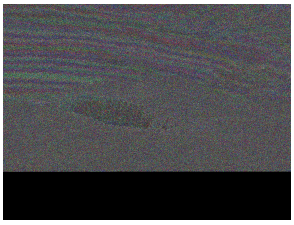


**Abbildung 1.3:** Normalisiertes Differenzbild aus Träger- und Ergebnisbild der Abbildung 1.2. Der untere schwarze Balken entspricht der ungenutzten Speicherkapazität im Trägerbild.

daten. Für das freie Auge sind Bitschwankungen dieser Größenordnung bei Fotos ohne direkten Vergleich nicht erkennbar. Zoomt und überlappt man das Quell- und Ergebnisbild zur genaueren Analyse in einem Bildbetrachtungsprogramm, macht sich die Einbettung durch dezentes Farbrauschen bemerkbar. Abbildung 1.3 zeigt das frei normalisierte Differenzbild des Träger- und Ergebnisbildes aus Abbildung 1.2.

Das in dieser Arbeit thematisierte steganographische Verfahren ist zwar nicht von solch visueller Natur, aber die Motivation ist dieselbe. Beim Konzept der Datentunnelung im netzwerktechnischen Sinne, ist das Trägermedium nun keine Bilddatei sondern ein Kommunikationsprotokoll (siehe Kapitel 2). Eine End-zu-End Verbindung bildet unter Verwendung eines solchen Protokolls logisch einen *Kanal* (engl. „channel“). Je nachdem ob das Protokoll einen datenstrom- oder paketorientierten Informationsfluss vorsieht, werden die versteckt zu übertragenden Daten unter Berücksichtigung des protokollspezifischen Formats beispielsweise in unscheinbare Nutzdaten integriert, in zeitliche Schwankungen des Paketaustausches transformiert oder in Form manipulierter header Felder einzelner Pakete repräsentiert.

### Definitionen

In der Literatur finden sich je nach Kontext verschiedene Bezeichnungen für einen *verdeckten Kanal*. Steffen Wendzel stellte diese in [58] zusammen:

- Steganographic channel
- Hidden channel
- Covert channel
- Subliminal channel
- Side channel
- Tunnel

Diese Begriffe sind nicht als Synonyme zu verstehen sondern unterscheiden sich in ihrer exakten Bedeutung. Ein *steganographischer* Kanal („steganographic channel“) ist ein Metabegriff für einen Kommunikationskanal in dem steganographische Methoden Anwendung finden.

In einem *verdeckten* Kanal („hidden channel“ oder „covert channel“) wird Information verborgen transportiert, wobei zur Erreichung dieses Ziels möglicherweise auf steganographische Mittel zurückgegriffen wird. Moskowitz argumentiert in [31], dass bei genauer Betrachtung zwischen einem steganographischen und einem verdeckten Kanal ein Unterschied bestehe. Dieser wird zum einen damit begründet, dass der covert channel per se versteckt ist, während der steganographic channel nur dann existiert, wenn seine Existenz verborgen ist. Zum anderen spielt bei der abstrakten Betrachtung des covert channel die Dauer des Datentransports keine Rolle, während diese beim steganographic channel durch den Typ des Trägermediums limitiert wird.

Verschiedene Definitionen der Begrifflichkeit finden sich in [15]:

- A communication channel is covert if it is neither designed nor intended to transfer information at all [24].
- A communication channel is covert (e.g., indirect) if it is based on “transmission by storage into variables that describe resource states” [44].
- Covert channels “will be defined as those channels that are a result of resource allocation policies and resource management implementation” [17].
- Covert channels are those that “use entities not normally viewed as data objects to transfer information from one subject to another” [22].

Gustavus Simmons bezeichnet einen Kanal mit mathematisch beweisbaren, steganographischen Eigenschaften innerhalb eines kryptographischen Systems als „subliminal channel“. Die Idee dahinter ist die Schaffung eines Kanals, der genauso schwer zu entdecken, wie der zugrunde liegende Verschlüsselungsalgorithmus zu knacken ist [27, 48].

Im Gegensatz dazu ist der „side channel“ ein Kanal, der vom Erzeuger oder Betreiber eines übergeordneten Systems unbeabsichtigt und potenti-

ell unerwünscht Informationen preisgibt. Kanäle dieser Art haben einen im Kontext dieser Arbeit irrelevanten, physikalischen Charakter. Es geht beispielsweise um das Analysieren von Zuständen eines kryptographischen Algorithmus anhand Zeitmessungen oder der elektrischen Leistungsaufnahme von Mikroprozessoren [4]. Auf diese Art observierte Daten sind dann aus einem nicht systemimmanenten *Seitenkanal* beziehbar.

Der etwas vage Begriff des Tunnels wird in akademischen Kreisen seltener verwendet, dafür ist er in der Sicherheitsszene und generell bei praxisbezogenen Themen umso populärer. So wird für Softwareimplementierungen eines verdeckten Kanals (siehe Kapitel 4) gerne die Bezeichnung „Tunnel“ im Projektnamen verwendet. Auch im Titel dieser Arbeit findet sich der vergleichsweise saloppe Begriff *Datentunnelung*, der neben der persönlichen Präferenz des Verfassers, auch die Praxisnähe der technischen Inhalte der folgenden Kapiteln widerspiegeln soll.

Während ein Tunnel im herkömmlichen Sinne beispielsweise im Straßenverkehr als Unterführung, zur Abkürzung von unwegsamem Gelände begriffen wird, hat im netzwerktechnischen Sinne ein Tunnel im Allgemeinen eher den gegenteiligen Effekt. Da er eine Indirektion darstellt, Information also nicht mehr unmittelbar zum Ziel fließt, sondern über einen separaten Kanal transportiert wird, ist in der Regel mit einer Verringerung der Bandbreite und Erhöhung der Latenz im Vergleich zu einer Direktverbindung zu rechnen.

### Klassifizierung

Einem verdeckten Kanal liegt ein bestimmtes Implementierungskonzept zugrunde. Obwohl es vom theoretischen Standpunkt aus keinen fundamentalen Unterschied gibt [10, S. 80], wird in der Literatur zwischen zwei Arten unterschieden [15, 2.2.1]:

- Storage channel
- Timing channel

In einem „storage channel“ wird Information ausgetauscht, indem Speicherstellen direkt oder indirekt beschrieben und gelesen werden. Das in dieser Arbeit vorgestellte Tunnelungskonzept verwendet einen Kanal dieser Art.

Der „timing channel“ kodiert Information zum Beispiel durch Variation von Antwortzeiten. Die im Trägermedium transportierten Daten bleiben dabei unberührt. Im Allgemeinen ist die Bandbreite der übertragbaren Information bei solch einem Kanal im Vergleich zum storage channel geringer.

# Kapitel 2

# Kommunikationsprotokolle

Es entfällt die Notwendigkeit, ein Interface zur Mensch-Maschine-Interaktion zu beschreiben, weil die Kommunikation der beschriebenen Softwarekomponente lediglich zwischen Computerprogrammen stattfindet. Damit sind ausschließlich wohldefinierte, digitale Protokolle erforderlich, die optimalerweise ohne Interpretationsspielraum ausgelegt sind.

Die Encyclopædia Britannica[1] definiert den Begriff des *Protokolls* wie folgt:

> Protocol, in computer science, a set of rules or procedures for transmitting data between electronic devices, such as computers. In order for computers to exchange information, there must be a preexisting agreement as to how the information will be structured and how each side will send and receive it. (...)

Nachdem das *Bit* in der Netzwerktechnik die fundamentale Einheit der Informationsübertragung ist, müssen miteinander vernetzte Geräte mit jeweils einem Sende- und Empfangsteil auf unterster Ebene mit binären Symbolen kommunizieren. Empfängt ein Kommunikationsteilnehmer Daten, so kommen diese in einem Strom aus Bits an, welche schließlich in Oktets zu *Bytes* zusammengefasst werden. Läge bei der Interpretation des eingehenden Datenstroms ein Versatz vor, indem beispielsweise das erste Bit fälschlicherweise übersprungen werden würde, dann schlüge die Interpretation sämtlicher nachfolgender Bits und Bytes fehl. Abbildung 2.1 illustriert die Problematik.

Aus diesem Grund ist die strikte Einhaltung eines Protokolls für robuste Datenübermittlung notwendig. Bei der Übertragung einer Bilddatei beispielsweise von einem Computer zum anderen, muss die Information in irgendeiner Form zerlegt und in einem Bitstrom serialisiert über eine Datenleitung geschickt werden. Diese Transformation der Darstellungsweise einer Information muss auf flexible Art und Weise geschehen, damit beliebige Daten transportiert werden können. Um dies zu ermöglichen, wird ein *Schalen-*

[1] http://www.britannica.com/EBchecked/topic/410357/protocol

(a) 0100100001100001011011000110110001101111001000010000...
(b) 01001000 01100001 01101100 01101100 01101111 00100001 000...
(c) 10010000 11000010 11011000 11011000 11011110 01000010 00...

**Abbildung 2.1:** Der Bitstrom (a) wird in Oktets gruppiert (b), wobei die entstehenden Bytes dem Text „Hallo!“ entsprechen. Bei (c) beginnt die Gruppierung fälschlicherweise beim zweiten Bit, was eine völlig andere Textinterpretation zur Folge hat: „ ÂØØÞB“. Als Kodierung sei Latin-1[2]angenommen.

| | |
|---|---|
| Höhere Schichten | Schicht 5-7 |
| TCP, UDP | Schicht 4 |
| Internet Protocol (IP), ICMP | Schicht 3 |
| Ethernet | Schicht 2 |
| Bitübertragung | Schicht 1 |

**Abbildung 2.2:** Skizze des OSI Referenzmodells mit Fokus auf die in dieser Arbeit relevanten Schichten.

oder *Schichtenmodell* verwendet, um eine hierarchische Schachtelung unterschiedlicher Protokolle realisieren zu können. Diese bauen aufeinander auf und sind so konzipiert, dass einzelne Schichten ausgetauscht werden können, ohne dass übergeordnete davon beeinträchtigt werden. Dieses Konzept wird durch das siebenschichtige OSI[3] Referenzmodell in [59, S. 28] beschrieben.

Abbildung 2.2 illustriert das Modell vereinfacht, wobei sich die Kernthematik dieser Arbeit auf Schicht 3, der *Vermittlungsschicht* (engl. „network layer“) bewegt. Die Bitübertragung auf unterster Ebene fand bereits Erwähnung und ist nicht weiter relevant. Die darüber liegende *Sicherungsschicht* wird in diesem Kapitel innerhalb binärer Datenpaketauszüge noch einmal auftauchen. Das *Internet Protokoll* der Version 4 und 6 wird inklusive des *ICMP* in seinen Grundzügen in den folgenden Abschnitten beschrieben. Bezüglich des Verhältnisses des IP zum ICMP heißt es in [37, S. 1]:

> ICMP, uses the basic support of IP as if it were a higher level protocol, however, ICMP is actually an integral part of IP, and must be implemented by every IP module.

Auf übergeordnete Protokolle der *Transportschicht* (engl. „transport layer“) wie dem *TCP* oder *UDP* wird bei den verwandten Tunnelungsansätzen im Kapitel 4 näher eingegangen.

[3] **O**pen **S**ystems **I**nterconnection Reference Model
[2] ISO/IEC 8859-1:1998

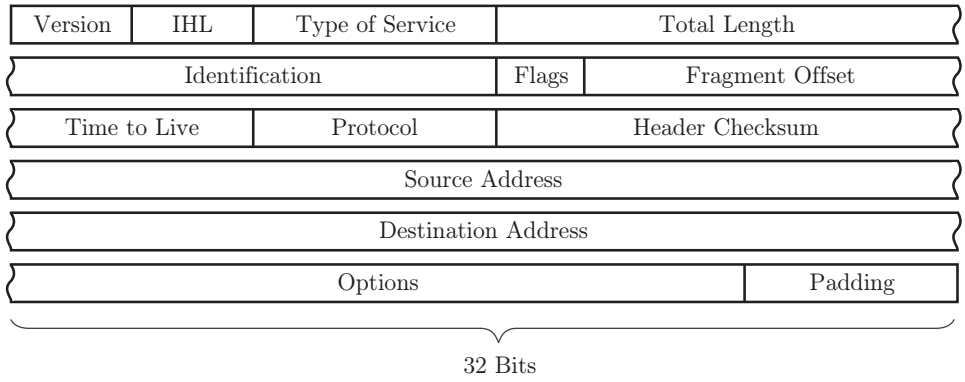



**Abbildung 2.3:** Vollständiges header Format des Internet Protokolls der Version 4 inklusive Options-, und Paddingfeld.

## 2.1 Internet Protocol

Das Internet Protokoll wird in zwei Versionen auf der Vermittlungsschicht verwendet. Es dient zur Übertragung von Datenpaketen, die anhand einer Quell- und Zieladresse mit versionsabhängig fest definierter Länge zwischen Kommunikationsteilnehmern in einem Computernetzwerk ausgetauscht werden. Außerdem sieht das Protokoll Mechanismen für die Fragmentierung von übermäßig großen Paketen, sowie deren erneute Zusammenfügung auf der Empfängerseite vor [38, S. 1]. Diese zwei grundlegenden Konzepte der *Adressierung* und *Fragmentierung* werden zusammen mit einigen Zusatzparametern in der Praxis auf eine Weise realisiert, die insbesondere im Hinblick auf Möglichkeiten zur Ausnutzung zwecks Steganographie interessant ist.

### 2.1.1 IPv4

Abbildung 2.3 skizziert das header Format eines IPv4 Pakets, wie es in [38, S. 11] definiert ist. Das Format enthält optionale und nicht optionale Datenfelder, wobei *Options* und *Padding* zu den optionalen Feldern gehören. Die Felder vor der Quell- und Zieladresse sind in jedem Paket vorhanden und teilweise aus historischen Gründen heutzutage redundant. Unter anderem diese Redundanzen können für steganographische Zwecke ausgenutzt werden. Die folgenden header sind bezüglich der Möglichkeiten zur verdeckten Kommunikation interessant [33, S. 4–5]:


Das Feld zur Angabe des Servicetyps ist 8 Bit breit und war für die Unterstützung eines *Quality of Service* Konzepts vorgesehen. In der Praxis wird dieses Feld aber kaum verwendet und hat normalerweise den Wert 0. Seit

1981, als [38] erschien, hat die Interpretation dieses Felds Änderungen erfahren. So wird ihm durch [42] eine spezielle Bedeutung bezüglich Flusskontrolle erteilt. Abgesehen von Sonderfällen in denen der Wert von Netzwerkkomponenten tatsächlich interpretiert wird, können diese 8 Bits zur Etablierung eines verdeckten Kanals benutzt werden [33, S. 4].

Die Verwendung des Feldes (nach der ursprünglichen Definition in [38]) wird in [16, S. 31], mit speziellem Augenmerk auf die zwei als „reserviert" definierten Bits, beschrieben. Der Gebrauch von Speicherstellen, die „für zukünftige Verwendung" freigehalten werden und deren Werte daher keine definierte Semantik haben, hat Vor- und Nachteile. Abschnitt 5.1 im Kapitel 5 erläutert die Problematik.


Das Identifikationsfeld wird für den Fragmentierungsfall benötigt. Muss ein Datenpaket aufgrund seiner Größe in kleinere Pakete aufgeteilt werden, soll anhand dieser 16 Bit breiten Identifikationsnummer die korrekte Zuordnung von Fragmenten ermöglicht werden.

Für die Berechnung eines gültigen Werts für dieses Feld müssen Faktoren, wie Eindeutigkeit und Unvorhersagbarkeit miteinbezogen werden. Solch ein Berechnungsalgorithmus kann allerdings auf eine Weise konstruiert werden, dass innerhalb des Bitmusters der erzeugten Nummer steganographische Information transportiert werden kann [33, S. 4].

Problematisch bei der Nutzung des Identifikationsfeldes ist, dass zur Tunnelung von Daten eine entsprechende Menge an fragmentierten Paketen erzeugt werden muss.


Das 3 Bit breite Signalisierungsfeld enthält wieder ein reserviertes Bit, sowie zwei für Fragmentierungsinformationen. In [2] ist die Möglichkeit eines Redundanzzustandes dieser beiden Bits beschrieben, der zur Bildung eines (mit einem einzelnen Bit sehr schmalbandigen) verdeckten Kanals verwendet werden kann.


Für den Fragmentierungsoffset sind 13 Bit vorgesehen. Er dient im Fragmentierungsfall zur Bestimmung der Position des Fragments im ursprünglichen, zusammengesetzten Paket. Über die Beeinflussung der Fragmentierungscharakteristik, können die Offsetwerte variiert werden. Damit ist wiederum die Kodierung von Information möglich [33, S. 5].

Die fragmentbasierten Techniken haben gemeinsam, dass vorsätzlich Fragmentierung in potentiell unüblichem Maß hervorgerufen werden muss.

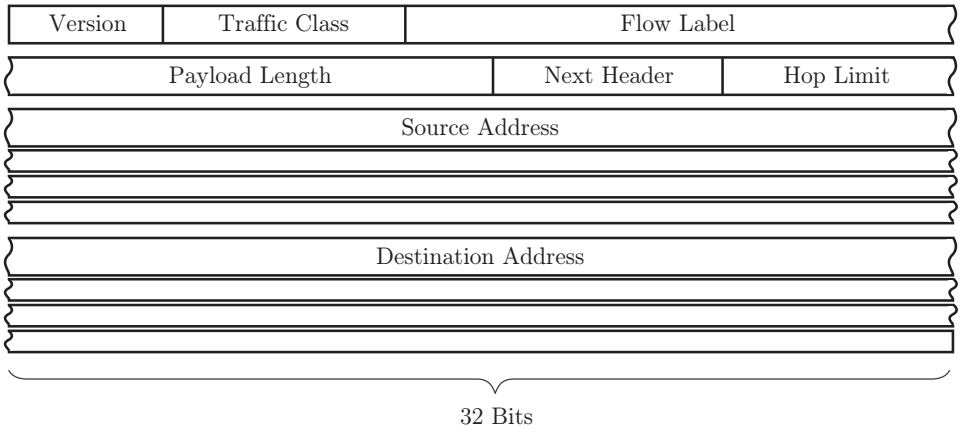



**Abbildung 2.4:** Basis header Format des Internet Protokolls der Version 6.


Der IPv4 header kann um spezielle Optionsfelder erweitert werden, was jedoch in regulärem Datenverkehr kaum stattfindet. Eine mögliche Verwendung, der in [38, S. 16] als *Internet Timestamp* bezeichneten Option, wird ebenfalls in [16, S. 31] erwähnt. So kann beispielsweise, durch das Senden eines Pakets mit einem geradzahligen oder ungeradzahligen Zeitstempel, binäre Information steganografisch transportiert werden [33, S. 5].

Techniken dieser Art funktionieren freilich nur, wenn Netzwerkkomponenten die eingefügte Information auch weiterleiten und nicht verwerfen, was in Firewallkonfigurationen gerne standardmäßig bezüglich sicherheitstechnisch fragwürdiger IP Optionen wie zum Beispiel *source routing*, der Fall ist.

### 2.1.2 IPv6

In Abbildung 2.4 ist das im Vergleich zum IPv4 header vereinfachte IPv6 header Format zu sehen. Definiert in [8, S. 4], ist es als Nachfolgeformat aufgrund der vereinfachten Struktur für Netzwerkknoten leichter zu interpretieren und benötigt weniger Bandbreite. Das aus IPv4 bekannte Konzept der *Optionen* wurde in Form von *header Erweiterungen* wiederverwendet und flexibler gestaltet. Auch der Servicetyp findet in Form der *traffic class* wieder Einzug. Bedingt durch die 128 Bit breiten IPv6 Adressen, benötigen die Quell- und Zielfelder im Vergleich zum IPv4 header den vierfachen Speicherplatz.

Auch im IPv6 header lässt sich Information verstecken. In [28] wird sehr detailiert auf unterschiedliche Möglichkeiten zur Etablierung verdeckter Kanäle eingegangen, sowohl innerhalb der Standard- als auch der optiona-

len Erweiterungsheaderfelder. Analog zu IPv4 ist die Benutzung der folgenden IPv6 Felder zur steganographischen Informationsübertragung denkbar [28, S. 5–6]:


Die Klasse eines IPv6 Pakets wird dazu benutzt, um in einem *Quality of Service* Schema unterschiedliche Prioritäten bei der Behandlung von Datenverkehr zu unterstützen. Es kann als 8 Bit breiter verdeckter Kanal benutzt werden, wobei zu beachten ist, dass Netzwerkknoten laut RFC die Veränderung von Bits dieses Feldes je nach Bedarf gestattet ist. Außerdem können eingeschleuste Daten zu Störungen des Pakettransports führen, sofern diese von Netzwerkknoten als legitime traffic class Werte missinterpretiert und verarbeitet werden [28, S. 5].


Die 20 Bit breite Flussmarke dient zur Gruppierung von Paketen, die logisch in einem „Fluss“ zusammengefasst werden sollen. Pro eindeutiger und zufällig errechneter Flussmarke muss dieselbe Quell- und Zieladresse verwendet werden. Das Datenfeld kann entweder als ganzes für die Beherbergung von Information benutzt, oder der Algorithmus eines Pseudozufallsgenerators auf eine Weise modifiziert werden, dass Daten steganographisch eingespeist werden können [28, S. 5].


Die Größe des Anhangs eines IPv6 Pakets wird durch ein 16 Bit breites Längenfeld spezifiziert. Der header selbst (ohne Erweiterungsheader) ist in der Größenangabe nicht enthalten. Durch die Änderung der Längeninformation und damit der Vergrößerung des Pakets, kann der Sender zusätzliche Information am Ende des Datenpakets anfügen [28, S. 5].


IPv6 header können mit Erweiterungsheader versehen werden. Laut Spezifikation müssen Netzwerkknoten, die das Paket auf seinem Weg zum Ziel passiert, alle unbekannten header ignorieren. Somit kann ein frei erfundener header eingefügt und an seiner Stelle beliebige Information transportiert werden. Kann die Implementierung des IPv6 Protokolls auf der Empfängerseite den ihr unbekannten header nicht interpretieren, wird eine entsprechende Fehlermeldung per ICMP gesendet (siehe Abschnitt 2.2). Dieser Umstand kann ebenfalls zur verdeckten Kommunikation ausgenutzt werden [28, S. 6].


Das hop Limit definiert die maximale Anzahl an Weiterleitungen des Pakets zwischen Netzwerkknoten. Jeder Knotenpunkt, über den das Paket geleitet wird, verringert diesen Zähler um eine Einheit. Sobald der Zählstand Null erreicht, wird das Paket als verwaist erachtet und verworfen.

Über eine spezielle Strategie bezüglich des Setzens von hop Limit Werten, kann auch hier Steganographie zum Einsatz kommen. Durch die Wahl eines anfänglichen Limits und der nachträglichen Variation durch Addition oder Subtraktion eines Deltawerts, lässt sich ein schmalbandiger verdeckter Kanal konstruieren [28, S. 6].

## 2.2 Internet Control Message Protocol

ICMP ist als Bestandteil des Internet Protokolls für reibungslose Kommunikation als Signalisierungsmechanismus vorgesehen. Es ist in vielerlei Hinsicht kein „verlässliches“ Protokoll, sondern ein simples Werkzeug zur Netzwerkdiagnose und Melden von Fehlern oder andere Anomalien, um anhand der damit gewonnenen Informationen effizientere Kommunikationsabläufe gestalten zu können. ICMP Nachrichten können in überlasteten Netzwerken oder fehlerhaft arbeitenden Netzwerkkomponenten verloren gehen, ohne dass dies dem Absender in irgendeiner Form bekanntgegeben wird. Insofern gilt außerdem die Regel, dass auf ein ICMP Paket, kein weiteres ICMP Paket als Antwort folgen darf, um eine unendliche Propagierung von „ICMP Nachrichten über ICMP Nachrichten“ von vornherein auszuschließen [37, S. 1]. Das Echokonzept (siehe Abschnitt 2.2.1) ist davon freilich ausgenommen.

Ein weiteres Problem ist die Möglichkeit des Fingierens von ICMP Nachrichten durch einen Angreifer im Netzwerk. Es gibt zumindest im Falle von IPv4 keine protokollimmanenten Authentifizierungsmechanismen. IPv6 schafft mit dem „authentication header“ [23] diesbezüglich Abhilfe – sofern er verwendet wird.

Es stellt sich nun die Frage, ob man bei diesen Problemen und insbesondere durch den Umstand, dass das Verlorengehen von ICMP Nachrichten ohne weiteres geschehen kann, auf ICMP nicht gänzlich verzichten könnte. Grundsätzlich ist zumindest in einem IPv4 Netzwerk eine komplette Filterung von ICMP möglich, ohne dass regulärer Datenverkehr dadurch in signifikantem Maße gestört werden würde. Es gibt jedoch Situationen, in denen eine Verbindung ohne ICMP Benachrichtigungen nicht aufgebaut werden kann, obwohl das Ziel durchaus erreichbar wäre. Wenn beispielsweise im „flags“ Feld des IPv4 header ein spezielles Bit zur Verhinderung von Fragmentierung gesetzt ist [38, S. 13], das Paket aber aufgrund seiner Größe nur in fragmentierter Form weitergeleitet werden kann, dann ist nur per ICMP Signalisierung das Problem erkenn- und lösbar.

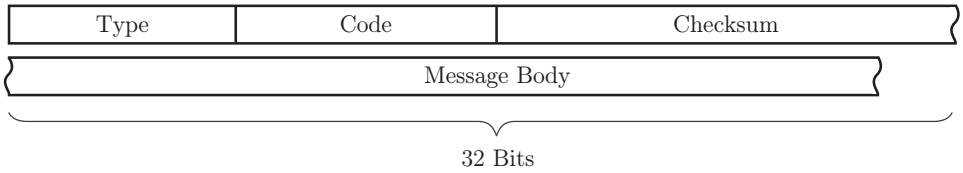



**Abbildung 2.5:** Allgemeines header Format eines ICMPv4 und ICMPv6 Pakets.

### 2.2.1 Format

Das grundlegende Format eines ICMP headers ist für die IPv4 und IPv6 Version gleich und in Abbildung 2.5 ersichtlich. ICMP wird genauso wie Datenpakete aus höheren Schichten innerhalb eines IPv4 Pakets gekapselt und im Fall von IPv6 als Erweiterungsheader eingetragen. Das „Protocol“ Feld im IPv4 header (vgl. Abbildung 2.3) bzw. das „Next Header“ Feld im IPv6 header (vgl. Abbildung 2.4) trägt für ICMPv4 [37] die von der IANA[4] vergebene Protokollnummer 1 und für ICMPv6 [6, S. 3] die Nummer 58.

Jedem Nachrichtentyp ist eine Zahl zugeordnet, die im acht Bit breiten „Type“ Feld eingetragen wird. Für ICMPv4 sind elf unterschiedliche Nachrichten vorgesehen [37, S. 20]. Bei ICMPv6 ist eine Gliederung in zwei Kategorien gegeben [6, S. 4]:

- Fehlermeldungen
- Informelle Nachrichten

Insgesamt sind darin neben sechs konkreten, auch einige für private und experimentelle Zwecke, sowie für „zukünftige Verwendung“ reservierte Typen definiert. Für diese Arbeit sind im Hinblick auf die beschriebene Tunnelungstechnik genau zwei davon interessant: Die *Echo Anfrage* („echo request“) und *Echo Antwort* („echo reply“). Diese fallen in die informelle Kategorie und dienen eigentlich Diagnosezwecken. Das Konzept ist simpel: Ein Teilnehmer im Netzwerk, der eine Echo Anfrage bekommt, schickt eine Echo Antwort zurück. Der Auszug 2.1 zeigt dieses Verfahren in der Praxis. Über eine Kommandozeile wird das ursprünglich 1983 von Michael Muuss[5] entwickelte und heute als Standardwerkzeug geltende Dienstprogramm namens *ping* angewiesen, einem entfernten Rechner eine Echo Anfrage zu senden. Der Begriff des Echos wurde offensichtlich wegen des akustischen Phänomens der Reflexion von Schallwellen gewählt, weil es dem Sender durch den reflektierten Schall möglich ist, auf die Existenz eines Objektes zu schließen. In einem Computernetzwerk kann nach diesem Prinzip und mithilfe des ICMP festgestellt werden, ob ein Teilnehmer unter einer bestimmten Adresse verfügbar und eventuell ansprechbar ist. Der Umkehrschluss ist allerdings nicht zulässig.

[4] http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml
[5] http://ftp.arl.army.mil/~mike/ping.html

```
$ ping6 ipv6.google.com
PING ipv6.google.com(2a00:1450:8001::93) 56 data bytes
64 bytes from 2a00:1450:8001::93: icmp_seq=1 ttl=55 time=14.2 ms
64 bytes from 2a00:1450:8001::93: icmp_seq=2 ttl=55 time=14.3 ms
64 bytes from 2a00:1450:8001::93: icmp_seq=3 ttl=55 time=14.6 ms
64 bytes from 2a00:1450:8001::93: icmp_seq=4 ttl=55 time=14.3 ms
64 bytes from 2a00:1450:8001::93: icmp_seq=5 ttl=55 time=14.7 ms

--- ipv6.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 14.256/14.447/14.747/0.248 ms
```

**Auszug 2.1:** Das Dienstprogramm *ping* (oder wie in diesem Fall ein IPv6 Äquivalent *ping6*) steht für Diagnosezwecke in jedem gängigen Betriebssystem zur Verfügung.

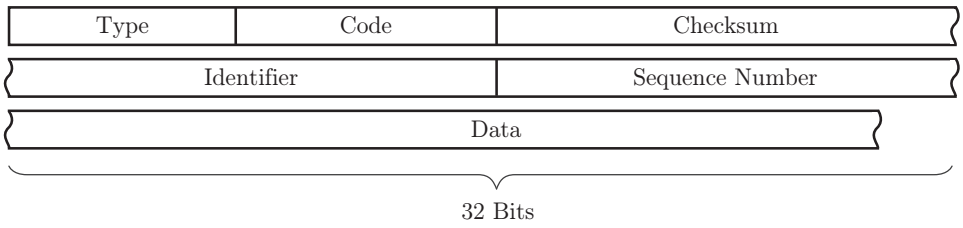



**Abbildung 2.6:** Spezifisches header Format für Echo Anfragen und Antworten. Das Layout ist dasselbe für ICMPv4 [37, S. 14] und ICMPv6 [6, S. 13–14]).

Ein Teilnehmer, der nicht auf Echo Anfragen reagiert, muss nicht notwendigerweise offline sein, sondern kann beispielsweise so konfiguriert sein, dass er ICMP Anfragen dieser Art verwirft und ignoriert.

Dies kann insbesondere zur Abwehr von Angriffen so eingestellt worden sein. ICMP kann für eine Reihe von Aktivitäten missbrauch werden, die dem anvisierten Computersystem schaden und zu einem „Denial of Service“ (DoS) führen können [12]. Dies ist eine Angriffsform, bei der die Terminierung der vom Server angebotenen Dienste beabsichtigt ist.

Das „Code“ Feld enthält bei manchen Nachrichtentypen eine zusätzliche, typspezifische Information. Es wird im Echo Fall nicht genutzt und muss laut Spezifikation Null sein [37, S. 8]. Die Prüfsumme im „Checksum“ Feld, wird zur Integritätsprüfung beim Empfänger einer ICMP Nachricht verwendet. Stellt dieser fest, dass die errechnete Prüfsumme des empfangenen Pakets nicht mit der im header eingetragenen übereinstimmt, dann wird das Paket als beschädigt erachtet und verworfen.

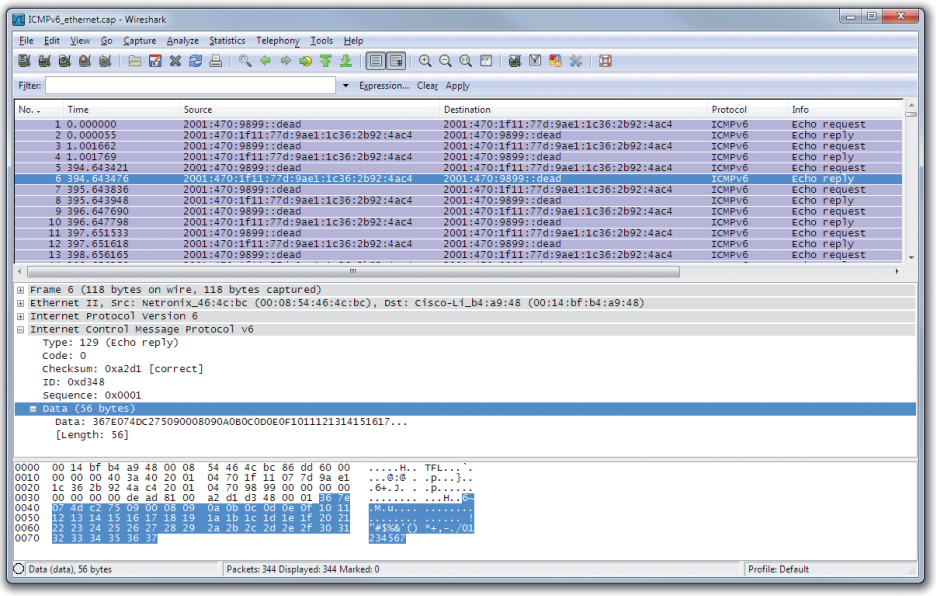



**Abbildung 2.7:** Anwendungsfenster des Programms *Wireshark*, das zum Aufzeichnen und Analysieren von Datenverkehr aus einem Computernetzwerk dient.

Abbildung 2.6 zeigt das Format des Echo Nachrichtentyps. Es enthält ein 16 Bit breites „Identifier“ Feld, das zur Zuordnung einer Echo Antwort zu einer Anwendung dient, welche die entsprechende Anforderung zuvor gesendet hat. Da ICMP ein verbindungsloses Protokoll ohne Portinformation ist, findet auf diese Weise eine Zuordnung statt. Jede Programminstanz, die solche ICMP Nachrichten senden und empfangen möchte, muss folglich systemweit einen eindeutigen Identifizierungswert generieren. Da unter Windows und Unix basierenden Systemen jeder Ausführung eines Programms eine eindeutige Prozessnummer zugewiesen wird, liegt es nahe, diese als „Identifier“ zu verwenden.

Die ebenfalls 16 Bit breite Sequenznummer im „Sequence Number“ Feld wird nach jeder gesendeten Echo Anfrage inkrementiert. Ausgesandte Anfragen erhalten somit eine aufsteigende Nummerierung, die neben dem Identifizierungswert zur Programmzuordnung, die Assoziation einzelner Echo Anfragen mit Antwortpaketen ermöglicht. Die Sequenznummer ist inbesondere von Bedeutung, weil sowohl das ICMP Paket mit einer Anfrage, als auch das mit einer Antwort ohne weiteres verloren gehen oder aufgrund netzwerktechnischer Anomalien mehrfach ankommen kann. Außerdem besteht die Möglichkeit, dass beim Senden zweier Echo Anfragen die Antwort auf die zuerst gesandte später ankommt, als die Antwort auf die danach gesandte Anfrage.

Für die Analyse des Kommunikationsablaufs im Allgemeinen und dem

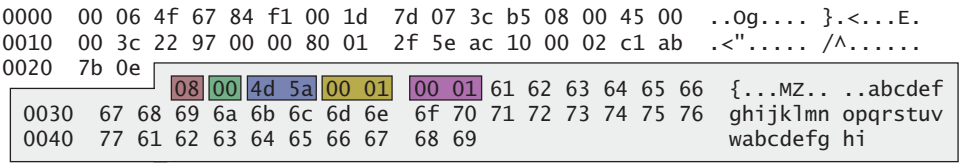



**Abbildung 2.8:** Byteweise Darstellung eines „echo request“ ICMPv4 Pakets (grau unterlegt), wie es von einem Windows System per „ping“ Kommando versendet wird. Bei den Bytes davor handelt es sich um einen Ethernet Rahmen und IPv4 header.

Verfolgen eines solchen Echo Wechsels im Besonderen, können sog. „packet sniffer“ verwendet werden. Es handelt sich dabei um Programme, die Schnittstellen des Betriebssystems zur Protokollierung des Netzwerkverkehrs ansprechen. Abbildung 2.7 zeigt *Wireshark*[6] unter Windows. Auf Unix basierenden Systemen ist das Programm *tcpdump*[7] populär, das sich über die Kommandozeile bedienen lässt. Mithilfe eines solchen Werkzeugs können Datenpakte Byte für Byte analysiert werden. Wireshark enthält eine Datenbank mit den Formaten unterschiedlichster bekannter und weniger bekannter Netzwerkprotokolle und kann deshalb über eine komfortable Benutzeroberfläche in seinem Analysefenster beispielsweise die header Felder eines ICMP Pakets interpretieren und aufgeschlüsselt darstellen.

Abbildung 2.8 zeigt eine Echo Anfrage in Form eines per Netzwerksniffer abgehörten Paketauszugs. Die Darstellung der Bytes erfolgt im Hexadezimalsystem. Außerdem befindet sich rechts eine Spalte mit den ASCII Entsprechungen der einzelnen Bytes. Beim abgebildeten Paket handelt es sich um eine Echo Anfrage, weil im Typfeld (rot) der Wert 8 steht [37, S. 14]. Sowohl die Identifizierungs- als auch Sequenznummer (gelb und magenta) haben den Wert 1, was zum einen bedeutet, dass es sich wohl um das erste Paket eines „ping“ Vorgangs handelt und zum anderen, dass die Prozessnummer offenbar nicht wie zuvor beschrieben herangezogen wurde. Unter Windows werden einer Echo Anfrage standardmäßig $n = 32$ Bytes an Testdaten angehängt. Diese beginnen mit dem Wert 97, der dem ASCII Buchstaben „a“ entspricht. Die Formel zur Wertbelegung für ein Byte $B$ an der Stelle $i$ ist gegeben durch

$$B_i = 97 + [(i - 1) \bmod 23], \quad \text{für } 1 \leq i \leq n. \tag{2.1}$$

Die auf diese Echo Anfrage erfolgte Antwort, ist in Abbildung 2.9 zu sehen. Die Pakete unterscheiden sich durch das Typfeld und folglich auch die Prüfsumme. Für eine Echo Antwort ist 0 als Wert für das Typfeld definiert [37, S. 14]. Das RFC schreibt ausdrücklich vor, dass die Testdaten aus einer Echo Anfrage unverändert in der Antwort zurückgeschickt werden

[6] http://www.wireshark.org
[7] http://www.tcpdump.org

```
0000  00 1d 7d 07 3c b5 00 06  4f 67 84 f1 08 00 45 00  ..}.<... Og....E.
0010  00 3c 4e ec 00 00 36 01  4d 09 c1 ab 7b 0e ac 10  .<N...6. M...{...
0020  00 02 00 00 55 5a 00 01  00 01 61 62 63 64 65 66  ....UZ.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                    wabcdefg hi
```

**Abbildung 2.9:** Die Antwort auf die in Abbildung 2.8 ersichtliche ICMP Nachricht.

```
0000  00 26 f1 e6 2d 00 00 06  4f 67 84 f4 08 00 45 00  .&..-... Og....E.
0010  00 54 00 00 40 00 40 01  b4 c9 c1 aa 87 7b c1 ab  .T..@.@. .....{..
0020  7b 0e 08 00 77 e8 6d df  00 01 07 62 c7 4d 54 84  {...w.m. ...b.MT.
0030  04 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15  ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25  ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35  &'()*+,- ./012345
0060  36 37                                            67
```

**Abbildung 2.10:** Eine Echo Anfrage wie in Abbildung 2.8, nur dass diese von einem Linux System aus versandt wurde und deshalb auch einen Zeitstempel enthält (türkis unterlegt).

müssen [37, S. 15]. Diese Einschränkung wird im Abschnitt 3.5 des Kapitels 3 beschriebenen Tunnelungsansatz übrigens absichtlich verletzt.

Auf Linux Systemen versendet die *ping* Implementierung aus der „iputils“ Sammlung standardmäßig $n = 56$ Bytes an Testdaten mit jeder Echo Anfrage. Abbildung 2.10 zeigt ein solches Paket, das von einer „Debian“ Linux Distribution[8] aus erzeugt wurde. Die Charakteristik der Wertbelegung unterscheidet sich von der unter Windows. Für den Wert des Bytes $B$ an der Stelle $i$ gilt, dass

$$B_i = (i - 1) \bmod 256, \quad \text{für } 9 \leq i \leq n. \tag{2.2}$$

Die ersten acht Bytes werden zur Speicherung eines Zeitstempels benutzt. Konkret wird eine `timeval` Struktur der folgenden Form verwendet:

```
1 struct timeval
2 {
3   time_t tv_sec; /* Sekunden */
4   suseconds_t tv_usec; /* Mikrosekunden */
5 };
```

Die Einbettung eines Zeitstempels durch den Sender macht zusammen mit dem Gebot, dass die Testdaten unverändert aus dem Anfragepaket in das Antwortpaket übernommen werden müssen, Sinn. Damit erspart sich der Sender das Mitführen einer Liste mit den gesendeten Paketen und den jeweiligen Sendezeiten dazu. Kommt ein Antwortpaket an, kann anhand der

[8] http://www.debian.org

Differenz des aktuellen Zeitstempels mit dem im Paket befindlichen, die Dauer der Datenübertragung ermittelt werden.

So wie im Fall der IP header, können auch Felder des ICMP headers als „covert storage channel“ verwendet werden. Konkret für Echo Pakettypen sind drei dafür geeignet:


Da für die Generierung eines Identifizierungswerts in [37, S. 15] keine konkreten Strategien vorgeschrieben sind, eignet sich dieses Feld als unauffälliger aber eher schmalbandiger verdeckter Kanal.


Für die Handhabung der Nummerierung einer Sequenz, gibt es ebenfalls keine zwingenden Vorschriften. Die standardmäßige Vorgehensweise ist jedoch das schrittweise Inkrementieren der Sequenznummer mit jeder gesendeten Echo Anfrage. Das 16 Bit breite Datenfeld kann folglich als solches zur Übertragung zweier Bytes verwendet werden oder – etwas unauffälliger – beispielsweise durch die Variierung von geraden und ungeraden Zahlen für einen steganographischen Informationstransport benutzt werden.


Das Datenfeld ist zu guter Letzt für diese Arbeit von zentraler Interesse. Es handelt sich um ein Feld variabler Größe, die sich aus der „Total Length“ (IPv4) bzw. „Payload Length“ (IPv6) aus dem IP header ergibt. Wie in den Abbildungen 2.8 und 2.10 ersichtlich, werden von den standardmäßig verfügbaren „ping“ Implementierungen ganz konkrete Muster in das Datenfeld geschrieben. Der Sinn dieses variabel lang gestaltbaren Feldes ist die Ermöglichung des Austestens der Übertragungswege im Netzwerk bei unterschiedlichen Paketgrößen.

Insbesondere für die Ermittlung der „Maximum Transfer Unit“ (MTU), also der Bestimmung der maximal verwendbaren Paketgröße bevor Fragmentierung auftritt, ist das Datenfeld nützlich. Die MTU gibt damit die obere Schranke für das Anfügen von Daten vor. Sie liegt bei Ethernet Rahmen konkret bei 1500 Bytes [40, S. 6]. Abzüglich des Platzbedarfs der IP und ICMP header ergibt sich dadurch eine maximale Datenfeldgröße von 1472 Bytes für IPv4 und 1452 Bytes für IPv6.

Nachdem im Gegensatz zu den einzelnen, wenige Bytes oder gar nur Bits fassenden header Feldern die Breite des „Data“ Bereichs theoretisch nur durch die MTU in einer Kilobyte Größenordnung begrenzt wird, ist dieser für den Transport von größeren Datenmengen interessant.

# Kapitel 3

# Flexible Datentunnelung

In diesem Kapitel wird ein System zum verborgenen Informationstransport über das ICMP Protokoll vorgestellt. Dabei taucht unter anderem das Konzept der Proxy-Kommunikation als zentrales Element auf, welches direkte Datenverbindungen in indirekte überführt.

Eine notgedrungene Begleiterscheinung der Direktverbindung ist, dass einem Kommunikationspartner die Adresse des jeweils anderen bekannt sein muss. Ohne eindeutig zuordbare Adressen, ist schließlich kein zielgerichteter Datenaustausch möglich. Diese zwangsmäßige Identifizierbarkeit birgt allerdings Risiken. Eine nähere Erläuterung der Problematik und Motivation des Unterfangens folgt als Einleitung im Abschnitt 3.1.

## 3.1 Motivation

Die Indirektion einer Datenverbindung kann aus unterschiedlichen Beweggründen von Interesse und Nöten sein. Es gilt technische und nicht-technische Gründe sowie den Umstand der etwaigen Verdecktheit der Kommunikationsanstrengung zu unterscheiden. Technische Faktoren, die eine Direktverbindung erschweren oder unmöglich machen, können beispielsweise sein:

- Ein Verbindungsziel oder Protokoll wird von einer Firewall blockiert.
- Die Kommunikationspartner verwenden unterschiedliche IP Versionen.
- Es liegt ein Defekt oder eine Überlastung vor, weswegen einzelne Protokolle oder Schichten ausgefallen sind. Ein möglichs Szenario wäre beispielsweise der Ausfall eines Zwangs-Webproxies, wovon nur ICMP nicht betroffen ist.

Im Abschnitt 3.3 wird näher auf Umstände dieser Art eingegangen und erklärt, inwiefern in solchen Fälle im Hinblick auf die Anforderungen des vorgestellten Tunnelungskonzepts Abhilfe geschafft werden kann.

Es sind vor allem auch nicht-technische, menschliche Faktoren, die in manchen Situationen unmittelbare Direktverbindungen zweier Kommunika-

tionspartner nachvollziehbar problematisch erscheinen lassen. Die folgenden Punkte sollen einen neutralen Eindruck möglicher Betroffener bieten:

- Der Dissident eines repressiven Staates befürchtet vom Geheimdienst beschattet zu werden.
- Der Whistleblower scheut die Zurückverfolgbarkeit seiner Person.
- Der Kranke will über seine Leiden diskutieren, ohne sich als Betroffener die Blöße geben zu müssen.
- Der Teenager möchte als Konsument von Schwarzkopien der neuesten Musikproduktionen unerkannt bleiben.

Dass Datenverkehr analysiert, protokolliert und schließlich blockiert wird, kann unterschiedliche Gründe haben. Je nach Situation und Nation, können diese rein technischer, finanzieller, juristischer oder durchaus auch politischer Natur sein.

Zur vorbeugenden E-Mail Spambekämpfung können Institutionen, wie Internet Zugangsanbieter oder Webhoster beispielsweise das *Simple Mail Transfer Protocol* auf Port 25 sperren. Firmen nutzen gerne die Möglichkeit, anhand schwarzer Listen soziale Netzwerke, Videostreaming Portale, Pornografieangebote und ähnliche Kategorien für die Mitarbeiter während der Arbeitszeit zu blockieren. Internet Relay Chat (IRC) und Filesharingprotokolle sind ebenfalls berüchtigte Sperrkandidaten. In den Räumlichkeiten der FH Hagenberg wird beispielsweise die Nutzung von IRC über dessen Standardports unterbunden.

Länder mit repressiven Machtstrukturen üben besonderen Druck aus, um Bürger von etwaigem oppositionspolitischen Engagement im Internet abzuschrecken. 2010 waren laut dem Jahresbericht[1] der NGO *Reporter ohne Grenzen* 62 Staaten von Internetzensur betroffen. Aktuell sollen 125 „CyberdissidentInnen“ inhaftiert sein, also Blogger oder Teilnehmer von sozialen Netzen, die regierungskritische Inhalte publiziert haben. Die NGO *Committee to Protect Journalists* listet[2] für das Jahr 2010 insgesamt 145 verhaftete Journalisten weltweit auf, wobei 69 davon das Internet als Medium zur Publikation genutzt haben. China und Iran führen die Statistik mit den meisten politisch motivierten Verhaftungen an.

Besonders für Reporter, Dissidenten und Freidenker in der Bevölkerung solcher Länder ist die Möglichkeit zur Kommunikation „auf verschlungenen Pfaden“ und vorbei an den Überwachungs- und Sperrinstrumenten der Regierung, angesichts der drohenden Gefängnis- oder gar Todesstrafen, von fundamentaler Wichtigkeit.

[1] http://en.rsf.org/journalists-in-2010-targets-and-30-12-2010,39188.html
[2] http://www.cpj.org/imprisoned/2010.php

## 3.2 Flexibilität

Der Begriff der Flexibilität kann auf unterschiedliche Weise interpretiert werden. Zur näheren Erläuterung gilt es zu allererst abzugrenzen, auf welchem Abstraktionsniveau die Datentunnelung geschieht. Auf das OSI Referenzmodell Bezug nehmend, kommen dafür in erster Linie folgende Schichten in Frage:

- Anwendungsschichten (5–7)
- Transportschicht (4)
- Vermittlungsschicht (3)

Nach dem Modell sollen einzelne Schichten ausgetauscht werden können, ohne dass jeweils darüberliegende dadurch beeinflusst werden [59]. Unter dieser Prämisse bedeutet ein Tunnelungskonzept auf der Vermittlungsschicht, dass automatisch alle darüberliegenden Protokolle ebenfalls transportiert und wie gewohnt verwendet werden können, ohne dass diese in den Prozess der Tunnelung gesondert eingegliedert werden müssen. Folglich steigt die Flexibilität je niedriger die Schicht ist, auf der die Tunnelung erfolgt.

Im Kapitel 4 werden Projekte vorgestellt, die unter anderem auf der Vermittlungsschicht operieren. Das in diesem Kapitel behandelte Tunnelungssystem ist allerdings auf einer Abstraktionsstufe höher angesiedelt. Dadurch büßt es zwar einerseits Flexibilität ein, weist aber andererseits Vorteile auf, die es in der Praxis als legitime Alternative qualifizieren. Als Argumente für die Wahl der Transportschicht sprechen unter anderem folgende Punkte:

1. Angesichts der Nachteile, wie geringerer Datendurchsatz, höhere Latenz und gesteigerte Entdeckbarkeit bei entsprechender Nutzung, ist der Betrieb des Tunnels weniger als vollwertiger Ersatz für einen regulären Netzzugang geeignet, sondern stellt ein Werkzeug für bestimmte zeitlich begrenzte Situationen und einzelne Programme dar, die verdeckte Kommunikation erfordern. Ein Tunnelungsbetrieb auf Schicht 3 benötigt allerdings eine persistente, virtuelle Netzwerschnittstelle, während der Betrieb auf höheren Schichten in Form frei konfigurierbarer Anwendungsprogramme mehr Dynamik gewährt.

   Diese Auffassung wird relativiert, wenn der Tunnel nicht zum Zweck der verborgenen Kommunikation verwendet wird sondern zur Überbrückung einer netzwerktechnischen Anomalie (siehe Abschnitt 3.1).
2. Während auf der Transport- und Anwendungsschicht die programmspezifische Nutzung eines Tunnelungssystems verhältnismäßig leicht konfigurierbar ist, stellt dies auf der Vermittlungsschicht ohne ausdrücklicher Unterstützung und Selektierbarkeit unterschiedlicher Netzwerkschnittstellen seitens der in Verwendung befindlichen Programme, ein Problem dar. Selektives Tunneln bestimmter Verbindungen muss ansonsten über das manuelle Setzen von statischen Routen gelöst werden, was nur Experten zugemutet werden kann.

bob.localhost example.com

**Abbildung 3.1:** Unmittelbare und ungehinderte Datenübertragung zwischen zwei Computern in einem Netzwerk.

3. Die Einrichtung eines Tunnelungssystems auf der Vermittlungsschicht bedeutet einen Eingriff auf Betriebssystemniveau, also unter Windows die Installation eines Netzwerktreibers, der für die jeweils verwendete Version des Betriebssystems verfügbar sein muss. Für auf Unix basierende Systeme müssen sog. TUN Kerneltreiber aktiviert werden, deren Unterstützung aber möglicherweise im Betriebssystemkern nicht einkompiliert worden ist.

Die Verwendung des Flexibilitätsbegriffs zielt in dieser Arbeit vor allem auf die Konzeption der Tunnelungsprotokolle und einer damit verbundenen Softwarearchitektur ab. Im Zuge der Diplomarbeit entstand die Software „PingGate“ zur Demonstration des Tunnelungskonzeptes. Auf diese wird in den folgenden Abschnitten Bezug genommen, um Kommunikationsabläufe und Architekturspezifika anschaulich zu beschreiben.

## 3.3 Kommunikationsschemata

Tauschen Kommunikationspartner Informationen aus, tun sie das indem ein logischer Übertragungskanal geschaffen wird, über den Daten mono- oder bidirektional fließen. Technisch wird dieser Vorgang im Internet mittels paketorientierten Übertragungsprotokollen realisiert.

Einleitend zeigt Abbildung 3.1 die triviale Netzwerktopologie im Fall einer direkten Verbindung eines Teilnehmers unter der Adresse bob.localhost zu einem unter example.com erreichbaren Server. Bei der Kommunikation zwischen beiden Akteuren sind diese Adressen in den Absender- und Empfänger header Feldern des verwendeten Internet Protokolls der Version 4 oder 6 eingetragen. Dadurch können Datenpakete durch das Internet geroutet und schließlich an die vorgesehenen Empfänger zugestellt werden.

Man stelle sich nun vor, dass bob.localhost ein Rechner in einer Firma ist, deren IT Abteilung eine Firewall betreibt. Diese hat mehrere Aufgaben. Zum einen soll sie Spam E-Mails filtern, Firmeninterna vom Internet abschirmen und ausgehenden Datenverkehr zu bestimmten Websites blockieren, die vom Chef in Form einer schwarzen Liste vorgegeben werden. Es sei weiters angenommen, dass der neu eingestellte Praktikant Einschränkun-

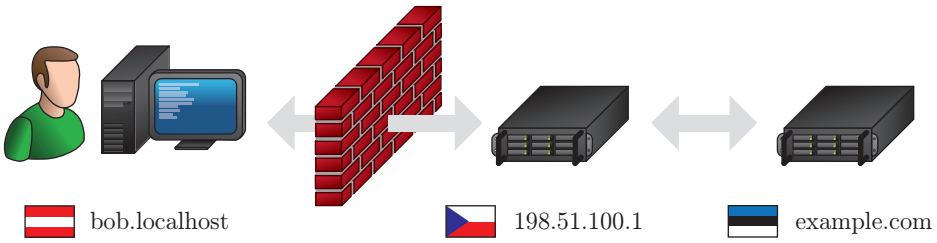



**Abbildung 3.2:** Ein Computer aus Österreich hinter einer Firewall ist über einen Proxy Server in Tschechien mit einem Computer in Estland verbunden.

gen dieser Art nicht respektiert und eine technischen Lösung zur Umgehung dieser Sperre anstrebt. Er möchte von seinem Arbeitsplatz aus die fiktive Domain example.com erreichen, welche dummerweise auf der schwarzen Liste verzeichnet ist. Jedes Datenpaket, das an die IP Adresse gerichtet ist, die sich hinter dem Domainnamen verbirgt, wird folglich von der Firewall konsequent verworfen, sodass keine Verbindung hergestellt werden kann.

Diese künstliche Unansprechbarkeit von example.com ist natürlich nur innerhalb der Firma und somit auch am Rechner des Praktikanten gegeben. Von jedem anderen Knoten im Internet, ist die Adresse aber grundsätzlich erreichbar. Diesen Umstand kann man sich nun zunutze machen und die Verbindung zu example.com *indirekt* über einen Knoten herstellen, der nicht von einer Sperre betroffen ist. In einer solchen Konstellation wird der Umleitungsknoten als *Proxy* Server bezeichnet. Abbildung 3.2 illustriert dieses Schema, wobei unter der Adresse 198.51.100.1[3] der Proxy Server betrieben wird, auf den der Benutzer von bob.localhost aus zugreift, um letztendlich example.com zu erreichen.

Die in der Illustration verwendeten Länderflaggen sollen unterstreichen, dass zwischen den involvierten Rechnern durchaus erhebliche geographische Distanzen liegen können. Dies ist vor allem aus einem juristischen Blickwinkel interessant und in der Praxis oft ein Kriterium für die Nutzung eines Proxy Servers. Während auf der einen Seite von Österreich aus eine Datenverbindung zu einem Rechner in Tschechien registriert werden kann, gilt dasselbe auf umgekehrte Weise von Estland aus. Die eigentliche, logische Verbindung von Österreich nach Estland wird somit verschleiert und kann ohne Einblick in möglicherweise mitgeführte Protokolle bezüglich ein- und ausgehender Datenverbindungen am tschechischen Server nicht nachvollzogen werden.

Beim Surfen im Internet kommt auch der Durchschnittsanwender gelegentlich in Situationen, in denen auf Websites die IP Adresse des anfragenden Rechners relevant ist. Anhand der IP Adresse kann nämlich bestimmt wer-

[3]Es werden in den Illustrationen die in [7, 11] für Beispiel- und Dokumentationszwecke reservierten Domänen und IP Adressen und verwendet.

den, in welchem Land sich der Surfer potentiell befindet. Die Assoziation von delegierten IPv4 und IPv6 Netzblöcken mit Länderkürzeln sind öffentlich einsehbar[4].

Aus Urheberrechts- und Lizenzgründen wertet beispielsweise das Videoportal *Youtube* das Herkunftsland des Surfers auf diese Weise aus und blendet gegebenenfalls statt dem angeforderten Video einen Hinweis ein:

> Dieses Video enthält Content von (...). Es ist in deinem Land nicht verfügbar.

Ebenso verfährt der amerikanische Fernsehsender *ABC*, wenn man außerhalb der USA versucht, deren Streamingangebote zu nutzen:

> You appear to be outside the United States or its territories. Due to international rights agreements, we only offer this video to viewers located within the United States and its territories.

Unter Verwendung eines Proxy Servers aus einer für das jeweilige Angebot erforderlichen oder günstigen geographischen Region sind Restriktionen dieser Art wirkungslos. In der Praxis stellt die Verfügbarkeit eines Proxies in konkreten Ländern, sowie dessen Übertragungsbandbreite ein Problem dar. Es gibt jedoch kommerzielle Anbieter, sowie kostenlose Dienste, die minütlich aktualisierte Listen von benutzbaren Servern inklusive deren Bandbreite und Nationalität bereitstellen[5]. Die Inanspruchnahme solcher Server ist rechtlich allerdings zum Beispiel im Hinblick auf §118a StGB *„Widerrechtlicher Zugriff auf ein Computersystem“* als kritisch zu werten. Immerhin erzeugt man für den Betreiber des Servers möglicherweise kostenintensiven Datenverkehr, belegt Ressourcen und tätigt Informationsaustausch im Namen des Betreibers.

Proxy Server unterstützen für gewöhnlich die Benutzung durch mehrere Teilnehmer gleichzeitig, wobei jeder Teilnehmer wiederum mehrere Datenverbindungen zu einem oder mehreren Zielen herstellen kann. Dieser Fähigkeit wird in Abschnitt 3.5.2 bei der Konzeption des Tunnelungsprotokolls besondere Beachtung geschenkt. Abbildung 3.3 dient als abstrakte Illustration der Topologie und Weiterentwicklung des Schemas aus Abbildung 3.2.

Die Proxy Funktionalität wird mithilfe einer Software auf den dargestellten Computern realisiert. Auf die Implementierung „PingGate“ Bezug nehmend, lassen sich zur exakteren Darstellung des technischen Aufbaus nun einige Symbole durch das PingGate Programmsymbol – ein Verkehrszeichen zur Visualisierung der Umleitung des Datenverkehrs – ersetzen. Abbildung 3.4 zeigt die Topologie mit Fokus auf die verwendete Software zur Kommunikation. Unterschiedliche Teilnehmer wie bob.localhost oder alice.localhost werden jetzt durch Programminstanzen von 1 bis $n$ konkretisiert. Obwohl es

[4] ftp://ftp.arin.net/pub/stats
[5] http://www.hidemyass.com/proxy-list

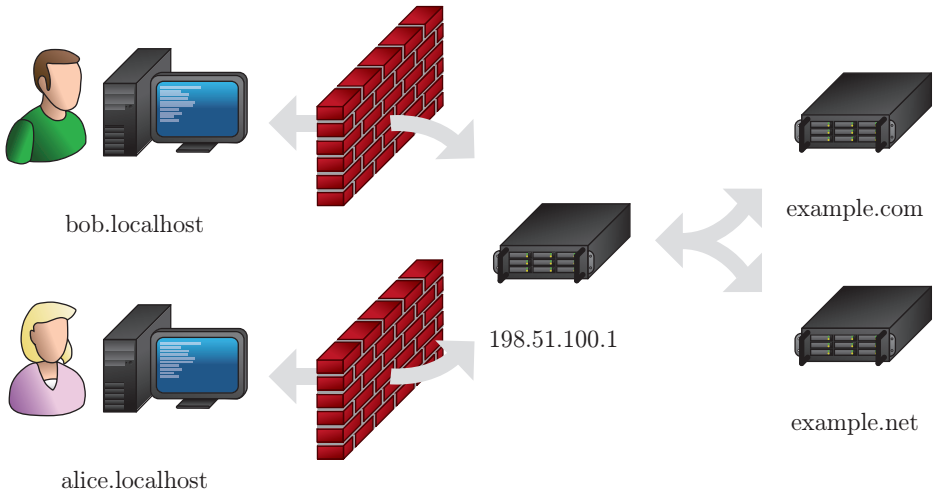



**Abbildung 3.3:** Zwei Teilnehmer greifen von unterschiedlichen Rechnern aus auf denselben Proxy Server zu, um jeweils unterschiedliche Zielsysteme zu erreichen.

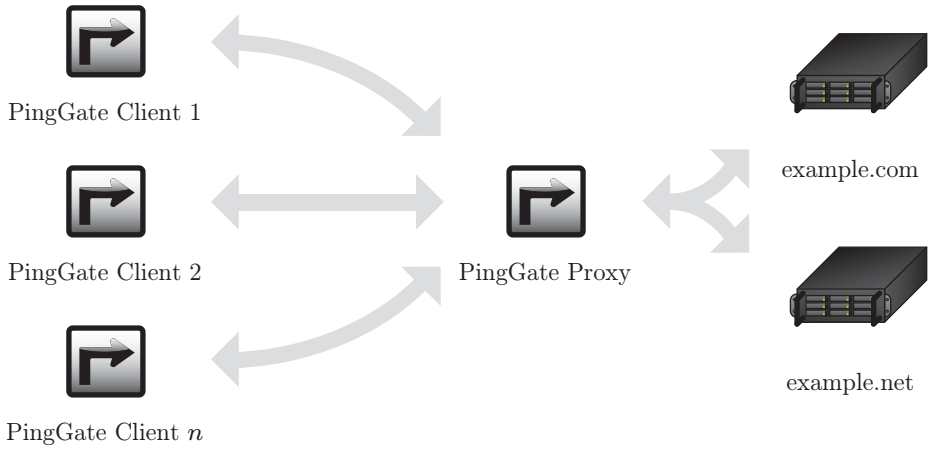



**Abbildung 3.4:** Verfeinerung der Topologie aus Abbildung 3.3 aus einer Softwareperspektive.

sich stets um dasselbe Programm handelt, erfolgt die Ausführung in einem von zwei Modi („PingGate Client“ und „PingGate Proxy“). Das Verhalten und die Anforderungen der Software unterscheiden sich in den beiden Modi zwar, aber es ist vor allem eine Bequemlichkeit aus Implementierungssicht, dieselbe Anwendung mit jeweils unterschiedlichen Konfigurationsparametern zu verwenden.

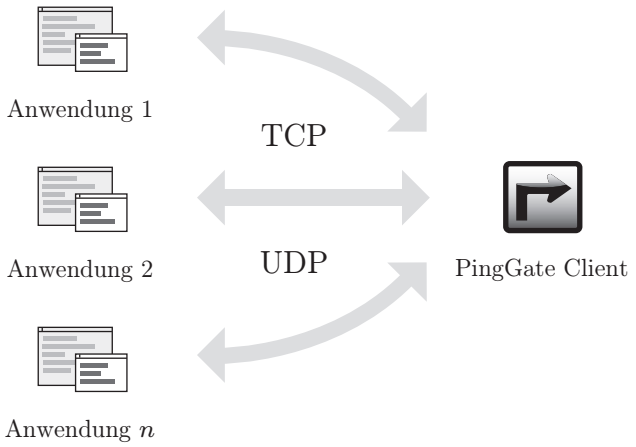



**Abbildung 3.5:** Eine theoretisch beliebige Anzahl an Anwendungen kommuniziert gegebenenfalls parallel mit der PingGate Client Ausführung und verwendet dabei das TCP oder UDP Protokoll.

## 3.4 SOCKS

Der Client und Proxy Modus von PingGate bilden logisch eine Einheit, die einen Proxy Server mit immanenter Datentunnelung darstellt. Ein solcher Server muss eine Schnittstelle anbieten, die Anwendungen benutzen können, um den Proxy Dienst in Anspruch zu nehmen. Im Fall der PingGate Implementierung ist diese Schnittstelle nur im clientseitigen Betriebsmodus relevant. Damit die Nutzung der Proxy Funktionalität für eine Vielzahl unterschiedlicher Programme möglich ist, muss eine etablierte und im besten Fall standardisierte Schnittstelle samt Protokoll verwendet werden. Das in [26] beschriebene *SOCKS* Protokoll, ein universelles und verhältnismäßig einfach zu implementierendes Proxy-Protokoll, fällt unter diese Kategorie und ist auch in der PingGate Client Anwendung implementiert. Das Protokoll ist in der Version 4 und 5 zwar weit verbreitet, aber selbst wenn ein Programm SOCKS nicht direkt unterstützt, kann über ein Hilfsprogramm trotzdem eine Verbindung zu einem – dann allerdings statischen – Verbindungsziel über den Proxy Server hergestellt werden. Fortan wird von SOCKS der Version 5 ausgegangen, das eine Erweiterung der weniger komplexen Version 4 ist und unter anderem IPv6 unterstützt. Abbildung 3.5 dient zur Visualisierung des Teilbereichs der Topologie, die in diesem Zusammenhang bezüglich Proxy Schnittstelle eine Rolle spielt.

Mit dem Proxy Server wird per TCP[6] und eventuell auch UDP[7] kommuniziert. Dabei handelt es sich um dem IP übergeordnete Protokolle der

[6] **T**ransmission **C**ontrol **P**rotocol [39]
[7] **U**ser **D**atagram **P**rotocol [36]

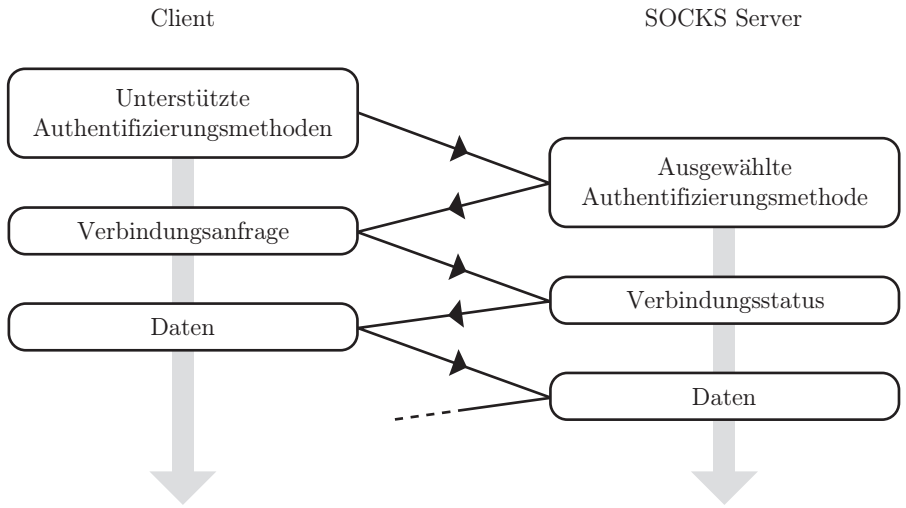



**Abbildung 3.6:** Eine wechselseitige SOCKS Kommunikation nach dem Aufbau einer neuen Verbindung zum Proxy Server.

Schicht 4, also der Transportschicht. Die weitere Beschreibung erfolgt im Bezug auf das TCP. Als verbindungsorientiertes Protokoll wird damit ein gesicherter Übertragungskanal hergestellt, der idealerweise die Vorgänge des Verbindungsauf- und -abbaus sowie die Übertragung von Nutzdaten enthält.

Möchte nun eine Anwendung per SOCKS eine Verbindung zu einem bestimmten Ziel wie beispielsweise example.com herstellen, dann darf dieses Ziel nicht direkt angesprochen, sondern lediglich eine Verbindung zum Proxy Server hergestellt werden. Steht diese Verbindung, muss das SOCKS Protokoll befolgt werden, welches aus mehreren Phasen besteht. Diese sind in Abbildung 3.6 in einem Ablaufdiagramm dargestellt.

Der Server kann so konfiguriert sein, dass die öffentliche Benutzung durch eine Zugangsbeschränkung verhindert wird. Deshalb beginnt der Ablauf mit einer Authentifizierungsequenz, durch welche der Administrator des Servers zum einen nur ausgewählten Benutzern den Zugriff gestatten und zum anderen Aktivitäten des Proxy Servers mit einem konkreten Benutzer assoziieren kann.

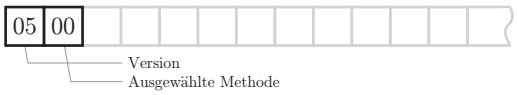
1. Die Anwendung gibt bekannt, welche Authentifizierungsmethoden sie unterstützt. Eine solche Methode wird durch eine Kennzahl spezifiziert, wobei in [26, S. 3] einige vordefinierte Methoden aufgelistet sind. Exemplarisch sind zwei davon:

   - Keine Authentifizierung (Methode 0)
   - Benutzername und Passwort (Methode 2) [25]

   Im Ablauf in Abbildung 3.6 ist kein Authentifizierungsvorgang skizziert, folglich wurde die Methode 0 gewählt. Wird von der Anwendung

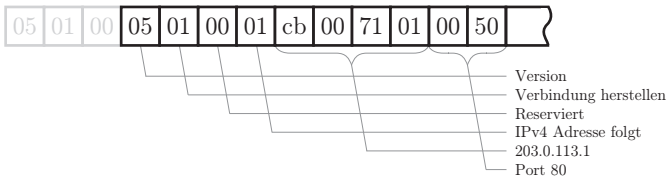
überhaupt kein Authentifizierungsmechanismus unterstützt, sind die ersten drei Bytes die von der Anwendung zum Server gesendet werden, die folgenden:


2. Serverseitig wird im nächsten Schritt überprüft, ob zumindest eine der angegebenen Authentifizierungsmethoden akzeptiert werden kann. Sollte der Proxy Server so konfiguriert worden sein, dass er auch ohne vorheriger Anmeldung Verbindungen zulässt, dann gilt die angebotene Methode 0 als akzeptabel. Es folgen zwei Bytes als Antwort vom Server:

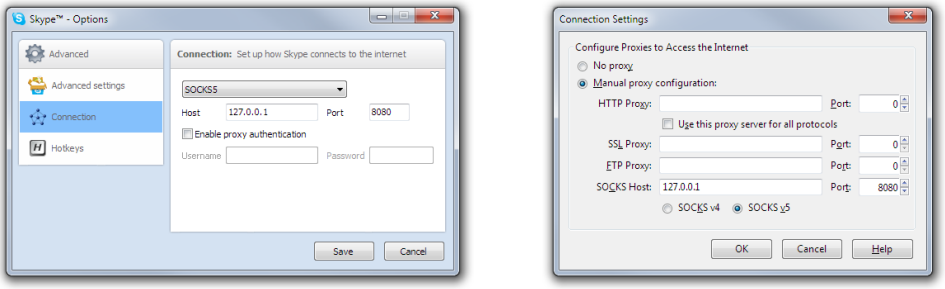



3. Nachdem in diesem Beispiel keine Authentifizierung notwendig ist, wird von der Anwendung nun einer von drei verfügbaren Befehle erwartet. Der Wichtigste ist der Verbindungsbefehl, der den Proxy Server anweist, eine Verbindung zu einem Zielrechner aufzubauen. Die beiden anderen Befehle werden für die Benutzung von UDP und der Möglichkeit des Zurückverbindens eines entfernten Rechners zum Proxy benötigt, was aber den Rahmen dieses Beispiels sprengen würde.

   Möchte die Anwendung nun eine Verbindung zur Adresse 203.0.113.1 aufbauen, werden weitere zehn Bytes an den Server gesendet:

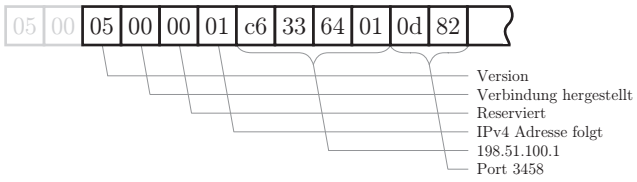



4. Der Server versucht daraufhin, eine Verbindung zur angegebenen Adresse herzustellen. Dieses Vorhaben kann entweder erfolgreich sein, oder aus verschiedenen Gründen fehlschlagen. In [26, S. 5] sind konkret acht verschiedene Fehlercodes definiert. In diesem Beispiel sei angenommen, dass die Verbindung hergestellt werden konnte, was mit dem Wert Null signalisiert wird. Die finale Antwort des Proxy Servers an die Anwendung ist nun diese Erfolgsmeldung:




(a) (b)

**Abbildung 3.7:** Einstellungsfelder mit der Möglichkeit der Auswahl des SOCKS Protokolls in der Telefoniesoftware *Skype 5.0* (a) und im Webbrowser *Firefox 4.0* (b). Die Screenshots wurden aus Übersichts- und Platzgründen nachbearbeitet und vereinfacht.




Zusätzlich wird die Socketspezifikation des Proxy Servers für diese Verbindung in Form einer IP Adresse und lokaler Portnummer mitgeliefert. Diese Information wird beispielsweise von Protokollen wie FTP[8] (zumindest im aktiven Modus) benötigt.

5. Ab der positiven Bestätigung der Verbindungsanfrage, können regulär Daten gesendet und empfangen werden. Es besteht nun auf beiden Seiten eine TCP Verbindung, also von der Anwendung zum Proxy Server und von diesem zum angegebenen Verbindungsziel. Trennt schließlich die Anwendung die Verbindung zum Server, kappt dieser auch die assoziierte Verbindung zu 203.0.113.1. Umgekehrt gilt natürlich dasselbe.

Die PingGate Client Ausführung mit der SOCKS Schnittstelle läuft typischerweise lokal am Rechner des Benutzers, weshalb auf die SOCKS Authentifizierung verzichtet wird. Sie erfolgt stattdessen im Zuge der Kommunikation mit dem PingGate Proxy (siehe Abschnitt 3.5.1).

Abbildung 3.7 zeigt Screenshots zweier populärer Programme, die unter anderem das SOCKS Protokoll unterstützen. Läuft eine lokale PingGate Client Instanz und soll die Anwendung über diese kommunizieren, dann wäre die SOCKS Schnittstelle unter einer Loopback Adresse wie 127.0.0.1 und dem Port 8080 erreichbar, wobei dieser anhand einer Konfigurationsdatei frei gewählt werden kann.

[8] **F**ile **T**ransfer **P**rotocol [41]

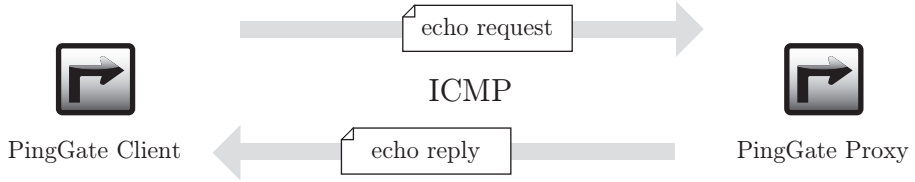



**Abbildung 3.8:** Zwischen PingGate Client und Server Instanzen wird ausschließlich über ICMP Nachrichten kommuniziert, wobei Echo Anfragen stets vom Client ausgehen.

## 3.5 Tunnelung

Die PingGate Client Ausführung nimmt über die SOCKS Schnittstelle Verbindungsanfragen von Programmen an, deren Kommunikation verdeckt in Form von ICMP Nachrichten transportiert werden soll. Es können nun jederzeit Verbindungsanfragen eingehen, die von einem einzelnen oder mehreren Programmen ausgehen können. Die Herausforderung besteht darin, diese Anfragen an den PingGate Proxy weiterzuleiten, welcher die Verbindung schließlich herstellen soll.

Abbildung 3.8 zeigt den Ausschnitt der Topologie, in dem die Datentunnelung stattfindet. Es werden ICMP Echo Nachrichtentypen verwendet, wobei der PingGate Proxy stets passiv kommuniziert. Er nimmt ausschließlich Echo Anfragen entgegen und beantwortet diese mit Echo Antworten, sendet aber selbst nie Anfragen an die Clients. Es handelt sich also um das Konzept des *Pollings*.

Dies hat den folgenden Grund: Anders als im Internet direkt erreichbare Server die bestimmte Dienste zur Verfügung stellen, befinden sich Heim- oder Arbeitsplatzrechner einzelner Personen in der Regel hinter Routern mit NAT. Im Einleitungskapitel wurde dieses Verfahren im Abschnitt 1.1 bereits im Zusammenhang mit der Verknappung des verfügbaren IPv4 Adressraumes erwähnt. Damit teilen sich mehrere Computer dieselbe IP Adresse, mit der sie im Internet aufscheinen. Aus dem Internet ist somit kein individueller Computer hinter der NAT Barriere ansprechbar.

Die Initiative des Verbindungsaufbaus muss daher immer vom Gerät hinter dem NAT Router ausgehen. Dieser kann nämlich durch die Beobachtung des Kommunikationsablaufs eingehende IP Pakete einer bestehenden Verbindung und damit einem konkreten Computer zuordnen und an diesen weiterleiten. Dasselbe gilt für ICMP Echo Typen. Die NAT Implementierung kann eine eingehende Echo Antwort mit einer zuvor von einem Computer gesendeten Anfrage assoziieren. Sollte diese aber an die gemeinsam genutzte IP Adresse gerichtet worden sein, kann der Router sie nur selbst beantworten. Da die Anfrage also nie bei einem individuellen Computer ankommt, kann PingGate Proxy keine aktive Kommunikationsrolle übernehmen.

Das Polling ist im Übrigen der Grund dafür, dass für eine bidirektionale Datentunnelung das ICMP Protokoll verletzt werden muss, zumindest wenn das Datenfeld in beide Richtungen mit derselben Kapazität als „storage channel“ genutzt werden soll. Anstatt nämlich eine Kopie des Datenfeldes zurückzusenden, wie [37, S. 15] es vorschreibt, bettet der PingGate Proxy seine Antwortdaten ein. Dabei handelt es sich um jene Daten, die per TCP oder UDP von einem der Server im Internet zurückgesendet werden, zu denen PingGate Proxy nach einer entsprechenden Anfrage des PingGate Clients, eine Verbindung hergestellt hat (siehe Abschnitt 3.6).

Neben der Notwendigkeit des Pollings, sind für eine zuverlässige Datentunnelung weitere Dinge zu beachten und Mechanismen zu implementieren. So wird im Fall des ICMP beispielsweise nicht garantiert, dass

- der Verlust eines gesendeten Pakets beim Sender oder Empfänger bemerkt wird.
- einzelne Pakete in der richtigen Reihenfolge vom Empfänger interpretiert werden.
- ein gesendetes Paket nur genau einmal beim Empfänger verarbeitet wird.

Nachdem aber unter anderem das gesicherte TCP (das alle aufgezählten Eigenschaften bietet) getunnelt werden soll, müssen dessen Vorteile, was die Sicherungsfunktionen betrifft, in irgendeiner Form mit dem diesbezüglich in allen Belangen unterlegenen ICMP Tunnel emuliert werden. Die Mechanismen der Fehlererkennung und -korrektur vom TCP stehen nicht zur Verfügung, weil ausschließlich die Nutzdaten der einzelnen Pakete im Tunnel transportiert werden, nicht jedoch die Paketheader wie im Fall der Tunnelung in der Vermittlungsschicht. Somit erscheint die Verbindung zwischen Anwendung und SOCKS Schnittstelle, sowie zwischen PingGate Proxy und dem Zielrechner im Internet – sofern natürlich keine zusätzlichen Beeinträchtigungen vorliegen – immer stabil und ungestört, obwohl im ICMP Tunnel dazwischen eventuell ein erheblicher Paketverlust zu verzeichnen ist.

Der Schlüssel zur Absicherung des Tunnels (im Sinne von der Emulation der Sicherungsmechanismen des TCP) ist das Mitführen einer aufsteigenden Sequenznummer, wofür praktischerweise ein eigenes Feld im ICMP Header (Abbildung 2.6) vorgesehen ist. Außerdem muss ein bestimmtes Kommunikationsverfahren angewandt werden, was das Sende- und Empfangsverhalten betrifft. Abbildung 3.9 illustriert das Konzept in Form eines Flussdiagramms für kommunizierende PingGate Client und Proxy Instanzen. Während des Betriebs befinden sich beide in einer Endlosschleife. Da die PingGate Proxy Ausführung nicht selbstständig die Verfügbarkeit neu eingegangener Daten signalisieren kann, muss trotzdem stets ein (inhaltloser) Datenaustausch stattfinden, auch wenn gar keine zu sendenden Daten clientseitig vorliegen.

Startet ein Benutzer seine lokale PingGate Client Instanz, dann versucht sich diese mit einem PingGate Proxy zu verbinden. Dadurch wird ein end-

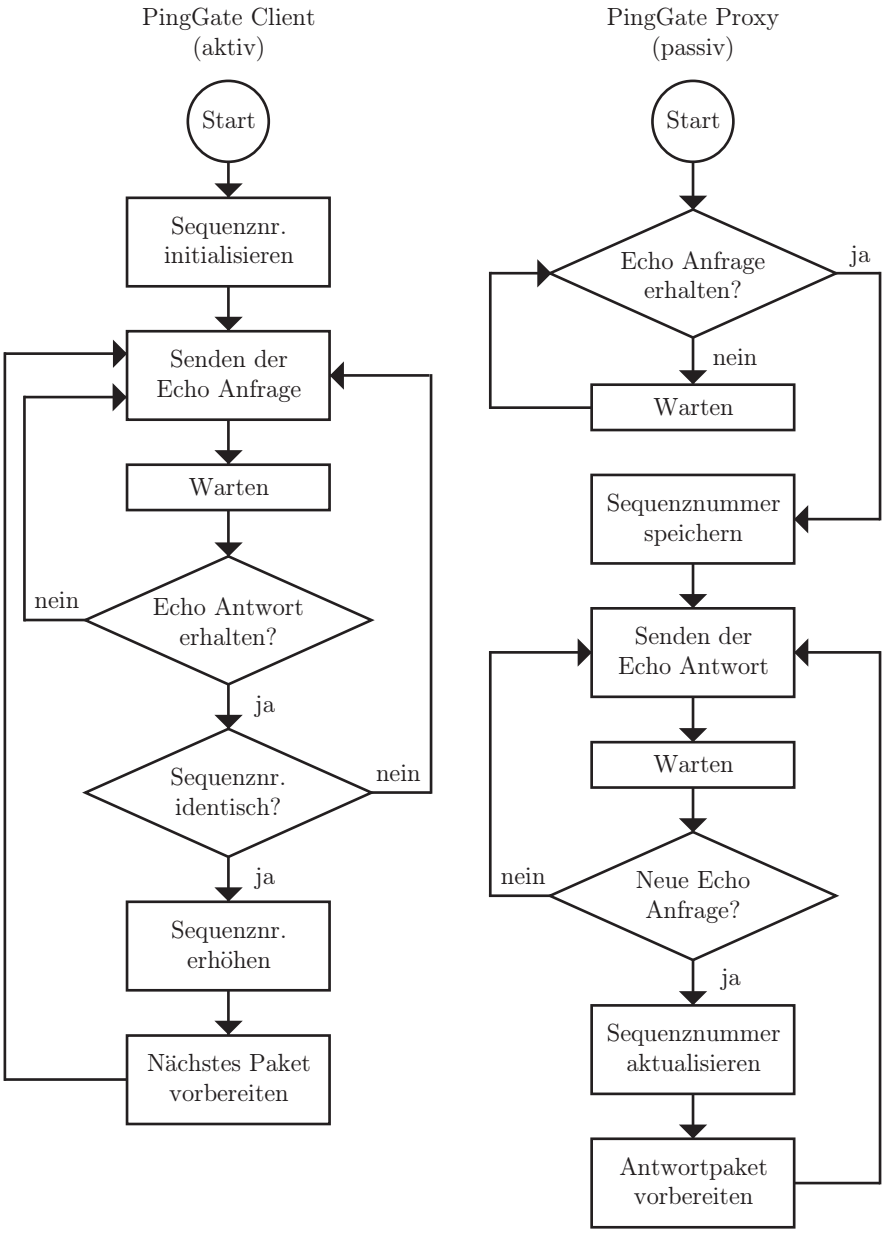



**Abbildung 3.9:** Die client- und serverseitigen Sende- und Empfangsvorgänge mit Fokus auf der Verwendung einer Sequenznummer.

licher Automat durchlaufen, der in Abbildung 3.10 skizziert ist. Von den fünf Phasen ist die Authentifizierung optional und die Abmeldung für den eigentlichen Vorgang der Datentunnelung nicht unmittelbar erheblich.

In den folgenden Abschnitten werden nun sämtliche Phasen des Nachrichtenaustausches im Detail beschrieben.

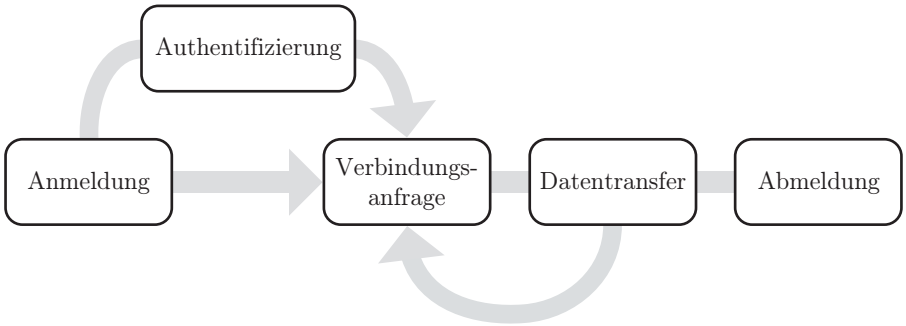



**Abbildung 3.10:** Während einer Sitzung werden in Form einer Zustandsmaschine mehrere Kommunikationsphasen durchlaufen.

### 3.5.1 Anmeldung

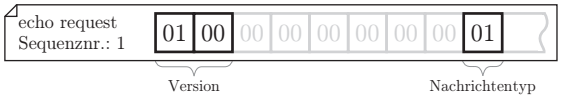
Jede PingGate Instanz muss sich nach dem Start anmelden. Erst wenn die Anmeldung erfolgreich war, können Anwendungen über die SOCKS Schnittstelle Verbindungsanfragen tätigen. Die PingGate Proxy Ausführung unterscheidet anhand der IP Adresse des Absenders, sowie des „Identifier“ Feldes im ICMP Paketheader (Abbildung 2.6), welche Instanz gerade kommuniziert. Durch die adressbasierende Unterscheidung werden einzelne Computer und durch den Identifikationswert einzelne Anwendungsinstanzen darauf separiert. Dieses Unterscheidungskriterium wäre für einen breiten Einsatz der Software allerdings suboptimal, weil auf zwei unterschiedlichen Rechnern hinter einem NAT Router die Identifikationsnummern kollidieren können.

Zur Identifizierung eines einzelnen Benutzers, der eine PingGate Client Ausführung bedient und die Tunnelungsfunktion benutzen möchte, gibt es mehrere Möglichkeiten. Im Grunde können an dieser Stelle, die im Abschnitt 3.4 beim SOCKS Protokoll beschriebenen Authentifizierungsmethoden genauso Anwendung finden. Der Einfachheit halber wird im Zuge dieser Sektion eine Anmeldung mit einem Benutzernamen und dazugehörigen Passwort beschrieben, wobei zur Steigerung der Sicherheit auch asymmetrische Kryptographie zum Einsatz kommen könnte.

Ein Passwort im Klartext zu übertragen ist problematisch, weil der Anwender dieses möglicherweise auch noch wo anders verwendet oder eventuell sein persönliches Schema zur Passwortgenerierung ableitbar wäre. Deshalb ist für den Fall, dass die Übertragung abgehört wird, die Transformation des Passworts in einen Streuwert sinnvoll. Dadurch ergibt sich außerdem der Vorteil, dass zum Beispiel bei der Wahl des SHA-256 Algorithmus unabhängig von der Passwortlänge stets eine feste Anzahl von 32 Bytes im Protokollformat vorgesehen werden kann.

Da stets die PingGate Client Instanz die aktive Rolle übernimmt und Echo Anfragen aussendet, initiiert sie auch den Anmeldevorgang. Sie sendet

dazu eine Begrüßungsnachricht an einen PingGate Proxy, die auch die Versionsnummer des Clients enthält. Die Zuweisung des Nachrichtentyps „Begrüßung“ an ein ICMP „echo request“ Paket, wie auch das Einbetten der Versionsinformation in selbigem, geschieht durch eine entsprechende Manipulation des Datenfeldes. Sämtliche zu transportierende Daten müssen in serialisierter Form als Bytesequenz vorliegen, damit sie ins Datenfeld eingebettet werden können. Das Format einer solchen Bytesequenz ist für jede Nachricht exakt definiert und sieht für die Begrüßungsnachricht folgendermaßen aus:




Die ersten zwei Bytes sind also für die Versionsnummer vorgesehen. Das höherwertige Byte könnte zur Angabe des „major“, und das niederwertige zur Angabe des „minor“ Teils der Softwareversion verwendet werden. In der Grafik wäre damit die Version „1.0“ kodiert.

Die Versionsnummer ist wichtig, sobald von einer Software unterschiedliche Versionen mit entsprechend verändertem Kommunikationsverhalten und Funktionsumfang in Verwendung sind. Es ist dann davon auszugehen, dass Situationen auftreten, in denen zwei Kommunikationspartner in jeweils unterschiedlichen Protokollversionen zu kommunizieren versuchen. Wird in einem solchen Fall auf die Versionierung keine Rücksicht genommen und werden keine Mechanismen zur Erkennung der Problematik verwendet, kann vor allem für den Anwender das Aufspüren des Problems schwierig sein. Bei nur kleinen versionsbedingten Formatunterschieden wäre dann unter Umständen die Kommunikation im Ansatz noch möglich, bricht schließlich aber beim Aufeinandertreffen der ersten Unterschiede unerwartet zusammen.

Ist im Protokoll aber von vornherein der Austausch der jeweiligen Versionen vorgesehen, kann auf Formatunterschiede entsprechend reagiert werden und vom Kommunikationspartner mit der neueren Version beispielsweise eine Kompatibilitätsschnittstelle angeboten werden. Ist keine Abwärtskompatibilität vorgesehen, kann immerhin die Verbindung mit einer entsprechenden Fehlermeldung ordnungsgemäß getrennt werden. Der Protokolldesigner muss bezüglich Versionsbehandlung also vorausschauend planen.

Der Nachrichtentyp wird aus einem bestimmten Grund stets im neunten Byte kodiert. Wie im Abschnitt 2.2.1 des Kapitels 2 ersichtlich ist, wird das Datenfeld der von einer standardmäßigen „ping“ Implementierung gesendeten Echo Anfragen, mit einem bestimmten Bytemuster befüllt. Unter Linux belegt die `timeval` Struktur die ersten acht Bytes mit systemzeitabhängigen Werten. Befände sich nun innerhalb dieser Bytes die Typnummer des Pakets, könnte sich abhängig von der Zeit, zufällig eine gültige Nummer ergeben und diese zu Missinterpretationen führen, wenn reguläre Echo Anfragen an den Rechner gesendet werden auf dem die PingGate Proxy Anwendung läuft.

| *Nummer* | *Bezeichnung* | *Client* | *Proxy* | *Vorgänger* |
|---|---|---|---|---|
| 1 | Begrüßung | • | | |
| 2 | Proxy Konfiguration | | • | 1 |
| 3 | Authentifizierung | • | | 2 |
| 4 | Auth. Ergebnis | | • | 3 |
| 5 | Leerlauf | • | | 2 od. 4 |
| 6 | Verbindungsanfrage | • | | 5 |
| 7 | Verbindungsstatus | | • | 6 |
| 8 | *Reserviert* | | | |
| 9 | Statusabfrage | • | | 7 |
| 10 | Client-Daten | • | | 7 od. 11 |
| 11 | Proxy-Daten | | • | 10 |
| 12 | Client-Abbruch | • | | 7 od. 11 |
| 13 | Proxy-Abbruch | | • | 10 |
| 14 | Abmeldewunsch | • | | ab 3 |
| 15 | Entlassung | | • | 14 |

**Tabelle 3.1:** Alle Nachrichtentypen des Tunnelungsprotokolls im Überblick. In der Client und Proxy Spalte ist ersichtlich, ob das Paket von einer PingGate Client oder Proxy Instanz gesendet wird.

Tabelle 3.1 zeigt unter anderem die Nummer und Bezeichnung jedes der insgesamt 14 definierten Nachrichtentypen, die zur flexiblen Datentunnelung benötigt werden. Nach Gleichung 2.2 für die standardmäßige Datenfeldbelegung unter Linux durch die *ping* Implementierung, hat das Byte an der Stelle 9 den Wert 8. Deshalb ist dieser Wert zum Ausschluss der Missinterpretation einer regulären Echo Anfrage reserviert und bezeichnet keinen unterstützten Nachrichtentyp. Der Vollständigkeit halber sei in Bezug auf Gleichung 2.1 für Windows Systeme zusätzlich der Wert 105 reserviert.

Nach dem Eingang der Begrüßungsnachricht am PingGate Proxy wird in diesem für den anfragenden Client ein Kontext erstellt und der in Abbildung 3.11 in Form eines Ablaufdiagramms dargestellte Anmeldevorgang durchlaufen. Mit jedem Kontext ist unter anderem ein Zeitstempel assoziiert, womit etwaige Zeitüberschreitungen registriert werden können und ab einer gewissen Dauer in der keine Kommunikation erfolgt, der Kontext wieder gelöscht und der PingGate Client somit abgemeldet wird. Die zulässige Dauer ist abhängig von der Tunnelnutzung und kann bei einem entsprechend verdeckten Betrieb und langen Latenzzeiten zwischen einzelnen Nachrichten,

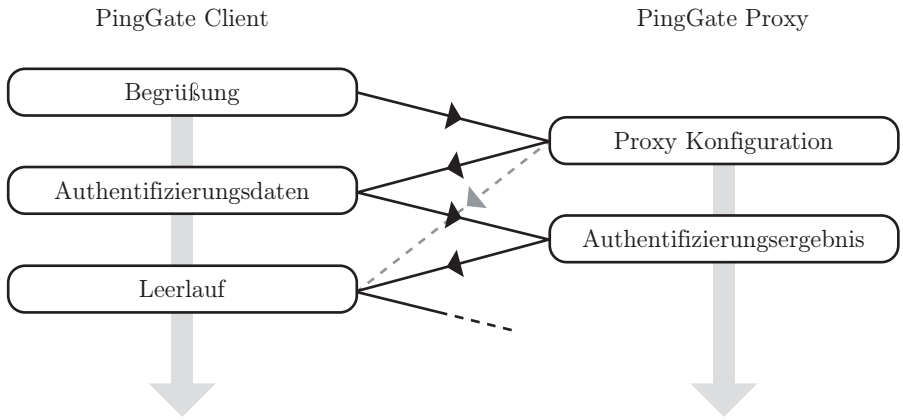
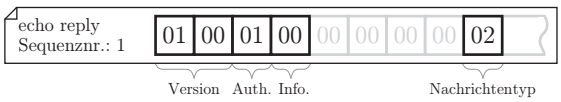



**Abbildung 3.11:** Ablauf des Anmeldevorgangs mit optionaler Authentifizierung.

sehr großzügig gewählt werden. Dabei handelt es sich um eine typische Benutzereinstellung, die über eine Konfigurationdatei je nach Motivation für den Tunnelbetrieb, entsprechend festgelegt werden kann.

Die erste Antwort des PingGate Proxies auf die Begrüßung erfolgt in Form der Darlegung seiner Konfiguration:



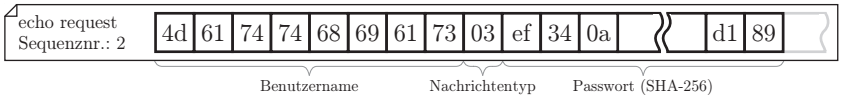

Diese umfasst wiederum ein Versionsnummer und den binären Hinweis, ob eine Authentifizierung erforderlich ist oder nicht. Außerdem ist ein Informationsfeld vorgesehen, dessen Wertbelegung beispielsweise eine der folgenden Bedeutungen haben kann:

1. Der Dienst steht regulär zur Verfügung
2. Die PingGate Client Version wird nicht unterstützt und weiterführende Verbindungs- oder etwaige Authentifizierungsversuche werden deshalb fehlschlagen.
3. Der Dienst steht nicht zur Verfügung, weil eine maximale Anzahl von Verbindungen oder Benutzern erreicht ist.
4. Der Dienst steht wegen Wartungsarbeiten temporär nicht zur Verfügung und sollte später erneut kontaktiert werden.
5. Der Dienst ist aufgrund eines Fehlers permanent außer Betrieb.

An dieser Stelle ist wieder eine vorausschauende Planung der möglichen Fehlersituationen notwendig. Gegebenenfalls ist zur flexibleren Information des Benutzers im Protokoll statt einer Zahl als Fehlercode, die Einbettung einer textuellen Fehlermeldung vorzusehen. Der Vorteil eines Codes ist allerdings, dass wie in diesem Beispiel ein einzelnes Byte zur Speicherung der

Information ausreicht. Außerdem sind auf diese Weise in Bezug auf Sprache, lokalisierte Ausgaben beim Benutzer möglich.

Es ist zu beachten, dass die ICMP Sequenznummer der Antwort stets dieselbe ist, wie die in der korrespondierenden Anfrage (siehe Flussdiagramm 3.9). Für die folgende Nachricht wird die Nummer vom PingGate Client nun inkrementiert, wobei der Typ des nächsten Pakets vom Authentifizierungsfeld des Proxy Konfigurationspakets abhängt. Ist eine Authentifizierung erforderlich, folgt nun das entsprechende Authentifizierungspaket, ansonsten die Leerlauf Benachrichtigung. In diesem Beispiel muss eine Anmeldung mit Benutzername und Passwort erfolgen, wobei der diesbezügliche Nachrichtentyp folgendermaßen aussieht:




Der Einfachheit halber wurden die ersten acht Bytes komplett für den Benutzernamen reserviert. Dieser darf also maximal acht Zeichen lang sein. Ist er kürzer, werden die unbelegten Bytes mit Nullen aufgefüllt. Sollte dies für einen seriösen Einsatz der Software zu unflexibel sein, kann beispielsweise ein Längenfeld im Format integriert werden und damit ab einer gewissen Stelle ein beliebig langer, nur durch die maximale Datenfeldgröße begrenzter Benutzername eingefügt werden.

Beim Schreiben und Lesen des Namens durch die PingGate Implementierungen ist eventuell auf die Zeichenkodierung zu achten. Aus technischen Gründen sind die gültigen Zeichen für einzelne Buchstaben eines Benutzernamens typischerweise eingeschränkt und lassen sich in 7-Bit ASCII Codes darstellen. Sollten aber beispielsweise die Wahl deutscher und türkischer Sonderzeichen zugelassen sein, lässt sich diese Kombination nicht mehr durch standardisierte 1-Byte Zeichensätze der ISO-8859 Gruppe darstellen und es müsste auf eine Unicode[9] Kodierung ausgewichen werden. Die konkrete Kodierung (UTF-8, -16, -32, UCS-2, etc.) muss dann selbstverständlich im Format exakt spezifiziert werden. In obigem hexadezimalen Paketauszug lautet der Benutzername egoistischerweise „Matthias“ und enthält ausschließlich ASCII Zeichen.

Das Passwort wird wie bereits beschrieben in Form eines Streuwertes nach dem Nachrichtentyp eingefügt und belegt zumindest bei der Wahl des SHA-256 Algorithmus stets 32 Bytes. Wird es von einem die Übertragung abhörenden Dritten abgefangen, wird diesem das ursprüngliche Klartextpasswort damit nicht offenbart. Er kann jedoch den Benutzernamen und Streuwert kopieren und während seiner eigenen Anmeldung einspeisen, um sich mit der Identität des Belauschten zu authentifizieren. Dieses Problem kann nur durch die Verwendung von Zertifikaten gelöst werden.

[9] http://www.unicode.org

Der kontaktierte PingGate Proxy überprüft schließlich die Gültigkeit der Anmeldedaten. Diesem müssen die registrierten Benutzer und deren Passwörter (in Streuwerte transformiert) vorliegen. In der Implementierung ist dafür eine Schnittstelle vorgesehen, über die unterschiedliche Datenbanken anhand spezialisierter Module angesprochen werden können. Die einfachste Form einer solchen Datenbank ist eine Textdatei mit einer zweispaltigen Liste, wobei die erste Spalte den Benutzernamen und getrennt durch ein bestimmtes Zeichen, welches in einem Namen nicht vorkommen darf, die zweite Spalte das Passwort enthält. Wenn als Trennzeichen ein Komma gewählt wird, entspricht dies dem CSV Format [46]. Mit Tabellenkalkulationsprogrammen, wie *Calc* aus der OpenOffice Suite oder Microsofts *Excel*, können Daten in diesem Format bequem verwaltet und exportiert werden.

Bei gehobeneren Ansprüchen ist auch die Auslagerung der Daten in relationale Datenbanken oder LDAP[10] Verzeichnisdiensten möglich, wofür die PingGate Proxy Implementierung ebenfalls Module bereitstellt. Die Verwendung einer MySQL Datenbank zur Speicherung der Benutzerinformationen bietet sich beispielsweise auf einem typischen LAMP[11] System an, wenn der Betreiber eines Datentunnels per Webinterface einzelne Benutzer (Kunden) freischalten und verwalten möchte.

Das Ergebnis der Überprüfung einer Anmeldeanfrage – mit welchem Modul auch immer – ist nicht unbedingt binärer Natur, sondern strenggenommen eine von drei Möglichkeiten:

1. Benutzername und Passwort stimmen überein und der Zugriff wird gestattet.
2. Benutzername und/oder Passwort sind ungültig und der Zugriff wird nicht gestattet.
3. Bei der Anmeldung ist ein Fehler aufgetreten, weshalb der Zugriff nicht gestattet wird.

Die Miteinbeziehung der dritten Möglichkeit ist aus usability Gründen auf der Seite des PingGate Clients sinnvoll, weil die Alternative des Anzeigens der Ungültigkeitsmeldung zur Verwirrung des Benutzers beiträgt und gegebenenfalls zu verzweifelten Änderungsversuchen des Passworts führt, sofern ein solcher Mechanismus zur Verfügung gestellt wird.

Ordnet man den drei Möglichkeiten die Zahlen 1–3 zu, dann sieht das betreffende Paket zur Rückmeldung des Authentifizierungsergebnisses so aus:

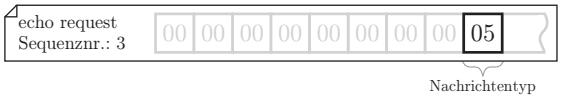

Nach einer erfolgreichen Anmeldung können nun die eigentlichen Datenverbindungen hergestellt werden. Da aber der Anmeldevorgang unmittelbar

[10] **L**ightweight **D**irectory **A**ccess **P**rotocol
[11] Ein System mit **L**inux, **A**pache, **M**ySQL und **P**HP als Standardausstattung

nach dem Start der PingGate Client Anwendung erfolgt und bis zum Ansprechen der SOCKS Schnittstelle durch einzelne Programme einige Zeit verstreichen kann, ist das im Flussdiagramm 3.9 dargestellte, periodische Austauschen von Phantomdaten ohne einer einzigen Datenverbindung, unnötig. Höchstens ein „keep alive“ Mechanismus kann in diesem Fall sinnvoll sein. Dabei wird mit deutlich verminderter Frequenz im Vergleich zum Betrieb mit aktiven Verbindungen eine Informationshülse übertragen. Diese dient nur zur Signalisierung, dass die Client Anwendung noch läuft und jederzeit aktiv werden kann.

In einer Situation jedoch, wo jeder unnötige Datentransfer vermieden werden soll und etwaige Zeitüberschreitungseinstellungen großzügig gewählt worden sind, kann die permanente, wechselseitige Paketübermittlung durch eine spezielle Leerlaufnachricht gestoppt werden. Diese wird in der Empfangseinheit des PingGate Proxies gesondert behandelt und führt dazu, dass die periodische Übermittlung der Anmeldebestätigung eingestellt wird und außerdem auf das Leerlaufpaket ausnahmsweise *keine* Antwort erfolgt. Optimalerweise sendet der PingGate Client die Leerlaufnachricht einmal, worauf diese vom PingGate Proxy empfangen und entsprechend verarbeitet wird. Sendet dieser aber weiterhin die Antwort des vorherigen Pakets, ist die Nachricht offenbar verloren gegangen und muss erneut übertragen werden. Die Leerlaufnachricht wird also geduldig auf jede Antwort des PingGate Proxies gesendet, bis schließlich keine mehr empfangen wird.

Das Paket zur Signalisierung des Leerlaufs ist unspektakulär und enthält außer der Nummer des Nachrichtentyps keine weiteren Daten:




### 3.5.2 Verbindungsanfrage

Nach der Signalisierung des Leerlaufmodus können jederzeit Verbindungsanfragen verarbeitet werden, die über die SOCKS Schnittstelle eingehen. Wegen der Möglichkeit der parallelen Benutzung durch einzelne Anwendungen, muss auf die Unterstützung des Managements gleichzeitiger Verbindungen besonderes Augenmerk gelegt werden.

Der Flexibilitätsbegriff im Titel dieser Arbeit kommt an genau dieser Stelle zu tragen. Das folgende Verfahren erlaubt im Gegensatz zur starren Punkt-zu-Punkt-Kommunikation, das flexible Auf- und Abbauen mehrerer gleichzeitigen Verbindungen zu beliebigen Zielen. Der diesbezügliche Anfragevorgang involviert drei unterschiedliche Nachrichtentypen. Zum einen muss dem Umstand Rechnung getragen werden, dass der Vorgang eine gewisse Zeit in Anspruch nimmt und zum Anderen, dass die Verbindungsanfrage fehlschlägt.

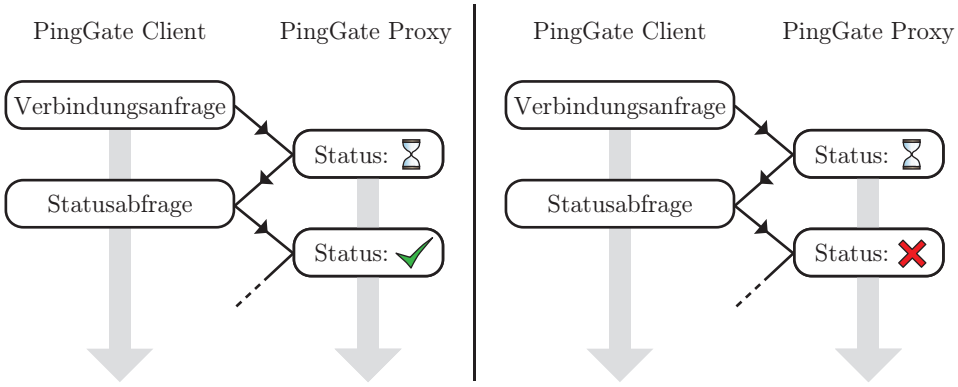
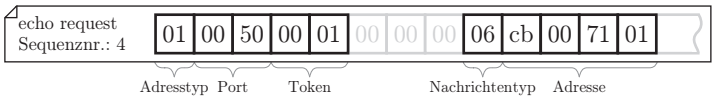



**Abbildung 3.12:** Die zwei möglichen Szenarien (Erfolg und Misserfolg) beim Veruch der Herstellung einer Verbindung.

Abbildung 3.12 zeigt die zwei möglichen Verläufe eines Anfragevorgangs. Im dargestellten Ablauf ist das Konzept der kontinuierlichen Statusabfrage angedeutet. Es wird solange eine Statusabfrage seitens des PingGate Clients gesendet, bis ein Resultat die Verbindung betreffend verfügbar ist.

Das Format des Pakets zur Initiierung einer neuen Verbindung und zum in Gang setzen des beschriebenen Ablaufs, ist wie folgt definiert:




Die Angabe des Verbindungsziels kann – genauso wie auch vom SOCKS Protokoll der Version 5 vorgesehen – auf drei Arten erfolgen:

1. IPv4 Adresse (z.B. 203.0.113.1)
2. IPv6 Adresse (z.B. 2001:db8:53fa:b74:8e77::5)
3. Hostname (z.B. www.example.com)

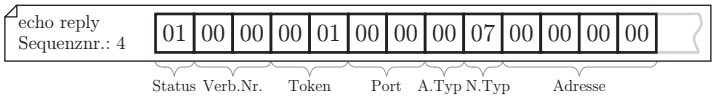
Der „Adresstyp“ als erstes Byte, gibt an, wie die nach dem Nachrichtentyp folgende „Adresse“ zu interpretieren ist. Eine IPv4 und IPv6 Adresse ist stets 4 bzw. 16 Bytes breit, aber im Fall der Angabe eines Hostnamens muss dessen Länge spezifiziert werden. Dies lässt sich beispielsweise durch die Verwendung des ersten Bytes der „Adresse“ als Längenfeld einfach realisieren.

Neben der Zieladresse ist freilich auch der zu verwendende Port als 16 Bit Wert anzugeben.

Eine Besonderheit stellt der „Token“ dar, bei dem es sich um eine vom PingGate Client wählbare Nummer handelt. Diese muss für jede Verbindung eindeutig sein, welche sich gerade im Aufbau befindet. Typischerweise dürfte der Token als Zählvariable implementiert werden, die nach jedem Vorgang inkrementiert wird. Der Sinn des Tokens ist die Zuweisung einer Nummer zu jedem aktiven Verbindungsaufbau, damit die folgenden Statusinformationen

richtig zugeordnet werden können. Nachdem die initiale Verbindungsanfrage vom PingGate Client ausgeht, muss dieser auch die Nummer vorgeben.

Die Antwort auf eine solche Anfrage erfolgt schließlich in Form einer Statusmeldung vom PingGate Proxy. Das Format eines solchen Pakets sieht folgendermaßen aus:




Vom Statusindikator hängt die Interpretation der Belegung der übrigen Datenfelder des Pakets ab. Der Indikator gibt an, welches der folgenden Ereignisse zutrifft:

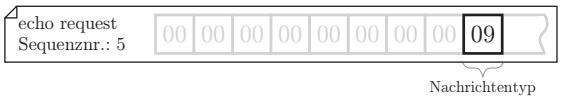
1. Verbindungsaufbau im Gange
2. Verbindung hergestellt
3. Verbindung fehlgeschlagen

Die Liste kann mit feiner abgestuften Fehlergründen erweitert werden (vgl. SOCKS Fehlercodes in [26, S. 5]). Die Fehlerfälle hängen auch teilweise vom verwendeten Adresstyp ab. So kann zum Beispiel die Auflösung eines angegebenen Hostnames in eine IP Adresse und nicht der eigentliche Verbindungsversuch fehlschlagen.

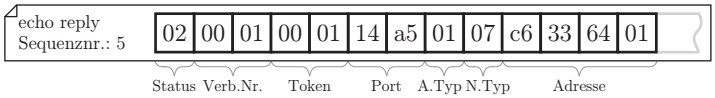
Die „Verbindungsnummer“ ist erst relevant, sobald die Verbindung hergestellt werden konnte und hat ansonsten den Wert Null. Im Gegensatz zum Token wird sie vom PingGate Proxy festgelegt und ersetzt im weiteren Kommunikationsverlauf den Token, dessen Zahlenwert als temporärer Bezeichner wieder frei wird und dadurch vom PingGate Client für zukünftige Verbindungsanfragen wiederverwendet werden kann.

Neben der Verbindungsnummer wird der in der Anfrage angegebene Token wieder in der Statusmeldung zurückgesendet, damit clientseitig die Assoziierbarkeit der Meldung mit einer konkreten Anfrage gegeben ist. Die Felder „Port“, „Adresstyp“ und „Adresse“ werden nur in der Nachricht über eine erfolgreich aufgebaute Verbindung interpretiert und sind ansonsten unbelegt, haben also effektiv den Wert Null.

Wegen etwaiger Namensauflösung, geringen Bandbreiten und überlasteten Servern, kann der Verbindungsvorgang eine Weile dauern, weshalb es durchaus mehrfach zu einer Benachrichtigung bezüglich des im Gange befindlichen Verbindungsaufbaus kommen kann, bevor die Erfolgs- oder Misserfolgsmeldung eintrifft. Zum Einholen des aktuellen Status, ist ein eigenes Paket vorgesehen. In dessen Format sind außer dem Nachrichtentyp selbst, keine weiteren Datenfelder vorgesehen:



Konnte schließlich die Verbindung hergestellt werden, sind der Statusindikator und die übrigen Felder entsprechend belegt:




Bei der Portnummer und Adresse handelt es sich jedoch nicht um die Daten bezogen auf den Zielrechner, denn diese sind der PingGate Client Ausführung ohnehin bekannt. Es ist vielmehr der Socket am PingGate Proxy als lokaler Endpunkt der TCP Verbindung zum Zielrechner. Zur vollumfänglichen Nutzung mancher höherer Übertragungsprotokolle wie zum Beispiel FTP, müssen einer Client-Anwendung diese Daten bekannt sein.

Die Möglichkeit der Angabe eines Hostnamens statt einer IP Adresse als Verbindungsziel hat im Übrigen einen besonderen Nebeneffekt. Hinter Hostnamen können sich nämlich sowohl sog. „A“ als auch „AAAA“ Einträge verbergen, wobei ersterer eine IPv4 Adresse und zweiterer eine IPv6 Adresse beschreibt [29, 54]. Somit besteht die Möglichkeit, dass eine PingGate Client Instanz ohne unmittelbare IPv6 Konnektivität, trotzdem einen nur per IPv6 zugänglichen Server erreichen kann, sofern am Rechner des PingGate Proxies ein entsprechender Zugriff möglich ist. Das „Adresstyp“ Feld unterscheidet sich im Fall der Angabe eines Hostnamens vom Feld der Verbindungsanfrage. Sollte der Name zu einer IPv4 Adresse aufgelöst worden sein, dann wird im Adressfeld eine externe IPv4 Schnittstelle des PingGate Proxy Rechners eingetragen, andernfalls eine entsprechende IPv6 Schnittstelle. PingGate kann somit als Brücke zwischen unterschiedlichen IP Versionen dienen.

### 3.5.3 Datentransfer

Nachdem eine Verbindung hergestellt und dies mit einem entsprechenden Statuspaket dem PingGate Client signalisiert wurde, kann unmittelbar mit dem Übertragen von den eigentlichen Nutzdaten begonnen werden. Ab diesem Punkt ist zur Assoziation der Daten mit einer konkreten Verbindung, die „Verbindungsnummer“ aus der Statusnachricht erforderlich und muss mit jedem Datenpaket mitgeliefert werden.

Abbildung 3.13 zeigt exemplarisch den Ablauf zweier Verbindungen, wobei zuerst reguläre Datenpakete ausgetauscht werden und schließlich die Trennung der Verbindung erfolgt.

Für den Transport der Nutzdaten sind zwei Nachrichtentypen vorgesehen, wobei sich deren Format gleicht. Aus Gründen der Erweiterungsfreundlichkeit, Ausdrücklichkeit und Konsistenz, wird auf die Verwendung eines gemeinsamen Nachrichtentyps für beide Kommunikationsrichtungen verzichtet. Für die Übertragung von Daten vom PingGate Client zum PingGate Proxy, sieht das „Client-Daten“ Paketformat folgendermaßen aus:

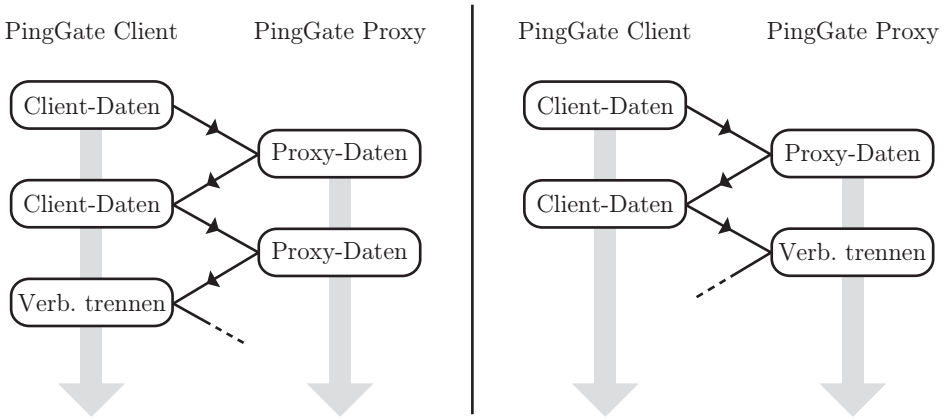
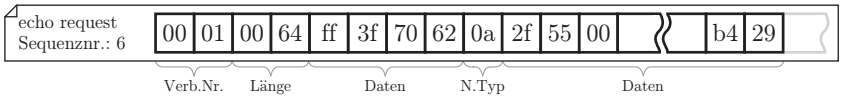
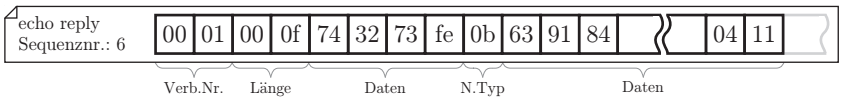



**Abbildung 3.13:** Ablauf der Nachrichtenübermittlung während einer Verbindung. Links erfolgt die Trennung der Verbindung von der PingGate Client Instanz und rechts vom PingGate Proxy aus.




In die andere Richtung, also vom PingGate Proxy zum PingGate Client als Antwort auf das „Client-Daten“ Paket, folgt das „Proxy-Daten“ Paket:




Nach der Verbindungsnummer ist in einem Längenfeld die Anzahl der Bytes eingetragen, die in den beiden als „Daten“ bezeichneten Bereichen des Pakets insgesamt gespeichert sind. Um möglichst keinen Platz zu verschwenden, werden auch die vier Bytes zwischen dem Längenfeld und dem des Nachrichtentyps verwendet.

Bezüglich der Längeninformation $L$ für die eingebetteten Nutzdaten gilt, dass $0 <= L <= Paketgr\ddot{o}\beta e - 5$ ist. Eine Länge von Null ist in all jenen Fällen von Nöten, in denen keine weiteren Nutzdaten vorliegen, aber trotzdem ein Client- oder Proxy-Daten Paket gesendet werden muss. Die PingGate Client Instanz befindet sich in dieser Situation, wenn das Senden der nächsten Nachricht im Zuge des periodischen Pollings ansteht. Der PingGate Proxy hat umgekehrt ein solch leeres Datenpaket zu senden, wenn er vom PingGate Client ein Client-Daten Paket erhalten hat, über die TCP Verbindung zum Zielrechner aber keine neuen Informationen eingegangen sind, die an dieser Stelle zurückgesendet werden müssten.

Schließt eine Anwendung die Verbindung zur SOCKS Schnittstelle oder ein Zielrechner die TCP Verbindung zum PingGate Proxy, muss dieses Ereig-

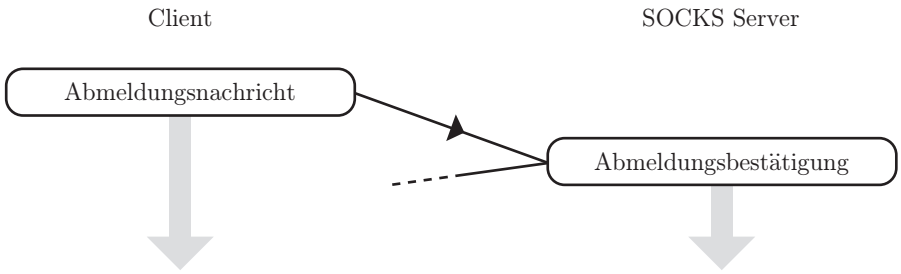
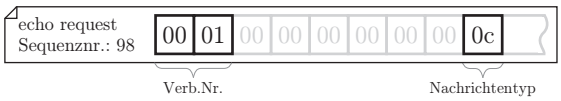


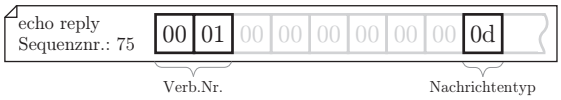

**Abbildung 3.14:** Der Abmeldungsvorgang erfolgt durch eine Bekanntgabe und dem Erhalt einer Bestätigung.

nis in Form einer dafür vorgesehenen Terminierungsnachricht dem jeweiligen Gegenüber mitgeteilt werden. Die Situation ist dieselbe wie im Fall der zuvor behandelten Datenpakete, wo für beide Seiten ein eigener Nachrichtentyp definiert wurde, obwohl das Format sich gleicht.

Im Fall der Verbindungstrennung seitens PingGate Client wird satt dem nächsten Client-Daten Paket, das folgende gesendet:




Umgekehrt wird bei der Terminierung einer Verbindung am PingGate Proxy statt der nächsten Proxy-Daten Antwort die entsprechende Meldung über die Verbindungstrennung erstattet:




Kommt ein solches Paket an, wird beim Empfänger, die mit der im Paket angegebenen Verbindungsnummer assoziierte TCP Verbindung geschlossen und damit ein beidseitige Einstellung der Kommunikation vollzogen.

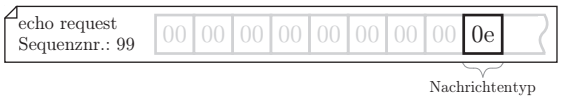
Die, in den beiden abgebildeten Paketauszügen verwendeten Sequenznummern 98 und 75 wurden frei gewählt, um beispielhaft zu verdeutlichen, dass vor den Terminierungsnachrichten ein regulärer Datenaustausch stattfand, bevor es schließlich zum Abbruch kam.

### 3.5.4 Abmeldung

Der zur vorherigen Anmeldung entgegengesetzte Vorgang der Abmeldung ist in Abbildung 3.14 dargestellt. Die Abmeldung wir bei der Beendigung einer PingGate Client Instanz durchgeführt, damit der PingGate Proxy den für die Instanz erstellten Kontext unverzüglich löschen und sonstige verknüpfte Ressourcen freigeben kann.

Erfolgt die Abmeldung nicht ordnungsgemäß, wird eine bestimmte (einstellbare) Zeitspanne gewartet und bei ausbleibenden Reaktionen der Kontext am PingGate Proxy ebenfalls gelöscht.

Das Datenpaket zur Signalisierung des Abmeldung kann nach erfolgreicher Anmeldung zu jedem Zeitpunkt vom PingGate Client gesendet werden und wird vom PingGate Proxy gesondert und priorisiert behandelt. Das Format des Pakets ist trivial:




Unmittelbar nach dem Eingang der Abmeldenachricht, werden am PingGate Proxy sämtliche bestehende Verbindungen getrennt, die mit dem abzumeldenden Kontext assoziiert sind. Als Antwort folgt die Bestätigung in Form einer Entlassungsnachricht:


Anhand dieser erkennt die PingGate Client Instanz, dass die Abmeldung erfolgreich abgeschlossen wurde und kann in der Folge ordentlich terminieren. Sollte die Entlassungsnachricht innerhalb eines gewissen Zeitfensters nicht eintreffen, wird das Programm trotzdem beendet und die etwaige Ressourcenbelegung bis zur Zeitüberschreitung am PingGate Proxy und der darauf folgenden automatischen Abmeldung, in Kauf genommen.

Der PingGate Proxy hat keine Möglichkeit zu erkennen, ob dieses abschließende Paket angekommen ist, weil das Protkoll für die finale Verabschiedung keine Bestätigung vorsieht. Diese bedürfte schließlich wieder einer Bestätigung der anderen Seite, sodass es zu einer endlosen Bestätigung von Bestätigungen kommen würde. Um die Zustellung der Entlassungsnachricht möglichst zu gewährleisten, wird sie deshalb vom PingGate Proxy innerhalb kurzer Abstände mehrfach gesendet.

## 3.6 Demultiplexing

Nachdem nun die Kommunikationsvorgänge auf der Seite des PingGate Clients über die SOCKS Schnittstelle beschrieben wurden, sowie die Datenübertragung in Form von ICMP Echo Anfragen und Antworten zwischen einzelnen PingGate Instanzen, verbleibt nun zur Komplettierung des Systems die Beschreibung des Verbindungsmanagements des PingGate Proxies.

Abbildung 3.15 zeigt als dritten Ausschnitt der Topologie die Kommunikation des PingGate Proxies mit Zielrechnern im Internet.

Die sequentiell übermittelten Verbindungsanfragen und -daten müssen bestimmten angemeldeten PingGate Client Instanzen zugeordnet werden

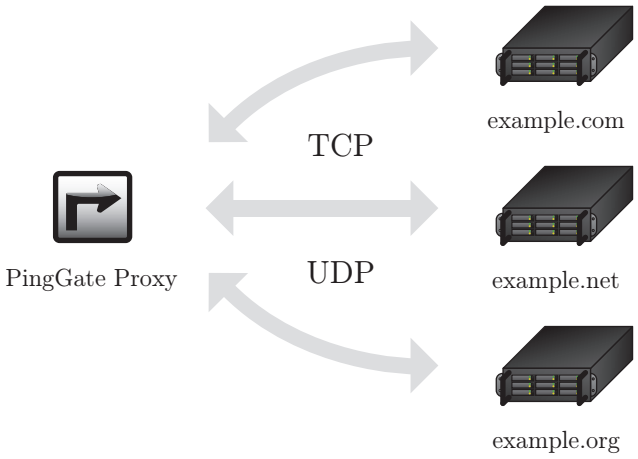



**Abbildung 3.15:** Jede PingGate Proxy Instanz kommuniziert mit einer theoretisch beliebigen Anzahl an Zielrechnern über das TCP und gegebenenfalls auch UDP Protokoll.

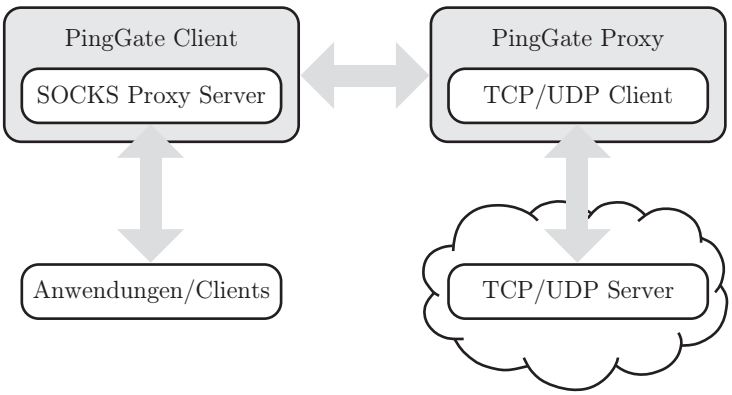



**Abbildung 3.16:** Die Beziehungen der mit „Client“, „Server“ und „Proxy“ bezeichneten Komponenten zueinander.

können. Dazu wird während einer Sitzung (der Dauer zwischen An- und Abmeldung) für jeden Benutzer ein Kontext erstellt, mit dem die einzelnen Verbindungen zu Zielrechnern im Internet assoziiert werden. Ein Paar aus IP Adresse und „Identifier“ Wert des ICMP headers (siehe Abbildung 2.6) bezeichnet dabei einen solchen Kontext.

Zusammenfassend soll Abbildung 3.16 vor allem bezüglich der Terminologie Klarheit schaffen. Ein möglicherweise zur Verwirrung beitragender Umstand ist, dass PingGate Client mit der SOCKS Schnittstelle auch eine Serverfunktionalität inne hat, während PingGate Proxy im Zuge der Verbindungsherstellung zu Zielrechnern im Internet, die Clientrolle einnimmt.

# Kapitel 4

# Alternative Ansätze

Das Konzept der Datentunnelung lässt sich mit verschiedenen Protokollen realisieren. Im Rahmen dieser Arbeit ist primär das ICMP von Interesse, deshalb werden in der Folge Alternativen zur vorgestellten Architektur aus Kapitel 3 erläutert. Außerdem erfolgt ein Ausblick auf steganographische Möglichkeiten im TCP, sowie in übergeordneten Protokollen wie dem HTTP oder DNS.

## 4.1 Datentunnel per ICMP

Die vorgestellte Tunnelungstechnik ist auf Schicht 4, also der Transportschicht angesiedelt. Abschnitt 3.2 beschreibt diesbezüglich die Vorteile im Vergleich zur Vermittlungsschicht. In den folgenden Sektionen wird sowohl ein zu PingGate sehr ähnliches Projekt auf der Transportschicht vorgestellt, als auch eine Gruppe von Implementierungen, die auf der Vermittlungsschicht operieren.

### 4.1.1 TCP over ICMP

Daniel Stødle stellte in [50] seine Implementierung eines Werkzeugs zur Datentunnelung per ICMP namens „Ping Tunnel“ vor. Es handelt sich dabei um eine Software zur Erstellung einer Portweiterleitung für TCP Verbindungen. Im Gegensatz zu PingGate, das eine integrierte Schnittstelle zur dynamischen Adressierung beliebiger Verbindungsziele enthält, kann „Ping Tunnel“ pro Ausführung lediglich ein konkretes Ziel ansprechen, welches außerdem beim Start der Serveranwendung am externen Proxy Computer anzugeben ist. Das Client und Proxy Konzept ist diesbezüglich dasselbe wie bei PingGate.

Zur Erweiterung der Funktionalität und Erreichung derselben Flexibilität wie in der PingGate Architektur, können wiederum höhere Protokolle über diese einzelne Verbindung getunnelt werden, welche dann für vielfäl-

tigere Nutzungsmöglichkeiten sorgen. Konkret ist an dieser Stelle auf SSH[1] zu verweisen. Entsprechend konfiguriert, kann damit ebenfalls eine SOCKS Schnittstelle zur Verfügung gestellt werden. Ein entscheidender Vorteil liegt bei der Verwendung von SSH in der Tatsache, dass damit automatisch eine zuverlässige Verschlüsselung des Datenverkehrs möglich ist. Nachteilig wirken sich die Anforderungen (Schlüsselaustausch, Protokolloverhead) des SSH Protokolls selbst aus, sodass bei einem sehr schmalbandigen Datentunnel aufgrund einer Zeitüberschreitung keine Verbindung zu Stande kommen und damit die Tunnelungssoftware nicht genutzt werden kann.

Ein „windowing“ Konzept sorgt für die schnellere Übertragung von Daten in jeweils eine Senderichtung, indem mehrere Pakete übertragen werden und die Bestätigung (Echo Antwort) gesammelt für diese Paketgruppe erfolgt. In PingGate wird dagegen nach jeder Anfrage auf eine entsprechende Antwort gewartet. Das hat den Vorteil, dass die Kapazität des Kanals in Sende- und Empfangsrichtung dieselbe ist. Aus Implementierungssicht erfordert „windowing“ eine komplexere Logik, was die optimale Wahl der Größe des Übertragungsfensters und das Verhalten im Fehlerfall betrifft.

Wie bei PingGate, ist beim „Ping Tunnel“ außerdem eine rudimentäre Authentifizierungsmöglichkeit mit Benutzername und Kennwort vorgesehen.

### 4.1.2 IP over ICMP

Beim Transport von Daten auf der Vermittlungsschicht fällt im Gegensatz zur Transportschicht die Notwendigkeit zur Emulierung diverser Protokolleigenschaften und überhaupt die Unterscheidung höherer Protokolle weg. Stattdessen müssen lediglich IP Pakete ohne Beachtung der darin gekapselten Protokolle getunnelt werden. Da diese Strategie eine hohe Flexibilität verspricht, wird sie von den meisten Implementierungen bevorzugt.

Die folgenden Programme unterscheiden sich teilweise stark in deren Funktionsumfang, verfolgen aber alle den selben Ansatz der Tunnelung auf der Vermittlungsschicht.

- **ICMPTX** ist eine von Thomer M. Gil[2] auf Basis des „proof of concept“ Codes aus [35] von Siim Põder entwickelte Software. Põder verwendete wiederum einen, im Jahr 1999 als **itunnel**[3] veröffentlichten, rudimentären Tunnelungsansatz der österreichischen Hackergruppe „Teso“.

  Es wird – genauso wie bei den folgenden drei Implementierungen – eine sog. TUN Schnittstelle als virtuelles Netzwerkgerät erstellt, über das die Kommunikation erfolgt. Problematisch ist dabei die Wahl der MTU (siehe Abschnitt 2.2.1). Für große IP Pakete, welche die eingestellte MTU überschreiten, ist keine vollständige Übertragung möglich.

[1] **S**ecure **Sh**ell, ein Protokoll mit vielseitigen Kommunikationsmöglichkeiten, integrierter Verschlüsselung und SOCKS Schnittstelle. Siehe http://www.openssh.com.

[2] http://thomer.com/icmptx

[3] http://packetstormsecurity.org/files/download/10500/itunnel-1_2.tar.gz

Der Autor beschreibt auf der Projektseite diesbezüglich aber eine Lösungsmöglichkeit, die auf der Einführung eines Fragmentierungsheaders beruht.

- **Hans**[4] basiert auf dem Konzept von ICMPTX und erweitert dessen Funktionalität unter anderem um die Unterstützung mehrerer gleichzeitiger Benutzer und ein Anmeldekonzept. Die Software läuft vollumfänglich unter Linux und zumindest im Clientmodus auch unter Mac OS, Free- und OpenBSD, also unixartigen Systemen mit dem Darwin bzw. BSD Kernel.

  Der Projektname soll übrigens einen Herren bezeichnen, der in den Alpen wohnt und über Echos deshalb bestens Bescheid weiß[5].

- **itun**[6] von Sergey Chvalyuk ist konzeptionell eine zu [35] sehr ähnliche Implementierung, die allerdings um einiges rudimentärer aufgebaut ist und keine erweiterten Funktionen, wie etwa Authentifizierung enthält.

- **YAPT**[7] (**Y**et **A**nother **P**ing **T**unnel) ist eine relativ neue, 2009 erstellte C++ Implementierung. Es ist bisher die einzige, die ICMPv6 und damit IPv6 unterstützt.

### 4.1.3 Kernel module

Bei der Software **Skeeve**[8] handelt es sich um eine vergleichsweise unkonventionelle Implementierung in Form eines Moduls für den Linux Kernel. Sie wurde von Ilya Zelenchuk im Jahr 2004 geschrieben und kann einen auf ICMP basierenden Datentunnel zwischen zwei statisch festzulegende (und einzukompilierenden!) IPv4 Adressen herstellen.

Dabei wird das „Protocol“ header Feld (vgl. Abbildung 2.3) der an die spezifizierte Zieladresse gerichteten IP Pakete auf den Wert für ICMP (1) gesetzt. Nach weiteren Manipulationen des headers wird das Datenpaket im Grunde als ICMP Nachricht maskiert und als solche versendet. Am Proxy Rechner (von Skeeve als „bouncer“ bezeichnet) wird diese Transformierung wieder rückgängig gemacht und das Originalpaket zum eigentlichen Zielrechner weitergeleitet.

Die Implementierung ist auf das Wesentliche reduziert und dient als „proof of concept“ Code. Als Kernelmodul muss die Software genau zur verwendeten Version des Betriebssystems passen und gegebenenfalls für jedes System eigens kompiliert werden, was aufgrund der statisch im Quellcode festgelegten IP Adressen ohnehin des Öfteren nötig sein dürfte.

[4] http://sourceforge.net/projects/hanstunnel
[5] http://code.gerade.org/hans
[6] http://sourceforge.net/projects/itun
[7] http://sourceforge.net/projects/yaping
[8] http://gray-world.net/poc_skeeve.shtml

## 4.2 Datentunnel über andere Protokolle

Die Benutzung des ICMP ist nur eine von vielen Möglichkeiten zum verdeckten Datentransport. Es gibt je nach Anforderung an die Kapazität des Kanals und den Rahmenbedingungen der Einsatzszenarios, auch für andere Protokolle Verfahren zur Einbettung von Zusatzinformation.

### 4.2.1 TCP

Für das TCP auf der Transportschicht gibt es eine Reihe unterschiedlicher Tunnelungsansätze, die auf einer alternativen Nutzung von header Feldern beruhen (vgl. IP aus Abschnitt 2.1).

In [1] ist unter anderem die Möglichkeit der Ausnutzung von Redundanzzuständen einzelner Bits im „Control Bits“ Bereich des TCP header [39, S. 15] beschrieben. Hauptsächlich beim Verbindungsauf- und -abbau werden zu Signalisierungszwecken binäre Schalter verwendet, die in bestimmten Kombinationen zum selben Ergebnis führen und damit als „storage channel“ benutzt werden können.

Außerdem unterstützt das TCP genauso wie das IP Zusatzfelder im Paketheader, die in Form von „Options“ integriert werden können. Eine solche Erweiterung ist die „TCP Timestamps Option“ [20, S. 13]. In [14] wird die Manipulation von niederwertigen Bits solcher Zeitstempel zur Kodierung von Information analysiert. Die Werte dieser Bits sind generell gleichverteilt, weil das Senden von Datenpaketen zu jedem Zeitpunkt ohne fixen Takt stattfinden kann. Wie in [16, S. 31] bezogen auf IP, kann auch hier ein schmalbandiger verdeckter Kanal geschaffen werden.

### 4.2.2 HTTP

Das „Hypertext Transfer Protocol“ ist für Tunnelungszwecke sehr populär, weil es das Standardprotokoll zur Übertragung von Websites im Internet ist und folglich am ehesten zur Verfügung steht. Das HTTP Protokoll bietet mit der sog. CONNECT Methode [13] eine integrierte Proxyunterstützung, die im weiteren Sinne zur Datentunnelung herangezogen werden kann.

Eine etwas subtilere Variante ist jedoch das Transportieren von Daten mittels regulärer GET und POST Methoden. Da per HTTP auch Binärdateien hoch- und heruntergeladen werden können, lassen sich – getarnt als Dateiup- und -download – beliebige Protokolle bidirektional verwenden. Das Werkzeug **httptunntel**[9] implementiert diese Übertragungsstrategie.

Wird in der Umgebung, aus der die Tunnelung von Daten heraus geschehen soll, die Benutzung des HTTPS gestattet, bleibt der „Missbrauch“ der Verbindung zu Tunnelungszwecken verborgen. HTTPS ist eine verschlüsselte Form des HTTP, wodurch ein Abhören und Feststellen etwaiger eingebette-

[9] http://www.nocrew.org/software/httptunnel.html

ter Sekundärprotokolle durch Firewalls oder wachsamen Systemadministratoren, grundsätzlich nicht möglich ist. Die Konfiguration eines Servers zur Unterstützung eines solchen Tunnels mithilfe des Apache Webservers und OpenSSH, ist einfach zu bewerkstelligen[10].

Im Gegensatz zur Datentunnelung anhand IP oder TCP header Feldern oder gar über „timing channels“, sind die Einbußen bzgl. Bandbreite im Gegensatz zu einer Direktverbindung bei HTTP Tunnelung minimal. Dadurch ist dieses Verfahren dort interessant, wo es auf schnelle und umfängliche Datenübertragung ankommt.

Es gibt allerdings auch Alternativen, die einzelne Zusatzfunktionalitäten des Protokolls nutzen. So wird in [5] die Möglichkeit der verdeckten Kommunikation über HTTP-Cookies vorgestellt.

### 4.2.3 DNS

Das „Domain Name System“ dient primär zur Auflösung von Domain Namen, wie z.B. example.com in IP Adressen und umgekehrt. In Umgebungen, in denen per HTTP der Zugang zum Internet gewährt wird, muss fast zwingend auch DNS verfügbar sein. Zur Anwahl einer Website wird schließlich vom Benutzer deren DNS Name anstatt deren dahinterliegenden IP Adresse eingegeben.

Im DNS gibt es unterschiedliche Nachrichtentypen, [30] die zweckentfremdet als Träger für beliebige zu transportierende Daten verwendet werden können. Dabei eignen sich Anfragetypen schlechter als Antworttypen, weshalb bei Tunnelungsprogrammen wie **Iodine**[11] signifikant höhere Download- als Uploadraten erzielt werden können, was aber zumindest für reguläres Surfen im Internet kein Problem darstellt, weil die Übertragungscharakteristik diesem Schema entspricht. Die genannte Software bietet außerdem interessante Zusatzfunktionen, wie Authentifizierung, Mehrbenutzerbetrieb und automatische Bandbreitenregulierung.

Ein Umstand macht die Verwendung eines Tunnels auf DNS Basis in gewissen Situationen attraktiv: Wie in [34, S. 2] beschrieben, ist in Umgebungen mit kosten- oder zumindest registrierungspflichtigem Internetzugang, das Senden und Empfangen von DNS Paketen möglicherweise nicht von vornherein blockiert. Als Begründung ist angeführt, dass ungültige Beantwortungen von DNS Anfragen vor der Bezahlung bzw. Registrierung, im DNS Cache des Betriebssystems verbleiben und nach der ordnungsgemäßen Anmeldung dann zu Auflösungsfehlern führen würde. Dass statt dem Antworten mit ungültigen Inhalten, das schlichte Verwerfen von Anfragen das Problem lösen dürfte, wird nicht in Betracht gezogen.

Eine Zusammenstellung und diverse Performancevergleiche zwischen Implementierungen zur DNS Tunnelung ist ebenfalls in [34] zu finden.

[10] http://dag.wieers.com/howto/ssh-http-tunneling
[11] http://code.kryo.se/iodine

# Kapitel 5

# Diskussion

Die unterschiedlichen Motive für die Nutzung von Datentunneln schaffen jeweils eine Reihe offener Diskussionspunkte. Spielt beispielsweise die Verborgenheit eines Kanals eine zentrale Rolle, drängen sich Fragen nach der Wahrscheinlichkeit einer Entdeckung der Kommunikationsvorgänge durch Dritte auf. Befindet man sich aber in der Position eines Netzwerkadministrators, der in einem Konzern eine große Verantwortung bezüglich der technischen Absicherung gegen Wirtschaftsspionage und Computerviren hat, ist plötzlich die Abwehrmöglichkeit von ungebetenen verdeckten Kanälen von höchster Bedeutung. Die folgenden Abschnitte gehen auf berechtigte Fragestellungen dieser Art näher ein.

## 5.1 Entdeck- und Abwehrbarkeit

Beim Entwurf eines steganographischen Verfahrens, ist die Bewertung der Entdeckbarkeit als Faktor dessen Güte, eine zwingende Notwendigkeit.

Simmons konstruierte 1983 in [48] eine Situation, in der zwei in getrennten Zellen sitzende Gefangene Botschaften auf Zetteln austauschen, indem ein Wärter als Überbringer der Nachrichten fungiert. Da sie potentiell Ausbruchspläne schmieden, liegt dem Wärter nun etwas daran, die Botschaften verstehen und gegebenenfalls beeinflussen zu können.

Diese Metapher des Gefängniswärters hat sich seither in der *Steganalyse*[1] in Form zweier Bezeichnungen manifestiert:

- aktive Wächter („active wardens“)
- passive Wächter („passive wardens“)

Während ein passiver Wächter Kommunikationsvorgänge lediglich beobachtet und im Fall eines ausgespürten, verdeckten Kanals einer übergeordnete Instanz Bescheid gibt, greift der aktive Wächter ins Geschehen ein

[1] Wortkombination aus „Steganographie“ und „Analyse“ zur Bezeichnung des Forschungsgebiets über die Untersuchung steganographischer Verfahren.

und versucht beispielsweise durch bewusst eingespeistes Rauschen oder der Erzwingung bestimmter Formate in fehlertoleranten Protokollen, verdeckte Kanäle zu verhindern [3, S. 6].

Wächter dieser Art können im Kontext eines Computersystems sog. „Intrusion Detection/Prevention“ Systeme (IDS/IPS) sein. Ein Beispiel dafür ist *Snort*[2], eine Open Source Software die Netzwerkverkehr im Stil eines „packet sniffers“ (vgl. Abschnitt 2.2.1) abhören, analysieren und darauf reagieren kann. Ausgeliefert wird dieses Programm unter anderem mit einer „icmp.rules“ Datei, welche vordefinierte Regelsätze speziell für ICMP Datenverkehr enthält. Einer davon lautet:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large ICMP
    ➥Packet"; dsize:>800; reference:arachnids,246; classtype:bad-
    ➥unknown; sid:499; rev:4;)
```

Der `dsize` Parameter ist an dieser Stelle interessant, weil dadurch alle ICMP Pakete verworfen werden, die größer als 800 Bytes sind. PingGate könnte also mit der standardmäßig auf 1024 Bytes festgelegten Paketgröße in einer Umgebung in der Snort mit diesem Regelsatz eingesetzt wird, ohne manuelle Verringerung des Wertes nicht betrieben werden. Das in Abschnitt 4.2.3 beschriebene Tunnelungsprogramm auf DNS Basis, bietet für genau solche Situationen die Möglichkeit der automatischen Ermittlung einer funktionierenden Paketgröße. Beginnend mit der MTU, lässt sich durch die schrittweise Verringerung bis zur erfolgreichen Übertragung, die maximal zugelassene Größe leicht bestimmen. Diese Funktion wäre als Erweiterung auch für PingGate denkbar.

Die Entdeck- und Abwehrbarkeit einer Tunnelungsanstrengung ist immer dann besonders einfach, wenn die Kommunikationsteilnehmer ein Protokoll verwenden, dessen Struktur anhand vordefinierter Formate bekannt ist und von einem Beobachter (Wärter) verglichen und verifiziert werden kann. Basiert ein Kanal beispielsweise auf der Verwendung nicht belegter und explizit als „für zukünftige Verwendung“ reservierter Datenbereiche in einem Protokoll, dann tut ein aktiver Wärter gut daran, dafür zu sorgen, dass diese Felder auch stets mit einem vorgesehenen Standardwert (oftmals Null) befüllt sind. Durch einen solchen Eingriff würde die Funktionalität des verwendeten Protokolls nicht eingeschränkt, die Möglichkeit der Nutzung von reservierten Bereichen zur Tunnelung aber vollständig eliminiert werden.

Beim Verstecken einzelner Bits in Trägerdaten, die eine gewisse Genauigkeitstoleranz aufweisen, ist das Aufspüren deutlich schwieriger. Die Herausforderung besteht darin, etwaige eingebettete Daten vom Signalrauschen zu unterscheiden. Wenn beispielsweise die niederwertigsten Bits eines Zeitstempels ursprünglich zufällig verteilt sind (vgl. „TCP Timestamps Option“ aus

[2] http://www.snort.org

Abschnitt 4.2.1), können sich durch eine Manipulation tendenziöse Verschiebungen der Verteilung abzeichnen. Das Vorhandensein einer steganographischen Informationsübermittlung ist dann denkbar.

Zur Bewertung solcher Unregelmäßigkeiten kommen statistische Verfahren zum Einsatz. In [49] wird per „Support Vector Machine“ eine Möglichkeit zur Aufspürung von verdeckten Kanälen in ICMP Echo Nachrichten vorgestellt und evaluiert. PingGate wäre davon unmittelbar betroffen.

Speziell bei auf ICMP Echo Nachrichten basierenden Datentunneln, ist bei einer entsprechenden Kapazität des Kanals das unnatürlich massenhafte Auftreten von einschlägigen Paketen im Netzwerk verdächtig. Das alleinige hochfrequente Eintreffen solcher Datenpakete kann entweder eine automatisierte Intervention einer Firewall auslösen oder aufmerksame Netzwerkadministratoren alarmieren, ohne dass der eigentliche Paketinhalt, mit möglicherweise steganographisch gut versteckten Daten, überhaupt analysiert wird.

Um die Möglichkeit des „Missbrauchs“ von ICMP für Datentunnelung von vornherein einzuschränken, listet Stuart Thomas in [51] folgende Punkte:

- Generell penible Sicherheitsrichtlinien im Unternehmen, die alle Mitarbeiter, Arbeitsplätze und -vorgänge einschließen.
- Einsatz von Firewalls zur restriktiven Zugangsregelung für Benutzer und Dienste.
- Einsatz von IDS als Ergänzung zur Firewall mit entsprechenden Regelsätzen zur besonderen Wachsamkeit bezüglich verdeckter Kanäle.
- Ausgiebige Protokollierung und Archivierung der Vorgänge im Netzwerk zur späteren Analysemöglichkeit im Schadensfall.
- Das über die Standardmaßnahmen hinausgehende Absichern der Infrastruktur („server hardening“).
- Regelmäßiges Überprüfen von Software und Hardware auf Sicherheitslücken, sowie sofortiges Einspielen von Updates nach Bekanntwerden von solchen.
- Informieren und Trainieren von Mitarbeitern, um ein allgemeines Bewusstsein für sicherheitsrelevante Themen zu schaffen.

Verdeckte Kanäle können aber in heutigen vernetzten Computersystemen auch mit noch so großen Anstrengungen im Endeffekt nie vollständig eliminiert werden [32, S. 1]. Es ist nur eine Frage der benötigten Kapazität des Kanals und des zulässigen zeitlichen Rahmens für Kommunikationsvorgänge, ob und wie wahrscheinlich eine Kompromittierung stattfinden kann.

Fängt beispielsweise die geheim platzierte Wanze im von Wirtschaftspionage gebeutelten Konzern das acht Zeichen lange Passwort des Firmenchefs ab, reicht die tägliche Übertragung eines einzelnen Bits aus, damit nach etwa zwei Monaten der Konkurrent Zugriff auf Firmeninterna erlangen kann. In diesem Extrembeispiel würde die Zeitspanne und schmale Bandbreite des benötigten Kanals, eine Entdeckung desselben de facto unmöglich machen.

## 5.2 Verschlüsselung

Bei dem vorgestellten Tunnelungskonzept im Kapitel 3 ist keine protokollimmanente Verschlüsselung vorgesehen. Das PingGate Protokoll soll möglichst auf das Wesentliche – den Transport von Nutzdaten – optimiert sein und dadurch in gewissen Situationen Vorteile bieten. Während des persönlichen Experimentierens mit Stødles „Ping Tunnel“ über einen schmalbandigen Internetzugang zeigte sich beispielsweise, dass eine Konsolenverbindung per Telnet erfolgreich hergestellt werden konnte, eine SSH Sitzung aber aufgrund einer Zeitüberschreitung während der Initialisierung fehlschlug. Die gänzlich ohne Verschlüsselung betriebene Telnet Verbindung konnte also bei den sehr beschränkten Rahmenbedinungen gerade noch getunnelt werden, die Anforderungen der Datenübertragung im Zuge des Diffie-Hellmann Schlüsseltausches bei der SSH Anmeldung, überschritten jedoch die zur Verfügung stehenden Ressourcen bereits. Aus diesem Grund wird eine integrierte und zwingende Verschlüsselung im Transportprotokoll vermieden.

Nur weil in diesem selbst keine Kryptographie vorgesehen ist, muss das aber nicht den generellen Verzicht darauf bedeuten. Es ist hierbei zwischen folgenden Prinzipien zu unterscheiden:

- Inhärente Kryptographie
- Delegierte Verschlüsselung

Insbesondere wenn asymmetrische Kryptographie in das Transportprotokoll der Tunnelungssoftware selbst integriert wird, steigt die Komplexität des Protokolls, die Notwendigkeit der Übertragung von Aushandlungsdaten und damit einhergehend die Fehleranfälligkeit, erheblich. Etablierte Verfahren wie TLS stehen allerdings in Form freier Codebibliotheken zur Verfügung und können die Ver- und Entschlüsselungsvorgänge handhaben.

Der alternative Ansatz ist die Delegierung etwaiger Verschlüsselung an die getunnelten Protokolle. Das Transportprotokoll selbst bleibt somit schlank und frei von Verwaltungsdaten für kryptographische Verfahren, die möglicherweise gar nicht benötigt werden und in manchen Situationen dafür sorgen, dass überhaupt keine Tunnelung zustande kommen kann. Mithilfe des flexiblen SSH Protokolls, lässt sich trotzdem bei Bedarf eine kryptographische Kapselung der Nutzdaten erreichen.

Ist beispielsweise die SOCKS Schnittstelle einer PingGate Client Ausführung auf Port 9090 zu erreichen und soll eine Verbindung zu einem Webserver unter example.com hergestellt werden (wobei ein verfügbarer SSH Server unter 203.0.113.1 angenommen wird), dann lässt sich eine Verschlüsselung der Datentunnelung folgendermaßen erzielen:

```
ssh -o ProxyCommand="connect-proxy -S 127.0.0.1:9090 %h %p" -L 8080:
    ➥example.com:80 203.0.113.1
```

Im Webbrowser muss dann statt example.com, die Adresse localhost:8080 eingegeben werden, um die gewünschte Seite getunnelt zu erreichen. Das als „ProxyCommand“ angegebene Programm „connect-proxy“ ist im OpenSSH Paket übrigens nicht enthalten und muss separat bezogen werden[3].

Es wäre auch denkbar, PingGate mit einer optionalen Datenverschlüsselung auszustatten, wodurch dem Benutzer ermöglicht wird, den Betriebsmodus je nach Anwendungsfall festzulegen. Außerdem ist gerade bei Werkzeugen zur Datentunnelung, die mit einer bestimmten Motivation installiert und von einem eingeschränkten Personenkreis genutzt werden, symmetrische Kryptographie in Form einer Blockchiffre wie AES[4] potentiell ausreichend. Die verwendeten Schlüssel könnten bei der Einrichtung der Software bereits hinterlegt werden.

Der Vorteil bestünde in vollständig verschlüsselten Nutzdaten ohne zusätzlichem Ressourcenbedarf, abgesehen von möglichen Füllbytes zur Erreichung der Blockgröße der Chiffre. Es müssten zum Beispiel im Fall von AES, das eine Blockgröße von 128 Bits hat, Daten in Paketen zu je 16 Bytes zusammengefasst werden. Je nach Einsatzzweck des Tunnels kann eine Stromchiffre wie RC4 [45] zu bevorzugen sein, da Verfahren dieser Art nicht blockorientiert arbeiten und der zusätzliche Aufwand der Datensegmentierung damit entfällt.

## 5.3 Probleme

Beim Betrieb eines Datentunnels per ICMP Echo Nachrichten ähnelt das hochfrequente Eintreffen von Datenpaketen der Charakteristik einer „Denial of Service“ Attacke (als „ping flood“ bekannt), bei der durch das Bombardieren eines Computers mit Datenpaketen eine Überlastung provoziert werden soll [12]. Firewalls und IDS können folglich die Kommunikationsvorgänge der Tunnelungssoftware als Angriff missinterpretieren und entsprechend reagieren. Lösen lässt sich dieses Problem durch die hinreichende Verminderung der Sendefrequenz.

Bezogen auf die PingGate Implementierung ist der Umstand problematisch, dass die Programmausführung nur mit erweiterten Benutzerrechten möglich ist. Diese kann unter Windows nur ein Administratorenkonto bieten und unter Linux typischerweise „root“. Der Grund hierfür ist, dass der Vorgang des Abhörens und Sendens von ICMP Nachrichten einem regulären Benutzerkonto aus Sicherheitsgründen vom Betriebssystem nicht gestattet wird. Sowohl das Abhören einer Netzwerkschnittstelle (und Herausfiltern der relevanten Echo Nachrichten), als auch das Erstellen eines einfachen ICMP Sockets erfordert erweiterte Rechte.

[3] http://www.meadowy.org/~gotoh/projects/connect
[4] **A**dvanced **E**ncryption **S**tandard, http://www.nist.gov/aes

# Kapitel 6

# Fazit

Das Format des ICMP Echo Nachrichtentyps ist mit dessen beliebig befüllbarem Datenfeld geradezu prädestiniert zur Einbettung von Zusatzinformationen. In der Praxis gibt es dafür eine handvoll legitime Einsatzszenarien, wobei die tatsächliche Brauch- und Einsetzbarkeit auf sehr spezielle Situationen beschränkt sein dürfte. Programme wie PingGate sind ausschließlich Werkzeuge für Sonderfälle, in denen technisch keine Alternativen zur Verfügung stehen, die eine performantere und zuverlässigere Kommunikation ermöglichen würden.

Je nach Konfiguration beträgt die Bandbreite eines PingGate Kommunikationskanals bei einem Sende- und Empfangstakt von 100 Millisekunden, sowie einem Nutzdatenfeld von 1000 Bytes ohne der Annahme von Paketverlust bidirektional 10 KB/s. Dies ist etwas schneller als die Übertragung zu Zeiten des 56k Modems möglich gewesen wäre und taugt zum Beispiel noch eingeschränkt zum Surfen, Chatten und Versenden von E-Mails.

Die Hürden für den Betrieb eines solchen Datentunnels sind angesichts der Anforderungen (ein im Internet erreichbarer Proxy Server, client- und proxyseitige erweiterte Benutzerrechte), sowie der einfachen Verhinderbarkeit solcher Tunnelungsanstrengungen anhand trivialer Firewallkonfigurationen, relativ hoch. Entsprechend ausgestattete Spezialisten wissen beim Bereitschaftsbetrieb einer PingGate Proxy Instanz jedoch vor allem zu schätzen, sich im Notfall vielleicht doch noch behelfen zu können und einen „tunnel of last resort“ zur Verfügung zu haben.

Da SSH neben der im Abschnitt 5.2 beschriebenen Port Weiterleitung übrigens auch selbst eine eingebaute SOCKS Schnittstelle bietet, wird der Aufwand für die Flexibilität des im Kapitel 3 vorgestellten Tunnelungsprotokolls in Frage gestellt. Dennoch gibt es angesichts des zusätzlichen Ressourcenbedarfs, sowie der nicht unbedingt gegebenen Verfügbarkeit von SSH in einem Windows Umfeld, hinreichend Gründe, um das thematisierte Konzept zumindest im akademischen Rahmen zu legitimieren.

# Literaturverzeichnis


[1] Ahsan, K.: *Covert channel analysis and data hiding in TCP/IP*. Diplomarbeit, Department of Electrical and Computer Engineering, University of Toronto, Aug. 2002.

[2] Ahsan, K. und D. Kundur: *Practical Data Hiding in TCP/IP*. ACM Workshop on Multimedia and Security, Juan-les-Pins, Frankreich, Dez. 2002. http://www.sigmm.org/archive/MMSec/mmsec02/ahsan_kundur.pdf.

[3] Anderson, R. und F. Petitcolas: *On The Limits of Steganography*. IEEE Journal of Selected Areas in Communications, 16:474–481, 1998.

[4] Bar-El, H.: *Introduction to side channel attacks*. Informationsschrift, Discretix Technologies Ltd., 2006. http://www.hbarel.com/publications/Introduction_To_Side_Channel_Attacks.pdf.

[5] Castro, S.: *How to cook a covert channel*. Hakin9, Jan. 2006.

[6] Conta, A., S. Deering und M. Gupta: *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. RFC 4443, http://www.ietf.org/rfc/rfc4443.txt, März 2006.

[7] Cotton, M. und L. Vegoda: *Special Use IPv4 Addresses*. RFC 5735, http://www.ietf.org/rfc/rfc5735.txt, Jan. 2010.

[8] Deering, S. und R. Hinden: *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460, http://www.ietf.org/rfc/rfc2460.txt, Dez. 1998.

[9] Dierks, T. und E. Rescorla: *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246, http://www.ietf.org/rfc/rfc5246.txt, Aug. 2008.

[10] DoD 5200.28-STD: *Trusted Computer System Evaluation Criteria*. DoD Computer Security Center, Dez. 1985. http://csrc.nist.gov/publications/history/dod85.pdf.

[11] Eastlake 3rd, D. und A. Panitz: *Reserved Top Level DNS Names*. RFC 2606, http://www.ietf.org/rfc/rfc2606.txt, Juni 1999.

[12] Eden, L. v.: *The truth about ICMP*. GSEC Lektüre, SANS Institut, Mai 2001. http://www.giac.org/paper/gsec/719/truth-about-icmp/101601.

[13] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616, http://www.ietf.org/rfc/rfc2616.txt, Juni 1999.

[14] Giffin, J., R. Greenstadt, P. Litwack und R. Tibbetts: *Covert messaging through TCP timestamps*. In: *Proceedings of the Privacy Enhancing Technologies Workshop*, S. 194–208, San Francisco, Kalifornien, USA, Apr. 2002.

[15] Gligor, V.: *A guide to understanding covert channel analysis of trusted systems*. Techn. Ber., National Computer Security Center, Nov. 1993. http://www.iwar.org.uk/comsec/resources/standards/rainbow/NCSC-TG-030.html.

[16] Handel, T. G. und M. T. Sandford, II: *Hiding data in the OSI network model*. In: *Proceedings of the First International Workshop on Information Hiding*, S. 23–38, London, UK, 1996. Springer-Verlag.

[17] Huskamp, J. C.: *Covert communication channels in timesharing systems*. Dissertation, California Univ., Berkeley, 1978.

[18] Internet Assigned Numbers Authority: *IPv4 address space*. http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml.

[19] Internet Systems Consortium: *Internet host count history*. http://www.isc.org/solutions/survey/history.

[20] Jacobson, V., R. Braden und D. Borman: *TCP Extensions for High Performance*. RFC 1323, http://www.ietf.org/rfc/rfc1323.txt, Mai 1992.

[21] Johnson, N. F. und S. Jajodia: *Exploring steganography: Seeing the unseen*. IEEE Computer, 31:26–34, 1998.

[22] Kemmerer, R. A.: *Shared resource matrix methodology: an approach to identifying storage and timing channels*. ACM Trans. Comput. Syst., 1:256–277, August 1983.

[23] Kent, S.: *IP Authentication Header*. RFC 4302, http://www.ietf.org/rfc/rfc4302.txt, Dez. 2005.

[24] Lampson, B. W.: *A note on the confinement problem*. Commun. ACM, 16:613–615, Okt. 1973.

[25] Leech, M.: *Username/Password Authentication for SOCKS V5*. RFC 1929, http://www.ietf.org/rfc/rfc1929.txt, März 1996.

[26] Leech, M., M. Ganis, Y. Lee, R. Kuris, D. Koblas und L. Jones: *SOCKS Protocol Version 5*. RFC 1928, http://www.ietf.org/rfc/rfc1928.txt, März 1996.

[27] LeMay, M.: *Covert channels*. Vorlesungsskript, University of Illinois, Apr. 2006. http://www.cs.uiuc.edu/class/sp06/cs523/lectures/25/lemay.ppt.

[28] Lucena, N. B., G. Lewandowski und S. J. Chapin: *Covert Channels in IPv6*. In: *Proceedings of the Privacy Enhancing Technologies Workshop*, S. 147–166, Dubrovnik, Kroatien, Mai 2005.

[29] Mockapetris, P.: *Domain names - concepts and facilities*. RFC 1034, http://www.ietf.org/rfc/rfc1034.txt, Nov. 1987.

[30] Mockapetris, P.: *Domain names - implementation and specification*. RFC 1035, http://www.ietf.org/rfc/rfc1035.txt, Nov. 1987.

[31] Moskowitz, I. S., L. Chang und R. E. Newman: *Capacity is the wrong paradigm*. In: *New Security Paradigms Workshop*, S. 114–126, Virginia Beach, Virginia, USA, 2002.

[32] Moskowitz, I. S. und M. H. Kang: *Covert channels - here to stay?* In: *COMPASS '94*, S. 235–243, Gaithersburg, Maryland, USA, 1994. IEEE Press.

[33] Murdoch, S. und S. Lewis: *Embedding covert channels into TCP/IP*. In: *Information Hiding: 7th International Workshop*, Bd. 3727, S. 247–261. Springer-Verlag, Juni 2005.

[34] Nussbaum, L. und O. Richard: *On robust covert channels inside DNS*. In: *24th IFIP International Security Conference*, Pafos, Cyprus, Apr. 2009.

[35] Põder, S.: *Case of a wireless hack*, 2005. http://www.scribd.com/doc/56216295/Case-of-a-Wireless-Hack.

[36] Postel, J.: *User Datagram Protocol*. RFC 768, http://www.ietf.org/rfc/rfc768.txt, Aug. 1980.

[37] Postel, J.: *Internet Control Message Protocol*. RFC 792, http://www.ietf.org/rfc/rfc792.txt, Sep. 1981.

[38] Postel, J.: *Internet Protocol*. RFC 791, http://www.ietf.org/rfc/rfc791.txt, Sep. 1981.

[39] Postel, J.: *Transmission Control Protocol*. RFC 793, http://www.ietf.org/rfc/rfc793.txt, Sep. 1981.

[40] Postel, J.: *TCP maximum segment size and related topics.* RFC 879, http://www.ietf.org/rfc/rfc879.txt, Nov. 1983.

[41] Postel, J. und J. Reynolds: *File Transfer Protocol.* RFC 959, http://www.ietf.org/rfc/rfc959.txt, Okt. 1985.

[42] Ramakrishnan, K., S. Floyd und D. Black: *The Addition of Explicit Congestion Notification (ECN) to IP.* RFC 3168, http://www.ietf.org/rfc/rfc3168.txt, Sep. 2001.

[43] Réseaux IP Européens: *IPv4 exhaustion.* http://www.ripe.net/internet-coordination/ipv4-exhaustion/faq.

[44] Schaefer, M., B. Gold, R. Linde und J. Scheid: *Program confinement in KVM/370.* In: *Proceedings of the 1977 annual conference*, ACM '77, S. 404–410, New York, USA, 1977. ACM.

[45] Schneier, B.: *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C.* John Wiley & Sons, Inc., New York, USA, 1995.

[46] Shafranovich, Y.: *Common Format and MIME Type for Comma-Separated Values (CSV) Files.* RFC 4180, http://www.ietf.org/rfc/rfc4180.txt, Okt. 2005.

[47] Shannon, C. E.: *A mathematical theory of communication.* The Bell system technical journal, 27:379–423, Juli 1948.

[48] Simmons, G. J.: *The prisoners' problem and the subliminal channel.* In: *CRYPTO'83*, S. 51–67, 1983.

[49] Sohn, T., J. Moon, S. Lee, D. Lee und J. Lim: *Covert channel detection in the ICMP payload using support vector machine.* In: *ISCIS'03*, S. 828–835, Antalya, Türkei, Nov. 2003.

[50] Stødle, D.: *The ancient art of tunneling, rediscovered.* 2600: The Hacker Quarterly, 22(3), 2005.

[51] Stuart, T.: *ICMP: Crafting and other uses.* GSEC Lektüre, SANS Institut, Aug. 2001. http://www.giac.org/paper/gsec/1354/icmp-crafting-issues/102553.

[52] The Federal Networking Council: *Definition of "Internet".* http://www.nitrd.gov/fnc/Internet_res.html.

[53] The Internet Society: *A Brief History of the Internet.* http://www.isoc.org/internet/history/brief.shtml.

[54] Thomson, S., C. Huitema, V. Ksinant und M. Souissi: *DNS Extensions to Support IP Version 6.* RFC 3596, http://www.ietf.org/rfc/rfc3596.txt, Okt. 2003.

[55] Trithemius, J.: *Steganographia: Hoc est: Ars per occultam scripturam animi sui voluntatem absentibus aperiendi certa.* Matthiae Beckeri, Frankfurt, 1606.

[56] Wamser, J.: *Standort Indien.* LIT Verlag, Münster, 2005.

[57] Watzlawick, P., J. Beavin und D. Jackson: *Pragmatic of Human Communication.* W.W. Norton & Co., New York, USA, 1967.

[58] Wendzel, S.: *Einführung in verdeckte Kanäle.* Talk am 9. Augsburger Linux-Infotag, http://www.wendzel.de/dr.org/files/Papers/covert.channels.einfuehrung.pdf, März 2010.

[59] Zimmermann, H.: *OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection.* IEEE Transactions on Communications, 28(4):425–432, 1980.