DIPLOMARBEIT

STEGANOGRAPHISCHE PROTOKOLLE FÜR DIE TELEKOMMUNIKATION

ausgeführt zum Zweck der Erlangung des akademischen Grades eines
„Diplom-Ingenieurs für technisch-wissenschaftliche Berufe“
am Masterstudiengang Telekommunikation und Medien
der Fachhochschule St. Pölten

unter der Erstbetreuung von
Univ.Doz. Dipl.-Ing. Dr.tech. Ernst Piller

Zweitbegutachtung von
FH Prof. Dipl.-Ing. Johann Haag

ausgeführt von
Marcus Nutzinger, BSc
tm071044

St. Pölten, 2. Februar 2009

„Suche nichts zu verbergen, denn die Zeit, die alles sieht und hört, deckt es doch auf.“

– Sophokles (496 – 405 v. Chr.)

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter oder einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der von den Begutachtern beurteilten Arbeit überein.

St. Pölten, 2. Februar 2009
Ort, Datum

Unterschrift

Kurzfassung

Durch den geeigneten Einsatz von Steganographie kann die Kryptographie in gewissen Bereichen verbessert werden, z.B. bei der Verteilung und Speicherung kryptographischer Schlüssel oder bei der Abwehr der in der Kryptographie oftmals schwierig zu beherrschenden *Man in the Middle* Angriffen. Ein Datenaustausch, vor allem Schlüsselaustausch, der von einem Man in the Middle nicht als solcher erkannt wird, kann von ihm auch nicht für einen Angriff ausgenutzt werden. Diese Arbeit stellt daher neue Verfahren zur Verbesserung der Kryptographie mittels Steganographie vor. Besonders wird hier auf Protokolle eingegangen, welche die herkömmliche steganographische Kommunikation unterstützen und einen einheitlichen – sicheren – Ablauf gewährleisten sollen.

Als Einstieg in die Protokollbeschreibungen wird ein Nachrichtenformat definiert, welches für die Kommunikation herangezogen werden kann. Danach werden vier Protokollvarianten (General Purpose Steganographic Protocol (GSP)), mit unterschiedlichen Lösungen zur Abwicklung der Authentifizierung und der Abwehr eines Man in the Middle Angriffs, vorgestellt.

Abstract

The use of steganographic techniques can help cryptographic systems in some areas, for example when distribution or storing cryptographic keys or when defending against *Man in the Middle* attacks, which are often hardly covered in the context of pure cryptography. An exchange of information, especially a key exchange, which can't be recognised by a Man in the Middle, can't be used for an attack either. This work presents new ways to improve cryptography through steganography. Especially protocol designs are emphasized, which can be used to assist traditional steganographic communications to ensure a coherent – secure – procedure.

To enter the protocol description, a message format is defined first, which can be used for further communications. Afterwards, four variants of a protocol (General Purpose Steganographic Protocol (GSP)) are introduced, which all bring (kind of) different solutions for the the authentication process handling and the defense against Man in the Middle attacks.

Danksagung

Bedanken möchte ich mich in erster Linie bei meinem Betreuer, Univ.-Doz. Dipl.-Ing. Dr. Ernst Piller, für seine Unterstützung im Laufe dieser Arbeit und die zahlreichen produktiven Gespräche mit gegenseitigem Ansporn durch neue Ideen. Hätte das Klima nicht so gut gepasst, wäre wohl nicht dieselbe Arbeit entstanden.

Weiters möchte ich mich ganz herzlich bei meinen Eltern bedanken, welche mich während meiner gesamten Studienzeit unterstützten und mir vor allem auch kurz vor Abschluss dieser Arbeit geholfen haben, trotz Krankheit die Motivation nicht zu verlieren.

Zu guter Letzt auch noch ein Dankeschön an all meine Freunde und Bekannten, welche sich während der letzten Monate über den Status dieser Arbeit erkundigten und sich im Anschluss längeren Gesprächen unterziehen mussten. Dadurch hatte ich die Gelegenheit, die Inhalte in meinem Kopf Revue passieren zu lassen.

Marcus Nutzinger
St. Pölten, Februar 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation für das Thema

Der Austausch von Informationen ist in einer Informationsgesellschaft eine wesentliche Aufgabe. Hierbei besteht sehr oft auch das Interesse daran, die gesendeten Daten geheim zu halten. Während kryptographische Verfahren hierfür inzwischen leistungsfähige Lösungsansätze bieten, gewinnt auch das Verbergen der Tatsache, dass überhaupt eine Kommunikation stattfindet, zunehmend an Bedeutung. Dies ist vor allem in Bereichen und Ländern entscheidend, wo Kryptographie verboten wurde, deren Einsatz unerwünscht ist oder allein schon die Kommunikation an sich verdächtig wirkt und deshalb aus strategischen Gründen verborgen bleiben soll. In diesen Fällen muss auf Algorithmen der Steganographie zurückgegriffen werden.

Die Steganographie ist schon eine sehr alte Wissenschaft und auch die IT-gestützte Steganographie kann auf eine relativ lange Vergangenheit verweisen. Dennoch ist die Forschungsarbeit fast ausschließlich auf die Entwicklung und Verbesserung der Basistechnologien (wie Einbettungsverfahren) konzentriert (eine Ausnahme bildet hier eventuell das Digital Rights Management (DRM) — diese Arbeit beschäftigt sich jedoch nicht mit diesem Spezialbereich der Steganographie). Für kryptographische Anwendungen sind zusätzliche Protokolle entwickelt worden, welche z.B. als Request for Comments (RFC) im Internet veröffentlicht werden. Diese Entwicklung wäre auch für die Steganographie relevant, jedoch ist in diesem Bereich bis jetzt relativ wenig geschehen, verwendete Protokolle sind Produktspezifisch und nicht öffentlich zugänglich.

Aus den genannten Gründen beschäftigt sich diese Arbeit mit der Entwicklung von Protokollen für steganographische Kommunikation. Ein Fokus wird hierbei auf die Telekommunikation (Sprachübertragung, e-Mail mit Bildern im Anhang, etc.) gelegt.

1.2 Forschungsleitende Fragestellungen

In dieser Arbeit werden die folgenden zwei Forschungsfragen behandelt:

- Welche Probleme bestehen bei einer „herkömmlichen“ steganographischen Kommunikation?
- Wie kann der Man in the Middle Angriff durch Einsatz von definierten Protokollabläufen abgewehrt werden?

1.3 Aufbau dieser Arbeit

Diese Arbeit ist in drei Bereiche geteilt. Der erste Teil (Kapitel 2) beschäftigt sich mit den Grundlagen der Steganographie, zeigt Verfahrensweisen auf und auch möglichen Probleme bei steganographischer Kommunikation. Hier wird auch die erste Forschungsfrage beantwortet. Des weiteren wird auf den Einsatz von Protokollen eingegangen um die Relevanz von Kommunikationsprotokollen darzustellen.

In Kapitel 3 werden die Steganographie und Kommunikationsprotokolle kombiniert. Hierbei werden wichtige Rahmenbedingungen für die zu entwickelnden Protokolle geklärt und das Format der einzubettenden Nachrichten definiert.

Den wichtigen Protokollphasen zur Initiierung der Kommunikation sowie zur Verteidigung gegen Man in the Middle Attacken widmet sich Kapitel 4. Hier werden auch die verschiedenen Protokollvarianten definiert (zweite Forschungsfrage) und deren wichtigste Eigenschaften verglichen. Eine Übersicht der Man in the Middle Angriffstechnik liefert Anhang B.

Kapitel 2

Steganographie und Kommunikationsprotokolle

2.1 Geschichte der Steganographie

Steganographie ist die Wissenschaft der verdeckten Kommunikation. Während in der Kryptographie der Schutz der Daten im Vordergrund steht, wird hierbei das Vorhanden sein von Informationen verschleiert. Das Wort kommt aus dem Altgriechischen $\sigma\tau\varepsilon\gamma\alpha\nu\acute{o}$ $\gamma\rho\alpha\phi$-$\varepsilon\iota\nu$ und heißt ursprünglich so viel wie „versteckt schreiben“, was das Verstecken von Informationen in anderen Informationen bedeutete (vgl. Petitcolas u. a., 1999).

Über die Anfänge der Steganographie berichtete bereits ein Grieche Namens Herodot (400 – 425 v. Chr.). Es wird ein Sklave beschrieben, welchem die Haare abrasiert und auf seine Kopfhaut steganographische Informationen tätowiert wurden. Nachdem die Haare nachgewachsen waren, sendete man ihn an sein Ziel. Die Information wurde somit „versteckt“ übertragen. Weitere geschichtliche Beispiele nennen den Einsatz von Wachstafeln, welche die Botschaften unter der Wachsschicht beherbergten. Anfang des 20. Jahrhunderts und im Zweiten Weltkrieg wurden auch Geheimtinten (chemische Mischungen, welche nur unter bestimmten Umständen, wie Wärmeeinwirkung, sichtbar werden) und *Microdots* zur Kommunikation eingesetzt. Microdots haben die Größe eines i Punktes und können somit auf unscheinbaren Dokumenten wichtige Daten beherbergen (vgl. Hübner, 2003, Petitcolas u. a., 1999, Piller, 2007).

Die moderne Steganographie wird vor allem in Situationen eingesetzt, wo eine intensive Kommunikation (auch, oder vor allem, wenn diese verschlüsselt ist) Verdacht erwecken könnte. Natürlich ist ihr Einsatz deshalb nicht ganz ungefährlich, da diese Methoden auch für kriminelle Aktivitäten herangezogen werden können. Moderne Medien für steganographische Kommunikation sind Bilddateien, Audio- und Videodaten und auch Telefongespräche über Voice over IP (VoIP), Global System for Mobile Communications (GSM) oder Universal Mobile Telecommunications System (UMTS) (vgl. Piller, 2007).

Der Rest dieses Grundlagenkapitels beschränkt sich ausschließlich auf die Verfahrensweisen der modernen Steganographie.

2.2 Abgrenzung zur Kryptographie

In der Literatur wird Steganographie einerseits als Teilbereich der Kryptographie, andererseits als eigenständige Wissenschaft dargestellt. Unabhängig von dieser Abgrenzung verfolgen beide Wissenschaften grundsätzlich verschiedene Ziele. Bei der Kryptographie ist es für einen Angreifer *klar ersichtlich*, dass Informationen ausgetauscht werden. Die Sicherheit wird dadurch erlangt, dass die Information verschlüsselt übertragen und daher nur mit Kenntnis eines geheimen Schlüsseln gelesen werden kann. Bei der Steganographie liegt die Sicherheit darin, dass ein Angreifer *nicht bemerkt*, dass *sensitive* Daten innerhalb einer Kommunikation übertragen werden (vgl. Wikipedia, 2008f). Auch wenn die steganographischen Nachrichten oftmals vor dem Übertragen verschlüsselt werden, liegt die Sicherheit dieser Wissenschaft darin, dass ein Angreifer *keine Kenntnis* über die Tatsache hat, dass geheime Daten übertragen werden.

2.3 Information Hiding

Die Steganographie ist ein Teilbereich eines umfangreicheren Fachgebietes, welches *Information Hiding* genannt wird. Eine Unterteilung in verschiedene Disziplinen ist in Abbildung 2.1 dargestellt.

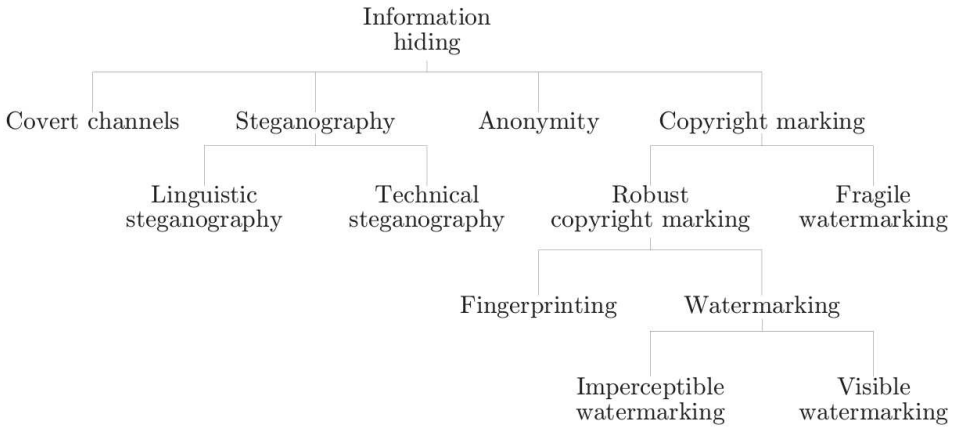

Abbildung 2.1: Klassifikation von Techniken des *Information Hidings* (Petitcolas u. a., 1999)

Wie Abbildung 2.1 zeigt, existieren neben der Steganographie noch andere Teilbereiche, von welchen vor allem das *Copyright Marking* immer größer und bedeutender wird. Hierunter fallen auch die Methoden für DRM und *Digital Watermarking*. Diese

Arbeit konzentriert sich jedoch auf den Bereich der Steganographie, welcher wiederum in mehrere Teilbereiche unterschieden werden kann (vgl. Hübner, 2003, Petitcolas u. a., 1999).

2.4 Wichtige Begriffe

Wie in allen technischen Disziplinen gibt es auch für die Steganographie einige spezielle Begriffe. Die *eingebetteten Informationen* sind jene Daten, welche versteckt übertragen werden sollen. Diese werden auch als *Payload* bezeichnet und in andere (unscheinbare) Daten, *Cover* bzw. *Trägermedium*, eingebettet. Ein *Stego-Key* wird eingesetzt, um den steganographischen Algorithmus zur Einbettung und Extraktion zu kontrollieren (vgl. Petitcolas u. a., 1999, Pfitzmann, 1996).

Für andere Bereiche des Information Hidings gibt es natürlich weitere Fachbegriffe. Diese werden hier jedoch nicht aufgeführt da für dieses Kapitel nur der Teilbereich Steganographie betrachtet wird.

2.5 Aufbau eines Stegosystems

Der Kommunikationskanal samt Sender und Empfänger wird als *Stegosystem* bezeichnet. Abbildung 2.2 zeigt den generellen Aufbau.

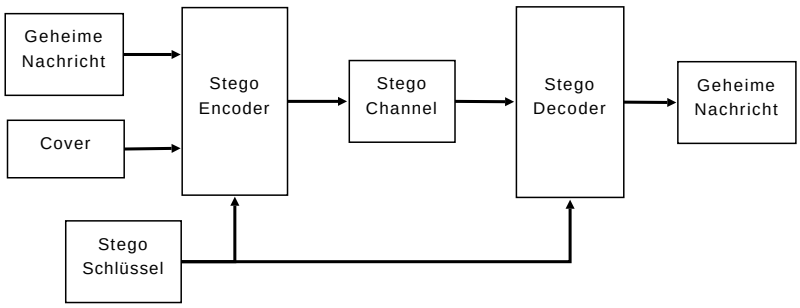

Abbildung 2.2: Aufbau eines Stegosystems (vgl. Mittelholzer, 1999)

Wie in der Abbildung zu sehen ist, werden die zu übertragenden Informationen mittels eines *Stego Encoders* in das Cover (Trägermedium) eingebettet. Der *Stego-Key* dient dazu, die Information nur berechtigten Entitäten zugänglich zu machen. Der *Stego Decoder* extrahiert mit Hilfe des Stego-Keys wiederum die Daten aus dem Cover (vgl. Mittelholzer, 1999). Da das gezeigte Stegosystem alle Daten im Klartext in das Cover einbettet ist es naheliegend, zusätzlich kryptographische Methoden für mehr Sicherheit einzusetzen. Hierbei werden die geheimen Daten vor der Übergabe an den Stego Encoder mittels kryptographischen Algorithmen (symmetrisch oder asymmetrisch) verschlüsselt.

C. Cachin liefert eine mathematische Definition eines Stegosystems, wobei dieselben Elemente wie in Abbildung 2.2 zum Einsatz kommen.

- Ein *Key Generation Algorithm* liefert einen Bitstrom zur Verwendung als Stego-Key.
- Ein *Steganographic Encoding Algorithm* übernimmt den Stego-Key und eine Nachricht und liefert als Ausgabe einen Stegotext, welcher zur Übertragung verwendet werden kann.
- Ein *Steganographic Decoding Algorithm* übernimmt den Stego-Key und einen Stegotext und liefert als Ausgabe entweder die zugehörige Nachricht oder einen Fehlercode, wenn keine Nachricht im Stegotext eingebettet war (vgl. Cachin, 2005).

Es ist zu beachten, dass in diesem Modell keinerlei Sicherheit gegenüber Veränderungen auf dem Übertragungsweg (*Stego Channel*) verfügbar ist. Dies ist in allgemeinen steganographischen Anwendungen immer der Fall, weshalb diese Arbeit auf die Verbesserung dieser Tatsache durch den Einsatz von Protokollen fokussiert ist (siehe Kapitel 3 und 4).

2.6 Paradigmen für steganographische Methoden

Features, welche von Techniken zum Information Hiding unterstützt werden sollten, umfassen:

- Die eingebetteten Daten sollen nicht (oder nur minimal) erkennbar sein (für Unbeteiligte).
- Bei Verdacht auf eine geheime Nachricht darf diese trotzdem nicht einfach aus dem Cover extrahiert werden können.
- Die Daten sollen direkt in die Rohdaten und nicht in Headerdaten eingebettet werden, da diese sich am Übertragungsweg ändern können (durch Änderungen von Protokollen, Codecs, etc.).
- Das Cover soll keine Anhaltspunkte über die Existenz von geheimen Daten geben.
- Die zu versteckende Nachricht muss kleiner sein als das Cover.
- Der Einsatz von Codes zur Fehlererkennung (Cyclic Redundancy Check (CRC)) soll angestrebt werden, um kleinere Datenfehler durch die Übertragung beheben zu können (vgl. Bender u. a., 1996, Piller, 2007).

Diese Punkte treffen sowohl für Steganographie als auch für allgemeine Techniken des Information Hiding zu. Des weiteren gibt es einige wichtige Kenngrößen, welche vor allem steganographische Anwendungen beeinflussen.

- Menge der zu versteckenden Daten,
- Robustheit des Covers, d.h. wie resistent das Cover und die eingebetteten Daten gegen Veränderungen (wie eine Analog/Digital Transformation) sind,
- Unsichtbarkeit der geheimen Daten — diese Eigenschaft steht mit der Menge der einzubettenden Daten in Beziehung,
- Für Echtzeitanwendungen ist die Rechenleistung, welche für das Einbetten und Auslesen benötigt wird, relevant,
- Der Aufwand für den Empfänger, die eingebetteten Daten zu extrahieren (vgl. Piller, 2007).

Bei steganographischen Systemen ist ein Verständnis des Kommunikationskanals sehr viel wichtiger als bei der Kryptographie. Besitzt ein Kanal z.B. *Natural Noise* (natürliches Rauschen), dann kann diese Eigenschaft zur Einbettung von Informationen in diesem bestimmten Kanal ausgenutzt werden. Andere Kanäle können jedoch wieder gänzlich andere Eigenschaften aufweisen, wodurch auch die Herangehensweise für die Einbettung geändert werden muss (vgl. Petitcolas u. a., 1999).

2.6.1 Verschlüsselung der geheimen Daten

Die zuvor schon erwähnte Verschlüsselung der Daten vor dem Einbetten in das Trägermedium dient nicht nur der zusätzlichen Sicherheit sondern auch dem Einbetten selbst. Durch die Verschlüsselung werden die Daten *rauschähnlich*, was die statistischen Eigenschaften verbessert. Werden die Daten in Medien eingebettet, welche natürliches Rauschen beinhalten (wie Audiodaten), können die geheimen (und verschlüsselten) Daten nicht mehr vom Hintergrundrauschen des Mediums unterschieden werden (vgl. Federrath und Wicke, 1997, Hübner, 2003).

2.6.2 Veröffentlichung von Algorithmen – Kerckhoffs' Prinzip

Wie auch in der Kryptographie darf man sich bei steganographischen Anwendungen nicht auf *Security through Obscurity* verlassen. Daraus ergibt sich, dass Algorithmen, welche zur Einbettung und zum Extrahieren der geheimen Daten herangezogen werden, öffentlich bekannt sein sollen. Dieses Paradigma ist für kryptographische Anwendungen schon seit dem 19. Jahrhundert als *Kerckhoffs' Prinzip* bekannt:

> „[...] the security of the encryption scheme must depend only on the secrecy of the key K_e, and not on the secrecy of the algorithms." (Ferguson und Schneier, 2003, S. 23)

Dies macht also auch für steganographische Anwendungen das Vorhanden sein eines Schlüssels (*Stego-Key*) notwendig, mit welchem der Algorithmus „angesteuert“ wird. Wie bei kryptographischen Algorithmen können nur jene Teilnehmer die geheimen Daten aus dem Covermedium extrahieren, welche den zugehörigen Stego-Key besitzen (und auch über das Verfahren Bescheid wissen) (vgl. Federrath und Wicke, 1997).

2.7 Beispiele für steganographische Verfahren

Für die Durchführung einer steganographischen Kommunikation gibt es verschiedene Algorithmen. Diese sind einerseits vom verwendeten Trägermedium abhängig, andererseits gibt es auch für dieselben Trägermedien unterschiedliche Einbettungsalgorithmen.

2.7.1 Least Significant Bit (LSB) Einbettung

2.7.1.1 24-Bit RGB Bilder

Bei 24-Bit Bildern ist jeder Bildpunkt mit seinen Anteilen an den Grundfarben Rot, Grün und Blau kodiert. Modifiziert man hierbei das Least Significant Bit (LSB), können $(3 * \text{Höhe} * \text{Breite})$ Bits eingebettet werden. Der Nachteil ist, dass diese 24-Bit Bilder unkomprimiert und deshalb sehr groß sind (d.h. auffällig) (vgl. Hübner, 2003).

2.7.1.2 8-Bit GIF Bilder

Beim Graphics Interchange Format (GIF) Format werden in einer Farbpalette 256 Farben gespeichert. Die Pixel verweisen somit auf einen entsprechenden Eintrag der Tabelle. Allerdings ist auch der Einsatz der LSB Methode in diesem Fall nicht mehr so einfach, da der steganographische Algorithmus auch dafür sorgen muss, dass die Farbpalette nach dem Einbetten entsprechend umsortiert wird, um die Änderungen am Bild nicht erkennbar zu machen. Es können $(\text{Höhe} * \text{Breite})$ Bits eingebettet werden (vgl. Hübner, 2003, Petitcolas u. a., 1999).

2.7.1.3 Komprimierte JPEG Bilder

Auch das Einbetten in Joint Photographic Experts Group (JPEG) Bildern ist möglich, jedoch wird hierbei der Einbettungsalgorithmus vor der Kodierung der Bilddaten, also bevor die fertige JPEG Datei erzeugt wird, auf die Daten angewandt (vgl. Hübner, 2003).

2.7.1.4 Audio- und Videodaten

Bei Daten wie Audio- und Videoströmen ist die Anwendung einer LSB Einbettung nicht immer sinnvoll, da die Veränderung dieser Bits mitunter hörbar/sichtbar werden. Dies beeinflusst somit die Qualität der Aufnahme einerseits, sowie auch die Undetektierbarkeit der Einbettung andererseits, negativ (vgl. Petitcolas u. a., 1999).

2.7.2 Einbettung im Transformationsraum

Die LSB Einbettung ist nicht sehr komplex zu implementieren aber bei einigen Anwendungsfällen auch mitunter nicht sinnvoll oder sicher genug. Deshalb wurden weitere Verfahren entwickelt, welche auf Einbettungsräumen basieren. Diese Einbettungsräume lassen sich über mathematische Transformationen berechnen (vgl. Petitcolas u. a., 1999, Piller, 2007).

Die einzubettenden Daten könnten grundsätzlich in den Zeit- bzw. Ortsraum des Covermediums eingebettet werden. Diese Vorgehensweise birgt jedoch das Problem, dass sie anfällig gegenüber Störungen ist. Deshalb wird mit Hilfe von mathematischen Transformationen das Cover in den Frequenzbereich übergeführt, wo der Payload in einzelnen Frequenzen versteckt werden kann (vgl. Piller, 2007).

2.7.2.1 Diskrete Fourier Transformation (DFT)

Das Cover wird hierbei mittels der Fourier Analyse in einzelne Sinus Frequenzen zerlegt, welche für die Einbettung herangezogen werden können (vgl. Piller, 2007).

2.7.2.2 Diskrete Kosinus Transformation (DCT)

Die Diskrete Kosinus Transformation (DCT) ist ähnlich zur Diskrete Fourier Transformation (DFT). Das Cover wird als eine Summe von Frequenzen dargestellt, wobei mit diesem Algorithmus ausschließlich Cosinus Frequenzen verwendet werden. Diese können wiederum für die Einbettung verwendet werden (vgl. Piller, 2007).

2.7.2.3 Wavelet Transformation

Ein Wavelet kann wie eine Sinus und Cosinus Funktion als eine Basisfunktion angesehen werden, aus welcher wiederum ein Signal (das Cover) als Summe dargestellt werden kann. Wavelets besitzen bestimmte mathematische Eigenschaften, weshalb sie für diese Verfahren eingesetzt werden (vgl. Piller, 2007).

Ist das Cover nun in seine Frequenzanteile zerlegt, können die geheimen Daten eingebettet werden. Hierbei gibt es wiederum mehrere Möglichkeiten, wie das *Echo Hiding*.

Bei diesem Verfahren wird dem Signal ein (nicht hörbares) Echo hinzugefügt, wobei die Amplitude, der Abstand zur Originalfrequenz und die Abklingrate den Payload beschreiben können (vgl. Petitcolas u. a., 1999, Piller, 2007).

Für einen umfassenden Überblick über eine Vielzahl von Einbettungsverfahren und -arten siehe (Johnson und Katzenbeisser, 2000). Abschnitt 4.3.1 stellt außerdem kurz einige Ansätze vor, welche vorhandene Kommunikationsprotokolle für steganographische Übertragungen heranziehen.

2.8 Sicherheit

Wenn man von steganographischer Sicherheit spricht, muss man das Ziel der Steganographie betrachten: Ein Angreifer darf keine Kenntnis darüber haben, dass geheime Daten über einen Kommunikationskanal übertragen werden. Dadurch macht schon das Erkennen von eingebetteten Nachrichten (auch wenn der Angreifer den Inhalt nicht extrahieren kann) das Verfahren unbrauchbar. Nimmt man nun an, dass der Angreifer genügend Zeit und Leistung zur Verfügung hat, können verschiedene Angriffsarten auf steganographische Systeme realisiert werden. Nur wenn die Existenz von eingebetteten Nachrichten nicht von einer dritten Person nachgewiesen werden kann, ist das System sicher.

Im folgenden werden nun Angriffsarten vorgestellt und auf die jeweiligen Probleme eingegangen, welche überhaupt zu den Angriffen führen. Das Ziel dieser Arbeit ist es, diverse Probleme zu lösen und eine sichere steganographische Kommunikation verfügbar zu machen.

2.8.1 Motivation für einen Angriff

Es gibt mehrere Motive für einen Angriff, die gängigstens sind:

- Aufdecken, dass geheime Daten in einer Kommunikation versteckt wurden,
- Beweisen, dass geheime Daten vorhanden sind (wo diese genau versteckt sind, etc.),
- Versteckte Daten extrahieren, zerstören, etc. (vgl. Hübner, 2003).

Bei anderen Arten des Information Hiding (siehe Abschnitt 2.3) existieren eventuell auch zusätzliche oder abgeänderte Angriffsmotive. Auf diese wird hier jedoch nicht weiter eingegangen.

Unabhängig der Motivation des Angreifers muss davon ausgegangen werden, dass dieser *alles* machen kann. Es können Nachrichten abgefangen, gelesen und geändert werden, ohne dass die Kommunikationspartner etwas davon bemerken (Man in the

Middle Problem). Dieses Problem besteht, da ein Empfänger die Authentizität des Senders nicht nachweisen kann. Daher weiß der Empfänger nicht, ob die Daten am Übertragungsweg geändert wurden. Dieses Problem wird in den folgenden Kapiteln adressiert, indem Protokolle definiert werden, welche zur herkömmlichen steganographischen Kommunikation noch eine Authentifizierung (und somit Man in the Middle Abwehr) hinzufügen.

2.8.2 Arten von Angriffen

Wie schon beschrieben gibt es verschiedene Angriffsmotivationen und ein Angreifer hat grundsätzlich alle Möglichkeiten der Manipulation. Je nachdem, wie weit der Angreifer nun in die Kommunikation eingreifen möchte, ergeben sich verschiedene Angriffsarten. Abbildung 2.3 zeigt nochmals das Stegosystem aus Abbildung 2.2, mit den möglichen Angriffsarten aus (Franz und Pfitzmann, 1999).

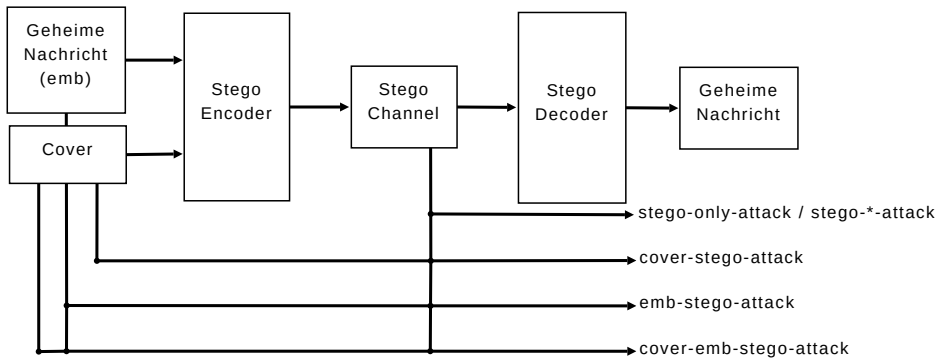

Abbildung 2.3: Mögliche Angriffe auf ein Stegosystem (vgl. Franz und Pfitzmann, 1999)

Die verschiedenen Angriffe aus Abbildung 2.3 sind auch unter anderen Namen bekannt. So gibt es auch die Bezeichnung *Chosen-Hiddentext* für den *emb-stego-attack*, wenn der Angreifer auch tatsächlich Inhalte ändert (*aktiver Angreifer*). Der *Chosen-Stegotext* ist mit dem *stego-only-attack* oder *stego-*-attack* vergleichbar, da ein aktiver Angreifer in diesen Fällen das Stego Objekt kontrolliert. Andere angeführte Angriffe stellen jeweils eine Kombination dieser Grundarten dar, wenn der Angreifer z.B. über mehr Wissen verfügt und somit nicht nur den Payload sondern auch das Cover entsprechend beeinflussen kann (vgl. von Ahn und Hopper, 2004, Franz und Pfitzmann, 1999).

Im folgenden werden der Chosen-Hiddentext und Chosen-Stegotext Angriff als verbreitete Vertreter der Angriffsarten beschrieben.

2.8.2.1 Chosen-Hiddentext

Der Chosen-Hiddentext Angriff geht davon aus, dass der Man in the Middle den Payload im Cover ändern kann. Durch beobachten der weiteren Kommunikation soll er nun herausfinden, ob die beiden Kommunikationspartner Steganographie verwenden (vgl. von Ahn und Hopper, 2004). Diese Form des Angriffs kann dadurch umgangen werden, dass die Verteilung der Bits im Payload ebenso „zufällig“ gemacht wird, wie jene im originalen Cover. Ist dies der Fall, kann ein Man in the Middle keine steganographische Kommunikation entdecken. Um eine Randomisierung zu erreichen, wird der Payload vor dem Einbetten verschlüsselt.

2.8.2.2 Chosen-Stegotext

Beim Chosen-Stegotext Angriff überwacht der Man in the Middle den Kommunikationskanal und kann auch eigene Nachrichten an die Kommunikationspartner versenden (mit oder ohne eingebettetem Payload) (vgl. von Ahn und Hopper, 2004). Dadurch, dass die Kommunikationspartner nicht wissen, ob die Nachricht vom Man in the Middle stammt oder nicht, werden sie versuchen einen Payload zu extrahieren. Kann der Man in the Middle Manipulationen an einer echten Nachricht vornehmen (und nicht nur eigene Nachrichten einschleusen) stellt dieser Angriff eine große Gefahr für die Integrität der Kommunikation dar. Dies ist nur durch Einsatz von Mechanismen zur Authentifizierung der Kommunikationspartner zu verhindern.

2.8.3 Probleme herkömmlicher steganographischer Kommunikation

Wie in den vorangegangenen Abschnitten betrachtet wurde, ist eine standardmäßige steganographische Kommunikation (ohne zusätzliche „Features“) unsicher. Diese Aussage bezieht sich nicht auf die Verfahren, welche technisch hochwertig sein können. Vielmehr ist das schwächste Glied hier der Kommunikationskanal, über den ein Dritter Angriffe auf die stattfindende Kommunikation durchführen kann. Ein Man in the Middle, welcher die Kommunikation abhört, kann somit auch in die Kommunikation einschreiten. Der einzige Schutz bleibt die Verschlüsselung, aber dennoch kann der Man in the Middle die Daten verändern.

Die Abhilfe gegen dieses Problem ist der Einsatz von Authentifizierung, für welche die herkömmliche Steganographie aber keine Verfahren bereitstellt. Deshalb ist es relevant, Protokolle (ähnlich zu kryptographischen Kommunikationsprotokollen) zu definieren, welche nicht nur die Kommunikation standardisieren sondern sich auch um zusätzliche Features (wie Man in the Middle Abwehr durch Authentifizierung) annehmen.

2.9 Vorteile von Kommunikationsprotokollen

Grundsätzlich dient ein Protokoll dazu, für eine Kommunikation einen geordneten Ablauf zu definieren. Der Vorteil liegt hierbei darin, dass durch den Einsatz eines geordneten Ablaufs einige Angriffsarten auf die Kommunikation abgewehrt (oder zumindest erschwert) werden können. Ein Beispiel hierfür ist der Einsatz von *Sequenznummern*. Diese bieten für den Empfänger die Möglichkeit, die empfangenen Datenpakete gemäß der Reihenfolge des Senders zu sortieren. Somit wird z.B. eine *Replay Attacke* erschwert, wenn der Dienst beim Empfänger vergangene oder doppelte Sequenznummern ignoriert. Des weiteren können durch Protokolle komplexe Abläufe, wie z.B. der Schlüsselaustausch bei Pretty Good Privacy (PGP), standardisiert werden.

Für Netzwerke stellt das Internation Organisation for Standardisation (ISO)/Open Systems Interconnection (OSI) Referenzmodell die Grundlage für Kommunikationsprotokolle dar (vgl. ISO/IEC, 1996). In Kapitel 4 wird auf dieses Modell nochmals im Kontext von bereits vorhandenen Lösungsvorschlägen und den in dieser Arbeit definierten Protokollen eingegangen.

Kapitel 3

Kombination der Technologien

Nachdem die Kunst der Steganographie sowie die Vorteile der standardisierten und anerkannten Protokolle für kryptographische Kommunikationen im letzten Kapitel beschrieben wurden, kann nun mit der Zusammenführung dieser beiden Themen begonnen werden. Die Vorteile davon sind einerseits die verdeckte Kommunikation, was einen Sicherheitsbonus gegenüber der herkömmlichen Kryptographie bedeutet und andererseits das Aufsetzen von Protkollen auf diese Mechanismen um einen einheitlichen und standardisierten Ablauf zu gewährleisten und Probleme der Steganographie (siehe Abschnitt 2.8.3) zu umgehen.

3.1 Definitionen

3.1.1 Parametersatz

Für die Definition von verwendeten Methoden und Algorithmen werden sogenannte *Parametersätze* zwischen den Kommunikationspartnern ausgetauscht. Beschrieben sind diese Parametersätze in den Basic Encoding Rules (BER) des Abstract Syntax Notation One (ASN.1) Standards nach ISO/International Electrotechnical Commission (IEC) 8825-1:2002 (vgl. ITU-T, 2002a,b). Die vollständige Ausprägung von Parametersätzen ist in den Abschnitten 3.4 und 3.6 beschrieben.

3.1.2 Basisparametersatz

Ein *Basisparametersatz* ist ein Parametersatz, welcher für die Protokolle definiert wurde um bei Kommunikationsbeginn verwendet werden zu können. Da diese öffentlich zur Verfügung stehen, sind hohe Anforderungen im Sinne der Sicherheit an die verwendeten Verfahren gestellt.

3.2 Voraussetzungen für das Protokolldesign

Im folgenden werden einige wichtige Voraussetzungen geklärt, welche für die Definitionen der Nachrichtenformate in diesem Kapitel relevant sind. Des weiteren sind die hier aufgelisteten Punkte auch für die unter Kapitel 4 beschriebenen Protokolle grundlegend.

3.2.1 Kennzeichnung der Verfahren

3.2.1.1 Steganographie

Die in diesem Dokument beschriebenen Protokolle sind vom verwendeten Einbettungalgorithmus unabhängig. Dieser beeinflusst lediglich den genauen Aufbau des Parametersatzes (siehe Abschnitt 3.4). Das bedeutet, dass alle Varianten wie LSB Einbettung, Echo Hiding, Pre/Post Masking, Frequency Masking, etc. (vgl. Petitcolas u. a., 1999) gleichermaßen zur Einbettung angewandt werden können.

3.2.1.2 Kryptographie

Wie auch bei den steganographischen Methoden ist die Auswahl von Algorithmen der Kryptographie nicht grundsätzlich eingeschränkt. In der Definition des schon erwähnten Parametersatzes ist beschrieben, wie welche kryptographischen Algorithmen für die Kommunikation aktiviert werden können und in welcher Form eventuell notwendige Schlüssel auszutauschen sind (siehe Abschnitt 3.4).

3.2.2 Einsatz von Verschlüsselung

Vor dem Einbetten wir jeder Payload mittels symmetrischer oder asymmetrischer Kryptographie verschlüsselt. Dies hat den Vorteil, dass die Informationen dadurch einerseits zusätzlich vor Angriffen (wie Mitlesen durch Man in the Middle) geschützt sind. Des weiteren dient der Einsatz der Verschlüsselung auch der Randomisierung der Daten. Wenn z.B. das ElGamal Kryptosystem eingesetzt wird, ergibt sich durch die Verschlüsselung immer eine echt zufällige Bitfolge, da in der ElGamal Verschlüsselungsfunktion eine Zufallszahl eingebracht wird (vgl. Menezes u. a., 1996, S. 294 f). Dies gibt den Daten auch bessere statistische Merkmale für das Einbetten des Payloads in das Cover (vgl. Wikipedia, 2008f).

3.2.2.1 Probleme beim Einsatz asymmetrischer Verschlüsselung

Wenn der Payload vor dem Einbetten mit dem öffentlichen Schlüssel des Kommunikationspartners verschlüsselt werden soll, ist es auch erforderlich, dass jede Entität

seinen Schlüssel publiziert (per e-Mail versenden, in einer Public Key Infrastructure (PKI) speichern, online stellen, etc.). Dies stellt in den meisten Fällen wohl kein Problem dar, da zumindest einer der Kommunikationspartner seine Schlüsselinformation zur Verfügung stellen kann, ohne Argwohn zu erregen.

Dennoch kann es vorkommen, dass keiner der Kommunikationspartner einen Schlüssel für den jeweils anderen zur Verfügung stellen kann. Ein Szenario wäre z.B. die Kommunikation von zwei Journalisten in einem fremden Land, welches ihrem Heimatland feindlich gesinnt ist und auch darüber Bescheid weiß, dass Kommunikation passiert. Die Regierung könnte den Austausch von öffentlichen Schlüsseln (über beliebige Kanäle) unterbinden, allein aus dem Grund da es einfach ist, einen Schlüssel als solchen zu erkennen. Deshalb stellen L. von Ahn und N.J. Hopper eine Methode vor, welche als *Steganographic Key Exchange (SKE)* bezeichnet wird. Dieses Protokoll tauscht innerhalb harmlos aussehender Nachrichten Informationen zwischen zwei Kommunikationspartnern aus, sodass beide am Ende einen *shared secret* berechnen können. Dieser Schlüssel kann dann für symmetrische Verschlüsselung des Payloads herangezogen werden (ähnlich einem Diffie-Hellman Protokoll für steganographische Anwendungen (vgl. von Ahn und Hopper, 2004)).

3.2.2.2 Paketorientierte Übertragung und Verschlüsselung

Wird der Payload in Paketen eingebettet um über ein Netzwerk versandt zu werden, dann ist es aufgrund der Eigenschaften der Kryptographie sinnvoll, jeweils einen verschlüsselten Block in einem Paket zu versenden. Dies ergibt eine erhöhte Fehlertoleranz der Applikation, da bei Verlust oder Übertragungsfehler von einem Paket auch nur ein verschlüsselter Block betroffen ist. Werden nun Verschlüsselungsmodi wie der *Counter Mode* verwendet, sind nachfolgende Blöcke von einem möglichen Verlust oder Übertragungsfehler nicht betroffen.

3.2.3 Einsatzgebiete der Protokolle

Der Fokus für das Design der Protokolle in Kapitel 4 liegt bei der steganographischen Übertragung von Daten in Audiostreams und Bilddaten. Das heißt es wird hauptsächlich auf Sprachübertragungen (Datenstrom in Echtzeit) und e-Mail Verkehr mit Bildern im Anhang, welche den Payload enthalten, eingegangen.

3.2.4 Format der übertragenen Informationen

Wie in den folgenden Abschnitten dargestellt wird, sind neben den eigentlichen Daten auch einige Zusatzinformationen bei der Übertragung notwendig, um funktionierende Protokolle implementieren zu können. Da alle Protokolle die selben Informationen verwenden, wird das Nachrichtenformat in diesem Kapitel definiert während das nächste Kapitel die eigentlichen Protokolle behandelt.

3.3 Analyse des Kommunikationskanals

Da die Protokollvarianten für eine steganographische Einbettung in beliebige Kommunikationskanäle herangezogen werden sollen, muss es auch eine ganzheitliche Betrachtung der Probleme geben, welche bei der Übertragung in den verschiedenen Kanälen auftreten können.

Im speziellen ist hier die Sprachübertragung (VoIP, GSM, UMTS, etc.) betroffen. Hier entsteht einerseits ein *Synchronisationsproblem*, was bedeutet, dass man nicht genau bestimmen kann, wann der Empfänger synchron mit dem Sender läuft. Daraus folgt, dass die Einbettung hier nicht ab dem ersten Bit erfolgen darf.

Das zweite Problem, *Kodierungsproblem* genannt, hängt mit der Fehlerrate von gesendeten Paketen zusammen. Während beim „normalen“ Paketversand (wie z.B. e-Mail Verkehr) das Transmission Control Protocol (TCP) das Nachsenden von verloren gegangenen Paketen regelt, gibt es bei VoIP Verkehr keine Maßnahmen diesbezüglich. Der Einsatz vom User Datagram Protocol (UDP) und die Echtzeit Eigenschaft von Sprache machen das eigentlich nicht notwendig. Für steganographische Kommunikation bedeutet das, dass eine entsprechende Kodierung eingesetzt werden muss, welche es ermöglicht, dass Übertragungsfehler (verlorene Pakete bzw. falsch übertragene Bitwerte) bis zu einem gewissen Grad erkannt und bereinigt werden können, unabhängig davon ob das Kommunikationsprotokoll (wie Simple Mail Transfer Protocol (SMTP), Hypertext Transfer Protocol (HTTP), etc.) Fehlerkorrigierende Maßnahmen zur Verfügung stellt.

3.3.1 Synchronisation

Problem 3.3.1.1 (Synchronisationsproblem) *Wie kann in einer Kommunikation garantiert werden, dass Sender und Empfänger synchron arbeiten? Kann die Synchronisation zwischen den Kommunikationspartnern verloren gehen?*

Bei Sprachübertragung ist eine Synchronisation sehr wichtig. Man darf nicht ab dem ersten Bit mit dem Einbetten beginnen, da der Sender nicht sicher sagen kann, ab wann der Empfänger fertig initialisiert wurde und auch bereit ist, Daten entgegenzunehmen bzw. auf steganographischen Payload zu achten. Daher kann man auch nicht von einem definierten Start und Ende der Kommunikation sprechen, der Payload wird in einen kontinuierlichen Datenstrom eingebettet (*Streaming-Daten*). Bei einem VoIP Gespräch wird der Datenstrom zwar dennoch in einzelnen Paketen zum Empfänger geschickt (typischerweise mit 20 ms Sprache pro Paket), aber die oben genannte Einbettung der Daten erfolgt schon eine logische Schicht darüber. Der Empfänger erhält nach dem Empfang und Zusammensetzen der Daten also auch wiederum einen Datenstrom, in welchem der Payload „versteckt“ ist. Durch diese Eigenschaft des Covers kann der Payload nicht ohne zusätzliche Informationen zum Auffinden eingebettet werden. (Vogel u. a., 2006) schlägt hier das Definieren von bestimmten Bytewerten vor, welche als Start- und Endmuster vor und nach einem Payload eingebettet werden.

Sofern der Empfänger ebenfalls die beiden Muster kennt, gewährleitet dieser Mechanismus die Lokalisierbarkeit der steganographischen Informationen im Datenstrom.

Neben diesen Start- und Endmustern benötigt man aber ebenfalls ein Bitmuster, welches den Start der Einbettung kennzeichnet. Der Unterschied zu den zuvor beschriebenen Mustern ist es, dass hierbei der Wert nicht eingebettet wird, sondern auf einen Wert im Datenstrom zurückgegriffen wird. Der Empfänger muss also, nachdem er das Muster mitbekommen hat, nach eingebetteten Informationen suchen (gemäß dem öffentlichen Parametersatz des Senders bzw. dem ausgehandelten privaten Parametersatz). Wenn der Sender bemerkt, dass das Startmuster unter den ersten wenigen Bytes (z.B. unter den ersten 20 Byte) vorkommt, ist es hier sicherer, den Payload zweimal einzubetten. Dies umgeht Probleme bei der Synchronisation mit dem Empfänger bzw. bei Verzögerungen im Initialisieren des Stego-Moduls. Als generelle Regel sollte der Payload also zumindest beim zweiten Aufteten des Synchronisationsmusters eingebettet werden. Abbildung 3.1 zeigt diesen Ablauf und auch die Unterschiede zum zuvor besprochenen Start- und Endmuster für den Payload.

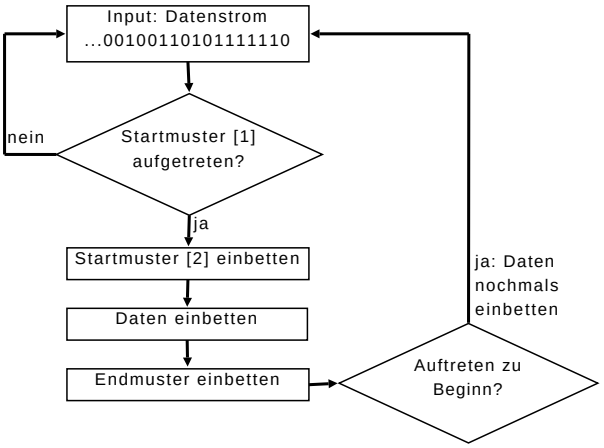

Abbildung 3.1: Regelwerk zur Einbettung mit Startmuster

3.3.1.1 Beantwortung des Synchronisationsproblems

Zu Beginn der Diskussion über Synchronisation wurden zwei Fragen aufgetan, welche nun beantwortet werden.

Wie kann in einer Kommunikation garantiert werden, dass Sender und Empfänger synchron arbeiten?

Für diesen Zweck sind eigene Start- und Endmuster vor bzw. nach dem Payload einzubetten (vgl. Vogel u. a., 2006). Dies garantiert dem Empfänger, dass er an der richtigen Stelle im Datenstrom beginnt, nach einem Payload zu suchen. Des weiteren muss auch das Vorhanden sein eines Payloads an sich gekennzeichnet werden. Hier ist es möglich,

ein Bitmuster zu vereinbaren, welches im Cover zufällig an einer Stelle vorkommt. So schafft man die Synchronisation von Sender und Empfänger in Bezug auf das Vorhandensein von steganographischen Informationen.

Kann die Synchronisation zwischen den Kommunikationspartnern verloren gehen?

Bei Paketverlust oder bei Paketen, welche „kaputt“ (mit geändertem Inhalt) ankommen, kann es passieren, dass die für die Synchronisation relevanten Start- oder Endmuster verloren gehen. Hier erscheint einzig das Neusenden des Payloads als hilfreiche Gegenmaßnahme. Eine Sequenznummer im Payload garantiert hier das Erkennen von verlorenen Informationen beim Empfänger (siehe Abschnitt 3.6).

3.3.2 Fehlererkennung (Integrität) und -korrektur

Problem 3.3.2.1 (Kodierungsproblem) *Welche Verfahren sind am besten geeignet, um mögliche Fehler am Übertragungsweg zu erkennen/beheben?*

Dieses Problem ist wiederum für Sprachübertragungen besonders zu betrachten. Wenn während der Übertragung Fehler auftreten und Bits vertauscht bzw. sogar Pakete verloren gehen, müssen Techniken zur sicheren Erkennung dieser Vorgänge vorhanden sein (wiederum unabhängig vom verwendeten Protokoll für die eigentliche Kommunikation, wie SMTP, HTTP, etc. und deren Mechanismen). Eine offensichtliche und einfach Maßnahme wäre im Fall von Paketverlusten, aber auch bei Bitfehlern, die mehrfache Übertragung des Payloads. Der Empfänger kann die empfangenen Daten danach vergleichen und entscheiden, welche fehlerhaft oder unvollständig sind. Ein klares Problem dieser Methode ist der zusätzliche Aufwand beim Empfänger sowie die Kapazität des Kanals, welche für doppelte (oder sogar dreifache) Redundanz „verschwendet“ werden muss.

Aus diesem Grund ist es sinnvoller, Verfahren zur Integritätssicherung anzuwenden, welche zur Prüfung der Korrektheit der Daten herangezogen werden können. Hierbei werden zwar ebenfalls redundante Daten versandt, aber dies geschieht nach einem bekannten und geregelten Verfahren, um beim Empfänger möglichst die gesamte Information wiederherstellen zu können.

Beispiele hierfür sind der Einsatz von Prüfsummen wie einem *Paritätsbit* oder eines *Cyclic Redundancy Check (CRC)* Codes als Anhang an den Payload. Des weiteren können natürlich auch Hashfunktionen zur Fehlererkennung und beim Einsatz kryptographischer Hashfunktionen auch als Integritätsprüfung herangezogen werden.

3.3.2.1 Paritätsbits

Fehlererkennung und -korrektur durch das Einfügen von Paritätsbits wurde bereits vor zwei Jahrzehnten in den ersten Ansätzen von Redundant Array of Inexpensive Disks (RAID) Systemen beschrieben (vgl. Patterson u. a., 1988).

Die Methodik hat sich seither nicht verändert und stellt eine simple Vorgehensweise zur Korrektur von Daten, welche durch Speicher- und Übertragungsfehler geändert wurden, dar. Durch die Einfachheit ist beim Einsatz von Paritätsbits jedoch zu bedenken, dass nur etwa 50% der Fehlermöglichkeiten erkannt werden. Daraus folgt dass mit Paritätsbits keine sichere Aussage über Fehler in der Datenübertragung getroffen werden kann (vgl. Wikipedia, 2008e).

Grundsätzlich werden bei *Paritätsprüfungen* immer Gruppen von Bits zu sogenannten *Wörtern* zusammengefasst, welchen ein Paritätsbit angehängt wird. Der Wert dieses Bits ist vom verwendeten Paritätsprotokoll abhängig. Es kann entweder gerade Parität (*even parity*) oder ungerade Parität (*odd parity*) signalisiert werden. In jedem Fall wird die Summe der Bits im Wort ermittelt, welche den Wert 1 enthalten. Je nachdem, ob nun gerade oder ungerade Parität verwendet wird, muss das Paritätsbit eine 1 oder eine 0 annehmen (vgl. Wikipedia, 2008e). Tabelle 3.1 demonstriert dies anhand eines einfachen Beispiels.

WORT	PARITÄTSBIT	
	GERADE	UNGERADE
1010001	10100011	10100010
1101001	11010010	11010011

Tabelle 3.1: Beispiel für gerade und ungerade Parität

Durch dieses Beispiel wird auch die zuvor schon erwähnte Schwäche dieses Verfahrens klar. Wenn eine gerade Anzahl an Bits fehlerhaft übertragen werden, dann stimmt die Parität dennoch überein. Es werden also nur Bitfehler erkannt, wenn deren Anzahl ungeradzahlig ist.

Mehrdimensionale Paritätskontrolle. Wird bei diesem Verfahren nicht nur ein Paritätsbit angehängt, sondern eine *mehrdimensionale Paritätskontrolle* eingesetzt, lassen sich Bitfehler nicht nur erkennen, sondern auch korrigieren. Solche Verfahren sind somit als fehlerkorrigierende Verfahren einsetzbar. Ein Beispiel ist der *Hamming-Code* (vgl. Glover und Grant, 2003, S. 364 f).

3.3.2.2 CRC Codes

Zyklische Redundanzprüfung wurde ebenfalls schon vor einigen Jahrzehnten zum ersten Mal beschrieben. Hierbei handelt es sich um ein Verfahren, wo von einem Datenblock vor dem Versenden entsprechend eines Algorithmus ein Prüfwert berechnet und an die eigentlichen Daten angehängt wird. Der Empfänger kann nun die empfangenen Daten durch den angehängten Prüfwert dividieren. Ergibt diese Berechnung Null, wurden die Daten ohne Fehler übertragen.

Cyclic Redundancy Check (CRC) Verfahren beruhen auf der Polynomdivision. Eine

Bitfolge kann als Polynom dargestellt werden, z.B. entspricht 101101 dem Polynom

$$x^5 + x^3 + x^2 + 1.$$

Für die Berechnung des Prüfwertes (CRC-Code) wird ein vordefiniertes *Generatorpolynom* benötigt, welches natürlich bei Sender und Empfänger gleich sein muss. Der zu sendende Datenblock wird nun um eine Anzahl von 0-Bits erweitert, die dem Grad des Generatorpolynoms entspricht. Danach erfolgt eine Berechnung des erweiterten Datenblocks XOR des Generatorpolynoms. Der hierbei erhaltene Rest wird an die ursprünglichen Daten angehängt und übertragen. Der Empfänger muss nun genau wissen, wo die Daten enden und der Prüfwert beginnt, um die Prüfung auf Übertragungsfehler durchführen zu können (vgl. Chung, 2001).

Es gibt viele verschiedene standardisierte CRC Verfahren, wie CRC-24 und CRC-32. Vergleichbar mit dem Einsatz von Paritätsbits ist CRC-1. Die Werte geben den Grad des Generatorpolynoms an, welches ebenfalls für alle Varianten von CRC definiert ist (vgl. Wikipedia, 2008b). Das CRC Verfahren wird unter anderem im TCP eingesetzt damit der Empfänger aufgetretene Fehler während der Übertragung erkennen kann.

3.3.2.3 Hashfunktionen und kryptographische Hashfunktionen

Hashfunktionen sind mathematische Funktionen, welche von einer Zeichenkette beliebiger Länge einen String (*Hashwert*, *Message Digest*) fixer Länge berechnen.

Jede Hashfunktion hat die Anforderung, dass für eine Nachricht M die Ausgabe von $H(M)$ bei jeder Berechnung gleich sein muss. Weitaus strengere Bedingungen stellt die Definition einer *kryptographischen* Hashfunktion (vgl. Menezes u. a., 1996, S. 321 ff):

- Diese Funktionen müssen *Einwegfunktionen* sein. Vom Ergebnis darf nicht auf die ursprüngliche Nachricht geschlossen werden können (*one-way function*).
- Es dürfen nicht zwei verschiedene Nachrichten auftreten, welche den selben Hashwert ergeben (*collision-resistant*).

Bekannte kryptograpische Hashverfahren sind Message Digest 5 (MD5) und die Secure Hash Algorithm (SHA)-Familie, welche einen Hashwert von 128 bis zu 512 Bits aus der Nachricht errechnen (vgl. Wikipedia, 2008c).

3.3.2.4 Vorgehensweise in OpenPGP

Als Beispiel für eine mögliche Kodierung der Daten soll die Verfahrensweise von OpenPGP (eine freie PGP Implementierung) betrachtet werden. Nachrichten in OpenPGP werden in einem Binärformat gespeichert, welches 8-Bit pro Zeichen zur Darstellung verwendet. Bei der Übertragung wird eine *Radix-64* Konvertierung vorgenommen, welche 8-Bit Zeichen als 7-Bit Zeichen (ASCII Kodierung) darstellt. Durch die

ASCII Kodierung wird sichergestellt, dass die Daten auf allen Systemen korrekt bearbeitet werden können, auch wenn nur der ASCII Zeichensatz unterstützt wird (*ASCII Armor* Verfahren). Radix-64 entspricht einer *Base-64* Kodierung der Daten, jedoch wird außerdem noch eine 24-Bit Prüfsumme an die Daten angehängt um Fehler am Übertragungsweg feststellen zu können. Die Prüfsumme ist somit ein CRC-24 Code (vgl. Callas u. a., 2007).

3.3.2.5 Beantwortung des Kodierungsproblems

Wie bei der Diskussion über Synchronisation (siehe Abschnitt 3.3.1) wurde auch zu Beginn der Behandlung von Kodierung und Integritätssicherung eine Problemstellung definiert welche nun beantwortet werden soll.

Welche Verfahren sind am besten geeignet, um mögliche Fehler am Übertragungsweg zu erkennen/beheben?

Um Fehler erkennen (und eventuell beheben) zu können, sind Algorithmen anzuwenden, welche eine Sicherung der Integrität gewährleisten. Die Auswahl reicht von einfachen (aber nicht vollständig sicheren) Verfahren wie Paritätsbits über mehrdimensionale Paritätskontrolle, bis zum Einsatz von kryptographischen Hashfunktionen. Diese benötigen zwar mehr Leistung bei der Berechnung (vgl. Contini u. a., 2007), stellen aber eine (fast) 100%-ige Möglichkeit dar, Fehler (gewollt vom Man in the Middle oder ungewollt) während der Übertragung zu erkennen. Als „günstigere“ Alternative setzt z.B. OpenPGP eine CRC Prüfsumme zum Schutz vor Übertragungsfehlern ein (vgl. Callas u. a., 2007).

Neben der Integritätssicherung ist es auch wichtig, dass alle Systeme die Darstellung der zu übertragenden Daten unterstützen. Wie bei OpenPGP implementiert (siehe Abschnitt 3.3.2.4), stellt das ASCII Verfahren hierbei den kleinsten gemeinsamen Teiler an Kodierungsverfahren dar. Es bietet sich daher an, alle Daten mit einem Verfahren wie Base-64 auf ihre ASCII Darstellung umzuwandeln bevor diese als Payload eingebettet werden.

3.3.3 Kompression

Das OpenPGP Verfahren unterstützt die Kompression der zu übertragenden Daten (vgl. Callas u. a., 2007). Diese Technik ist auch für die steganographischen Protokolle interessant, da es durchaus üblich ist, dass die Kapazität des Kommunikationskanals nicht sehr groß ist (wie es z.B. bei Bildern der Fall wäre). Um den Payload im Cover nicht zu erkennen, sollte nur ein gewisser Anteil des Covers für die einzubettenden Daten verwendet werden. Durch Einsatz von Kompression gelingt es so, die Kapazität zu erhöhen, indem die Größe des Payloads verringert wird.

Bekannte Algorithmen zur Kompression sind *ZIP*, *ZLib* und *BZip2*. Diese werden auch von OpenPGP Implementierungen angeboten (vgl. Callas u. a., 2007).

3.4 Nachrichtenformat - Aushandeln des Parametersatzes

Die Übertragung des Parametersatzes mit allen nötigen Informationen wird in der ersten Protokollphase durchgeführt. In dieser Phase findet zumindest eine Übertragung eines Payloads mit den hier beschriebenen Informationen statt (für genaue Beschreibung der Protokollphasen siehe Kapitel 4). Zum Einbetten dieses ersten Payloads gibt es mehrere Möglichkeiten. Einerseits kann der Initiator der Kommunikation den öffentlichen Parametersatz des Empfängers bzw. einen für die Protokolle definierten (ebenfalls öffentlichen) Parametersatz verwenden. Das bedeutet, dass die Informationen zur Einbettung in diesem Schritt auch für Angreifer verfügbar sind. Demnach sollte die Informationsmenge möglichst gering sein, um durch das Einbetten keine Aufmerksamkeit zu Erregen (oder das Cover sehr groß um es entsprechend verteilen zu können).

Eine andere (sichere) Möglichkeit besteht darin, dass die Kommunikationspartner zuvor auf einem vertraulichen Weg (sicherer Kanal) die Informationen austauschen, welche danach für diese erste Übertragung herangezogen werden können. Dies stellt eine sicherere Alternative zum öffentlich verfügbaren Parametersatz dar.

Unabhängig vom gewählten Weg muss der Initiator auf jeden Fall die Informationen vor dem Einbetten verschlüsseln. Hierfür kann z.B. der öffentliche (kryptographische) Schlüssel des gewünschten Kommunikationspartners verwendet werden. Diese Verfahren werden durch das eingesetzte Protokoll (siehe Kapitel 4), sowie den Basisparametersatz bestimmt.

In der Praxis reichen die genannten Daten (Stego-Verfahren und eventuell kryptographische Informationen) jedoch nicht aus, um bei allen verfügbaren Covern eine problemlose Erkennung der eingebetteten Daten zu gewährleisten. Deshalb muss die erste Information etwas erweitert werden, um einerseits eine gewisse Redundanz zur einfacheren Erkennung zu erreichen und andererseits auch wichtige Protokollinformationen wie eine Versionsnummer und Informationen zur Fehlererkennung und -korrektur aufzunehmen.

3.4.1 Aufbau eines Parametersatzes

In der vollständigen Ausprägung enthält ein Parametersatz folgende Informationen:

- Versionsnummer,
- Protokollnummer,
- steganographisches Verfahren (Einbettung) + Parameter und Schlüssel,
- kryptographisches Verfahren + Parameter und Schlüssel,
- Startbedingungen (Muster) und Synchronisationsregeln.

Je nach gewähltem Protokoll sind zusätzliche Informationen erforderlich, wie ein *Hashwert* über später folgende Informationen, Angaben zum Einsatz von *Biometrie*, eine *digitale Signatur* über den gesamten Payload sowie Codes zur Fehlerkorrektur. Damit Fehler am Übertragungsweg erkannt werden können, ist am Ende ein Prüfwert in Form eines CRC anzuhängen.

Das Nachrichtenformat für die Aushandlung des Parametersatzes ist in Anhang A.1 dargestellt. Die folgenden Beschreibungen der Felder beziehen sich auf diese Darstellung.

3.4.2 Optionale Informationen

Wie in der Beschreibung des Nachrichtenformates zu erkennen ist, sind fast alle Angaben optional. Dies ist deshalb der Fall, da in der ersten Protokollphase (siehe Abschnitt 4.2.1) bei bidirektionaler Kommunikation eine Antwort auf den ersten Payload (gewählten Parametersatz) erforderlich ist, welche entweder Werte ändert oder das Angebot akzeptiert (in diesem Fall werden alle optionalen Felder weggelassen).

3.4.3 Version

An erster Stelle erfolgt die Versionsangabe. Der Wert wird in einem Byte abgebildet und bezeichnet die Version des Protokolls. Dies darf nicht mit der Protokollvariante verwechselt werden, welche mit diesem Feld nicht gekennzeichnet wird.

Der Defaultwert ist 0x01.

3.4.4 Protokoll

Ebenfalls in einem Byte dargestellt ist die eingesetzte Protokollvariante. Definiert sind die Werte für alle in diesem Dokument beschriebenen Protokolle, siehe Tabelle 3.2.

PROTOKOLL	
GSP-I	0x01
GSP-II	0x02
GSP-III	0x03
GSP-IV	0x04

Tabelle 3.2: Protokollfeld im Parametersatz

3.4.5 Steganographisches Verfahren

Bei der Wahl des zu verwendenden Einbettungsalgorithmus ist aus der definierten Liste (Feld *Algorithm*) der Verfahren (siehe Anhang A.1) das passende auszuwählen. Die

möglichen Werte sind ebenfalls bereits in der ASN.1 Darstellung definiert, die Liste ist jedoch einfach erweiterbar. Des weiteren sind für das jeweilige Verfahren die Parameter für das Einbetten anzugeben. Hierbei steht das Feld *Parameter* zur Verfügung, welches beliebige Daten beinhalten kann.

(Eine genaue Definition des Formats ist aufgrund der verschiedenen Anforderungen abhängig vom gewählten Einbettungsalgorithmus nicht möglich.)

3.4.6 Kryptographisches Verfahren

Da der Payload in jedem Fall verschlüsselt wird, sind Angaben über die kryptographischen Algorithmen nötig.

Der erste Wert (*Algorithm*) definiert, welcher Verschlüsselungsalgorithmus für die Daten angewandt werden soll. Hierbei sind sowohl asymmetrische (Public-Key Kryptographie), als auch symmetrische (Secret-Key Kryptographie) Verfahren möglich. Die verwendbaren Werte sind in Anhang A.1 dargestellt, wobei die Liste einfach erweiterbar ist. Als optionale Parameter (*KeyLength, Key*) kann der zu verwendende Schlüssel und dessen Länge angegeben werden. Dies ist vor allem für symmetrische Verfahren nötig, wenn der Schlüsselaustausch auf diesem Weg durchgeführt werden soll.

3.4.7 Synchronisation

Für die Synchronisation von Sender und Empfänger können grundsätzlich verschiedene Methoden eingesetzt werden. In Anhang A.1 ist die Methode *Pattern* definiert.

Werden Streaming-Daten, wie VoIP Gespräche, als Cover verwendet, sind alle drei Werte anzugeben. Der erste Wert (*HasEmbData*) gibt die Bitfolge an, welche im Cover auftreten muss, damit der Empfänger danach nach eingebetteten Daten sucht. *StartAt* und *EndAt* sind wiederum die Bitmuster, welche vor bzw. nach dem Payload in den Datenstrom vom Sender eingebettet werden müssen, um eine klare Erkennung des Payloads in den empfangenen Daten zu gewährleisten.

3.4.8 Hashwert

Der Hashwert dient dem Protokoll GSP-III. Die genaue Beschreibung des Anwendungsbereichs im Rahmen von GSP-III folgt in Abschnitt 4.6. Hier wird vorerst auf die Darstellung des Hashwertes eingegangen. Wird das genannte Protokoll nicht verwendet, ist auch dieses Feld nicht verwendbar.

Beim Hashwert (Feld *Hash*) ist zuerst der verwendete Algorithmus anzugeben (*Algorithm*). Die Liste der definierten Werte findt sich wiederum in Anhang A.1 (wobei wie bei den Verschlüsselungsalgorithmen und steganographischen Verfahren auch gilt, dass die Liste einfach zu erweitern ist). Der zweite Wert (*FutureHash*) stellt das Ergeb-

nis einer im ersten Feld gewählten Hashfunktion dar. Der letzte Wert welcher für das Hashing herangezogen werden kann, ist *HashedData*. Mit diesem Element kann der Sender kennzeichnen, welche Daten der Hashwert repräsentiert (z.B. durch Angabe einer eindeutigen Kennung).

3.4.9 Biometrie

Bei den verfügbaren biometrischen Merkmalen sind all jene in der Beschreibung des Parametersatzes eingetragen, welche vom ISO/IEC 7816-11 Standard für biometrische Authentifizierung mittels Chipkarten festgelegt wurden (vgl. Struif, 2002). Hierbei handelt es sich um eine ziemlich vollständige Aufzählung möglicher Merkmale (siehe Anhang A.1), was auch durchaus wünschenswert ist, da (eventuell) nicht alle Methoden gleichermaßen bei allen Menschen oder in allen Situationen sinnvoll angewendet werden können. Zu Beachten ist ebenfalls, dass auch die Option *Password* unter die biometrischen Merkmale fällt.

Diese Merkmale können anstelle von Hashwerten, digitalen Signaturen, etc. für eine Authentifizierung herangezogen werden.

3.4.10 Digitale Signatur

Das Anfügen einer digitalen Signatur ist optional und für die Protokolle GSP-II und GSP-IV relevant (siehe Abschnitte 4.5 und 4.7). Bei Verwendung ist das Feld *Signature* mit drei Werten zu belegen. Zu Beginn wird der kryptographische Algorithmus angegeben, welcher für die Erstellung und Prüfung der digitalen Signatur verwendet wird (Feld *Algorithm*). Optional unter den Werten zur Signatur ist ein kryptographischer Schlüssel (*Key*). Dieser muss in Fällen angegeben werden, in welchen z.B. seperate Schlüssel zum Verschlüsseln und Signieren verwendet werden sollen. Der finale Wert ist das Ergebnis der Signaturberechnung (*SignatureValue*).

3.4.11 Wichtigkeit der Daten

Der Initiator der Verbindung kann optional angeben, wie „wichtig“ vom ihm gesendete Daten vom Empfänger eingestuft werden sollen. Daraus ergibt sich eine Anforderung für die Stabilität der Verbindung, welche unter *Importance* (*Mode*) definiert werden kann. Die Auswahl kann im einfachsten Fall das mehrmalige Senden der Daten bedeuten, über das Feld *Parameter* können aber auch fehlerkorrigierende Codes und sonstige Daten zur Erhöhung der Fehlertoleranz eingefügt werden. Des weiteren steht noch die optionale Angabe einer Priorität (Feld *Priority*) zur Verfügung. Dies kann in Fällen eingesetzt werden, wenn zwischen Sender und Empfänger klar ausgehandelt wurde, wie mit verschiedenen Prioritätsklassen zu verfahren ist (ähnlich einem Quality of Service (QoS) Mechansimus).

Wie unter Anhang A.1 dargestellt, sind in dieser Arbeit die Methode des öfteren Versendens der zu übertragenden Daten (*MultipleTransmission*), sowie das Anhängen von Paritätsbits (*Parity*) im Parametersatz aufgenommen. Es steht Implementierungen wiederum offen, weitere Verfahren in die Liste aufzunehmen.

Aus der definierten Wichtigkeit der Daten ergibt sich für den Empfänger auch die Vorgehensweise am Ende der Kommunikation, ob nicht empfangene Daten nochmals angefordert werden müssen (siehe Abschnitt 3.6.6).

3.4.12 ACK und NACK Mechanismus

Für die erste Protokollphase (siehe Abschnitt 4.2.1) ist ein einfacher Mechanismus zum Bestätigen oder Ablehnen des empfangenen Parametersatzes anzuwenden. Hierfür ist das Feld *ACK* mit den Ausprägungen *true* und *false* verfügbar. Akzeptiert der Empfänger den Parametersatz, muss eine Nachricht zurückgeschickt werden, wo ACK auf *true* gesetzt ist (entspricht einem ACK, der Empfänger akzeptiert den Parametersatz). Andernfalls ist ACK auf *false* zu setzen und die gewünschten Änderungen sind ebenfalls mitzuschicken.

Wenn keine Kombination von Verfahren möglich ist um eine Kommunikation aufzubauen, kann mit einer „leeren“ Nachricht und einem ACK gleich *false* (was einem *NACK*, der Ablehnung des Parametersatzes, entspricht) die Kommunikation abgebrochen werden.

Für genauere Beschreibungen zu diesem Mechanismus siehe Abschnitt 4.2.1.1.

3.4.12.1 Unidirektionale oder bidirektionale Kommunikation

Es ist zu beachten, dass der Parametersatz alle Werte aus Sicht des Initiators der Verbindung beschreibt. Besonders bei kryptographischen Schlüsseln ist diese Tatsache relevant. Das Feld *Key* unter *KrytoData* (siehe Anhang A.1) beinhaltet jenen Schlüssel, welcher für Datenverkehr vom Empfänger zum Sender eingesetzt werden kann. Des weiteren enthält das Feld *Key* unter *SignatureData* nur jenen Schlüssel, welcher zur Überprüfung einer digitalen Signatur vom Empfänger herangezogen werden kann.

Soll die Kommunikation nun bidirektional erfolgen, muss der Empfänger zusätzlich zu einem ACK oder NACK Parametersatz auch seine entsprechenden kryptographischen Schlüssel mitteilen. Durch diesen Mechanismus kann gewährleistet werden, dass beide Kommunikationspartner verschlüsselte Inhalte des jeweils Anderen entschlüsseln und auch die (optionale) digitale Signatur überprüfen können. Abbildung 3.2 stellt dies für die beiden kryptographischen Schlüssel graphisch dar.

Erfolgt die Kommunikation nur in eine Richtung, muss der ACK/NACK Mechanismus natürlich entfallen. In diesem Fall kann der Empfänger den angebotenen Parametersatz nicht durch Rückantworten verändern und muss ihn somit akzeptieren.

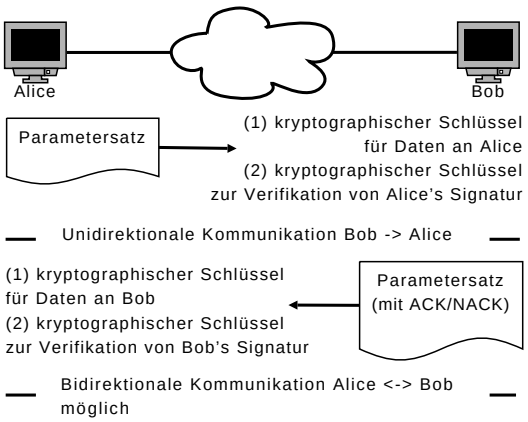

Abbildung 3.2: Uni- und bidirektionale Kommunikation

3.4.13 CRC

Jeder Nachricht ist eine Prüfsumme als letzter Wert anzuhängen. Wie bei OpenPGP wird ein CRC eingesetzt (vgl. Callas u. a., 2007), da diese Verfahren zuverlässig arbeiten und keinen zu hohen Aufwand bei der Erstellung bedeuten.

Als CRC Variante wird ein CRC-32 verwendet, welcher ein Ergebnis von 32 Bits berechnet. Dies ist ein anerkanntes Verfahren und wurde z.B. vom Institute of Electrical and Electronics Engineers (IEEE) für den Ethernet Standard herangezogen um die Frame Check Sequence (FCS) im Media Access Control (MAC) Header zu berechnen (vgl. IEEE, 2005, S. 52).

3.4.14 Tags für BER-Kodierung

In der folgenden Tabelle 3.3 werden alle Tags für die in Anhang A.1 beschriebenen Datenfelder definiert. Besitzt das jeweilige Feld eine fixe Länge wird diese ebenfalls dargestellt. Ein leerer Eintrag im Feld „Länge“ kennzeichnet, dass das betroffene Feld eine variable Länge besitzt.

3.5 Abkürzen des Payloads

Bei der Übermittlung eines Parametersatzes mit dem Nachrichtenformat aus Anhang A.1 entsteht bei vielen Kommunikationen möglicherweise ein Overhead. Deshalb bestehen zwei Möglichkeiten, um einen gewünschten Parametersatz zu übertragen, ohne dabei das gesamte Nachrichtenformat verwenden zu müssen:

- Start der Kommunikation mit ACK oder NACK
- Verwendung von Tags zur Abkürzung von definierten Parametersatzen

	KLASSE (ASN.1 CLASS)	TYP (TAG)	LÄNGE (IN BYTE)
GSP	Private	0xe1	
VERSION	Universal	0x04	0x01
PROTOCOL	Universal	0x04	0x01
STEGANOGRAPHY	Private	0xe2	
ALGORITHM	Private	0xc3	0x01
PARAMETER	Universal	0x04	
CRYPTOGRAPHY	Private	0xe4	
ALGORITHM	Private	0xc5	0x01
KEYLENGTH	Universal	0x02	
KEY	Universal	0x04	
SYNCHRONISATION	Private	0xe6	
PATTERN	Private	0xe7	
HASEMBDATA	Universal	0x03	0x01
STARTAT	Universal	0x03	0x01
ENDAT	Universal	0x03	0x01
HASH	Private	0xe8	
ALGORITHM	Private	0xc9	0x01
FUTUREHASH	Universal	0x03	
HASHEDDATA	Universal	0x04	
BIOMETRY	Private	0xca	0x01
SIGNATURE	Private	0xeb	
ALGORITHM	Private	0xcc	0x01
KEY	Universal	0x04	
SIGNATUREVALUE	Universal	0x04	
IMPORTANCE	Private	0xed	
MODE	Private	0xce	0x01
PARAMETER	Universal	0x04	
PRIORITY	Universal	0x02	
ACK	Universal	0x01	0x01
CRC	Universal	0x04	

Tabelle 3.3: Tags für das Nachrichtenformat - Parametersatz

3.5.1 ACK/NACK im ersten Schritt

Wird im ersten Payload bereits ein ACK gesendet, signalisiert dies dem Empfänger, dass der letzte ausgemachte Parametersatz für die aktuelle Verbindung herangezogen werden soll. Diese Möglichkeit ist sinnvoll, wenn beide Kommunikationspartner den letzten Parametersatz sicher speichern konnten und dieser außerdem eindeutig identifiziert werden kann, um wiederverwendet werden zu können.

Eine weitere Möglichkeit besteht darin, ein NACK im ersten Payload mitzusenden. Mit dieser Methode kann der Initiator Veränderungen am letzten ausgemachten Para-

metersatz vornehmen. Parameter, welche nicht angegeben werden, werden somit vom letzten Parametersatz übernommen.

3.5.2 Tag-Konzept

Damit für die gleichen Parametersätze nicht immer wieder alle Daten wie in Anhang A.1 beschrieben übertragen werden müssen, wird das Konzept der *Tags* eingeführt. Dies sind vordefinierte, komplette Beschreibungen von Parametersätzen, welche in einigen wenigen Bytes kodiert übertragen werden können. Dies bedeutet, dass nur dynamische Informationen (wie kryptographische Schlüssel) zusätzlich zu dem gewünschten Tag übertragen werden müssen.

Für Implementierungen der in diesem Dokument beschriebenen Verfahren ist es möglich, dass eigene Tags für bestimmte Parametersätze definiert werden. In Anhang A.3 sind einige Beispiele für dieses Tag-Konzept dargestellt.

3.6 Nachrichtenformat - Datenübertragung

Nachdem das Format für den Payload bei der Übertragung des Parametersatzes beschrieben wurde ist es auch erforderlich, ein definiertes Format für die eigentliche Datenübertragung zu beschreiben (Phase II im Protokollablauf - siehe Abschnitt 4.2.2). Die Informationen werden nach dem Aushandeln der Parameter zum Einbetten, Verschlüsseln, etc. gemäß dieser Parameter im Cover versteckt. Jedoch sind auch hierbei einige zusätzliche Informationen wichtig.

Das Nachrichtenformat für die Datenübertragung ist in Anhang A.2 dargestellt. Die Felder *Version*, *Protocol* und *CRC* entsprechen den gleichnamigen Feldern im Nachrichtenformat für die Übertragung des Parametersatzes (siehe Abschnitt 3.4). Die möglichen Werte und Bedeutungen können somit den Abschnitten 3.4.3, 3.4.4 und 3.4.13 entnommen werden.

3.6.1 Sequenznummer

In Netzwerken werden zu versendende Daten in Pakete bestimmter Größe gepackt und, zusammen mit Steuerinformationen, einzeln versandt. Auch wenn die steganographischen Informationen in einen Datenstrom (wie bei VoIP) eingebettet werden, sendet die darüberliegende Applikation (der VoIP Client) wiederum einzelne Pakete an die Zielstation. Auch bei anderen Anwendungen wie dem e-Mail Verkehr ist das der Fall. Da in Netzwerken wie dem Internet keine Signalisierung stattfindet und es häufig mehr als einen Weg zu einem bestimmten Ziel gibt kann es vorkommen, dass Pakete über unterschiedliche Hops zum Empfänger gelangen. Dies wiederum impliziert die Tatsache, dass Pakete unterschiedliche Latenzzeiten aufweisen können und es möglich ist, dass später gesendete Pakete früher beim Empfänger ankommen als ihre

Vorgänger. Ebenfalls vorstellbar ist es, dass Pakete verloren gehen. Die Protokolle (wie TCP), auf welche Applikationen wie e-Mail Clients aufsetzen, sind in solchen Fällen für das erneute Senden des verlorenen Pakets verantwortlich.

Aus den genannten Gründen ist es sinnvoll, eine Sequenznummer (Feld *SequenceNumber*) zusätzlich zu den eingebetteten Informationen mitzusenden. Dies ermöglicht es einerseits dem Empfänger, die steganographischen Informationen zu ordnen um die Daten in richtiger Reihenfolge verarbeiten zu können. Außerdem kann durch diese Sequenznummer ein einfacher Mechanismus zum Anfordern von nicht empfangenen Sequenznummern (Paketen) implementiert werden (unabhängig vom eingesetzten Kommunikationsprotokoll – anders als beim TCP werden nämlich z.B. beim UDP verloren gegangene Pakete nicht erkannt und somit auch nicht erneut gesendet).

3.6.2 Daten

Das Feld *Payload* beschreibt die eigentlichen Daten, welche zwischen den Kommunikationspartnern übertragen werden sollen. Hierbei handelt es sich wirklich nur um die „Rohdaten“ ohne Prüfsummen, Redundanzen und Ähnlichem.

Bei der Übertragung erfolgt keine Interaktion zwischen den Kommunikationspartnern, die Daten werden in Blöcken unterteilt in mehreren Paketen in einem Schritt gesendet (eingebettet).

3.6.3 Erkennung des letzten Payloads

Für den Empfänger kann es wichtig sein, das Ende der Übertragung zu erkennen. Mit dem Feld *DataEnd* kann der Sender dem Empfänger mitteilen, dass dieser Payload die letzten zu übertragenden Daten dieser Kommunikationssitzung enthält (mögliche Werte sind *true* und *false*). Ähnlich dem ACK/NACK Mecahnismus (siehe Abschnitt 3.4.12) kann der Empfänger nun nach dem Ende der Übertragung eventuell nicht erhaltene Daten neu anfordern. Ob Daten während der Übertragung verloren gingen kann über das Feld *SequenceNumber* verifiziert werden.

Für eine genaue Beschreibung des nachträglichen Anforderns von Daten in Phase II siehe Abschnitte 3.6.6 und 4.2.2.2.

3.6.4 Informationen zur Authentifizierung und digitale Signatur

Welche Authentifizierungsmethoden angewandt werden, hängt vom verwendete Protokoll (siehe Kapitel 4) und von dem ausgehandelten Parametersatz ab. Wird eine Authentifizierung verwendet, kann der entsprechende Wert in das Feld *AuthData* eingetragen werden.

Des weiteren besteht noch die Möglichkeit einer digitalen Signatur (siehe Parameter-

satz in Anhang A.1), welche im Feld *Signature* dargestellt werden kann.

3.6.5 Fehlerkorrektur

Abhängig von der ausgehandelten „Wichtigkeit der Daten“ (siehe Abschnitt 3.4.11) ist es bei der eigentlichen Übertragung eventuell notwendig, entsprechende Parameter zur Fehlerkorrektur einzufügen. Hierfür steht das Feld *CorrectionCode* (unter *ErrCorrection*) zur Verfügung.

3.6.6 Nachträgliches Anfordern von Daten

Ähnlich zum ACK/NACK Mechanismus beim Aushandeln des Parametersatzes (siehe Abschnitt 3.4.12) besteht am Ende des Datenaustausches die Möglichkeit, nicht empfangene Elemente neu anzufordern. Das Feld *ResendInfo* kann daher nur dann sinnvoll verwendet werden, wenn zuvor vom Kommunikationspartner *DataEnd* mit *true* belegt wurde. Für den Sender (oder Empfänger) besteht nun die Möglichkeit, im Feld *MissingSeqNrs* jene Sequenznummern einzutragen, welche nicht empfangen wurden. Ob (und in welcher Ausprägung) diese Methode verwendet wird, hängt von der „Wichtigkeit der Daten“ ab, welche vom Sender der Daten im Parametersatz definiert werden kann (siehe Abschnitt 3.4.11). Da dieses Feld (*ResendInfo*) optional ist, kann dementsprechend auch auf ein nachträgliches Anfordern von Daten verzichtet werden.

3.6.7 Tags für BER-Kodierung

Wie für das Nachrichtenformat des Parametersatzes (siehe Abschnitt 3.4) werden auch hier die Tags definiert, welche für die Datenfelder aus Anhang A.2 zu verwenden sind – siehe Tabelle 3.4.

3.6.8 Probleme mit einem Man in the Middle

Es ist darauf zu achten, dass ein möglicher Man in the Middle Informationen im definierten Nachrichtenformat nicht unerkannt abändert und somit z.B. die Authentifizierung zwischen den Kommunikationspartnern kompromittiert. Damit dies nicht geschieht werden einerseits die allgemeinen Möglichkeiten wie das Verschlüsseln des Payloads vor dem Einbetten angewandt. Außerdem ist die Protokollvariante, welche für die Kommunikation zur Anwendung kommen soll, im Parametersatz wie auch in den Zusatzinformationen zu jedem Payload integriert. Da das Protokoll auch die Authentifizierungsmethode definiert, ist dies für die Kommunikationspartner schon durch den Parametersatz eindeutig geregelt. Wenn also nach dem Aushandeln des Parametersatzes abgeänderte Informationen ankommen, dann muss mit einem kompromittierten Kanal gerechnet werden. Informationen, welche sich zwischen dem Parametersatz und den weiteren Daten nicht verändern sollten, umfassen:

	KLASSE (ASN.1 CLASS)	TYP (TAG)	LÄNGE (IN BYTE)
DATA	Private	0xe1	
VERSION	Universal	0x04	0x01
PROTOCOL	Universal	0x04	0x01
SEQUENCENUMBER	Universal	0x02	
PAYLOAD	Universal	0x04	
DATAEND	Universal	0x01	0x01
AUTHSIG	Private	0xe2	
AUTHDATA	Universal	0x04	
SIGNATURE	Universal	0x04	
ERRCORRECTION	Private	0xe3	
CORRECTIONDATA	Universal	0x04	
RESENDINFO	Private	0xe4	
MISSINGSEQNRS	Universal	0x11	
CRC	Universal	0x04	

Tabelle 3.4: Tags für das Nachrichtenformat - Daten

- die *Version* des Protokolls,
- die gewählte Protokollvariante,
- die Authentifizierung in den Zusatzinformationen des Payloads muss dem gewählten Protokoll, d.h. den Methoden des ausgehandelten privaten Parametersatz, entsprechen.

Kapitel 4

Protokollphasen und Protokolle

Im vorigen Kapitel wurden wichtige Rahmenbedingungen für die Protokolle beschrieben und die Parametersätze definiert, welche die zu verwendenden Verfahren (z.B. Methoden der Steganographie und Kryptographie) beschreiben. Dieses Kapitel geht nun genau auf die Phasen ein, welche im Protokollablauf verwendet werden. Danach folgt eine Definition der Protokollvarianten selbst.

4.1 Aufgabenbereich der Protokolle

Bevor auf die Protokollphasen und Protokolle eingegangen wird, soll hier noch kurz hervorgehoben werden, was die Relevanz der beschriebenen Protokolle darstellt.

Die Protokolle bieten

- eine versteckte Kommunikation,
- Vertraulichkeit,
- Authentizität,
- Integrität und Fehlertoleranz

für die steganographische Kommunikation zwischen zwei Entitäten. Es ist jedoch darauf zu achten, dass nicht alle vier Eigenschaften bei allen in diesem Kapitel beschriebenen Protokollen zutreffen. Dies ist im Sinne des Designs, da mehrere Protokolle vorgestellt werden um für verschiedene Situationen (z.B. Wichtigkeitsstufe der Kommunikation) eine Basis zu schaffen.

4.2 Protokollphasen

Alle Protokolle arbeiten in zwei Schritten. Im ersten Schritt wird unter Verwendung einer kleinen Auswahl von vordefinierten Parametersätzen (*Basisparametersätze*, siehe Definition in Abschnitt 3.1) auf einen geheimen Parametersatz umgeschaltet, welcher für die zweite Phase, der eigentlichen steganographischen Datenübertragung, herangezogen wird. Eine Ausnahme beschreiben hierbei die Parameter der Kryptographie, welche sofort nach dem ersten Schritt von Phase I (Übermittlung des privaten Parametersatzes) auf die erhaltenen Methoden umgeschaltet werden soll. Dies stellt eine Erhöhung der Sicherheit dar und kann ohne ACK/NACK durchgeführt werden, da zu erwarten ist, dass alle gängigen kryptographischen Mechanismen bei den Kommunikationspartnern implementiert sind.

Hierbei werden zwei unterschiedliche Nachrichtenformate verwendet, einerseits für das Aushandeln des Parametersatzes (siehe Abschnitt 3.4) und andererseits für die eigentliche (steganographische) Datenübertragung (siehe Abschnitt 3.6).

Außerdem gibt es noch ein drittes Nachrichtenformat (*Tags*, siehe Abschnitt 3.5.2), welches ebenfalls (optional) in der ersten Phase eingesetzt werden kann um den zu übertragenden Payload möglichst kurz zu halten.

4.2.1 Protokollphase I

Wie schon erwähnt dient die erste Phase im Protokollablauf dazu, einen geheimen Parametersatz auszumachen, welcher für den weiteren Verlauf als Basis zur Einbettung der Informationen und zur Sicherheit der Kommunikation dient. Alle hier beschriebenen Protokolle enthalten definierte Basisparametersätze, aus welchen der Sender einen Parametersatz für die erste Phase auswählen kann.

Der Ablauf sieht nun vor, dass der Sender einen dieser Basisparametersätze heranzieht, wenn er eine steganographische Kommunikation beginnen möchte. Eine andere Möglichkeit besteht darin, dass ein Parametersatz verwendet wird, welcher vom Sender schon auf vertraulichem Weg an gewünschte Kommunikationspartner verteilt wurde. Wenn der Parametersatz nun feststeht, wird vom Sender der erste Payload eingebettet und verschickt. Dieser erste Payload setzt sich wiederum aus dem vom Sender gewählten Parametersatz für die eigentliche Kommunikation (Phase II) zusammen.

Da für das Einbetten dieser geheimen Informationen die Basisparametersätze herangezogen werden können, sind an diese hohe Anforderungen im steganographischen Sinne gestellt. Die Kommunikation muss sicher und versteckt erfolgen, damit kein Angreifer den geheimen Parametersatz „entdecken“ kann.

Wenn nun Daten bei einem potentiellen Kommunikationspartner eintreffen, muss dieser alle verfügbaren Basisparametersätze durchprobieren, um erkennen zu können, ob steganographische Informationen eingebettet wurden. Dieser Schritt entfällt bei Verwendung eines geheimen Parametersatzes in Phase I. Auf jeden Fall bedeutet der

Empfang von steganographischen Daten gleichzeitig die Initiierung von Phase I beim Empfänger und somit, dass der Payload eine Beschreibung des vom Sender gewählten geheimen Parametersatzes enthält. Abbildung 4.1 zeigt den genauen Ablauf dieser Schritte zwischen zwei Kommunikationspartnern.

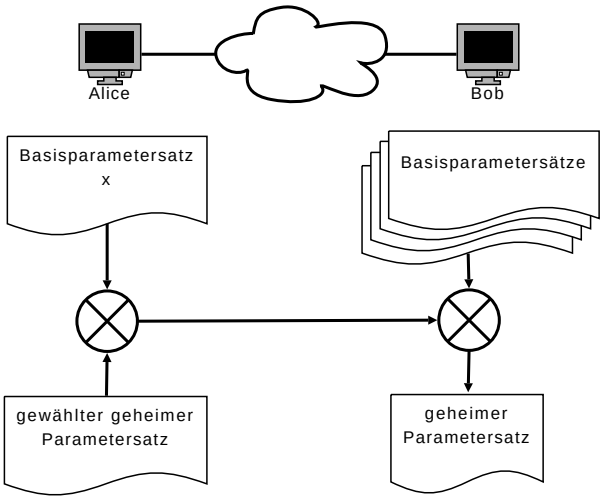

Abbildung 4.1: Ablauf der ersten Schritte von Phase I

Wie die Abbildung zeigt, haben nach diesem Schritt beide Kommunikationspartner einen geheimen Parametersatz, welcher für die zweite Phase des Protokollablaufs herangezogen werden kann.

4.2.1.1 ACK und NACK Mechanismus

Es kann nun der Fall eintreten, dass der Empfänger den vom Sender gewählten Parametersatz ablehnt. Mögliche Gründe hierfür sind, dass der Empfänger ein gewähltes Verfahren nicht unterstützt, eine zu kurze Schlüssellänge aufgrund seines hohen Sicherheitslevels nicht zulassen kann oder nur ein bestimmtes der nachfolgend beschriebenen Protokolle implementiert.

Um diesen Fall ebenso abhandeln zu können, sieht Phase I vor, dass vom Empfänger eine Antwort an den Initiator der Kommunikation gesendet werden muss nachdem der geheime Parametersatz empfangen wurde. Dieser Ablauf ist natürlich nur bei bidirektionaler Kommunikation möglich. Werden die Daten nur in eine Richtung übertragen, muss der Empfänger den erhaltenen Parametersatz ohne Rückantwort akzeptieren (siehe Abschnitt 3.4.12.1).

Der Empfänger muss für seine Antwort das optionale Feld *ACK* des Parametersatzes mit einem Wert belegen (siehe Nachrichtenformat unter Anhang A.1 und Beschreibung des Feldes ACK unter Abschnitt 3.4.12). Verfügbare Werte sind hierfür *true* und *false*.

- Ein Wert von *true* (ACK) in der Antwortnachricht teil dem Initiator mit, dass der gewählte Parametersatz vom Empfänger akzeptiert wurde.

- Ist das Feld ACK auf *false* gesetzt (NACK), muss der Empfänger seine Änderungen in dem zurückgesendeten Parametersatz an den Initiator der Kommunikation mitteilen.

Danach ist es für den Sender natürlich wiederum möglich, Anpassungen vorzunehmen und diese in einer Antwort (mit gesetztem NACK) an den Empfänger zurückzusenden.

Im Endeffekt soll ein Parametersatz ausgehandelt sein, welcher für beide Kommunikationspartner in Phase II anwendbar und auch sicher für eine steganographische Übertragung ist. Sollte einer der beiden Kommunikationspartner die Änderungen nicht akzeptieren können, aber auch keine erneuten Änderungen vornehmen, kann die Verbindungen abgebrochen werden. Dafür ist ein Parametersatz mit gesetztem NACK und ohne zusätzliche (geänderte) Parameter zu senden (siehe Abschnitt 3.4.12). Dies signalisiert dem Empfänger dieses Payloads, dass der Parametersatz zwar nicht akzeptiert werden kann, aber auch keine Änderungen seitens des Kommunikationspartners durchgeführt wurden. In diesem Fall wird die Kommunikation in Phase I abgebrochen.

Daraus ergibt sich ein individueller Protokollablauf bei bidirektionaler Kommunikation, wie in Abbildung 4.2 dargestellt. Das Nachrichtenformat für die Antworten entspricht dem des ersten Kommunikationsschritts von Phase I (siehe Anhang A.1), wobei die gewünschten Werte abgeändert zurückgeschickt werden. Akzeptierte Parameter sind wegzulassen.

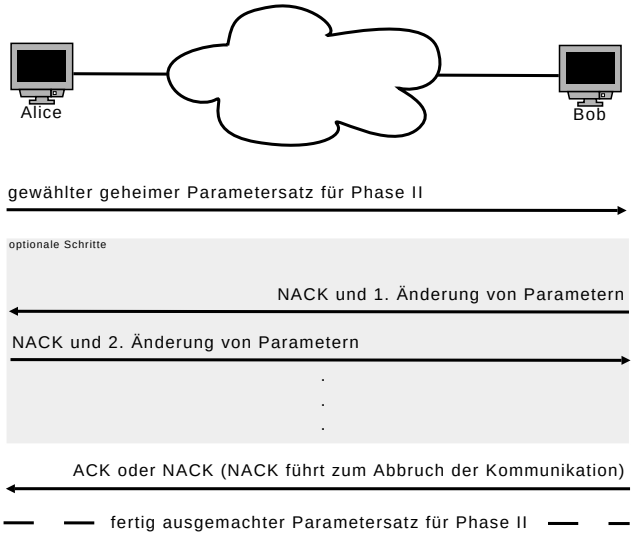

Abbildung 4.2: Ausmachen des Parametersatzes für Phase II

Umschalten auf neue kryptographische Parameter. Ein sehr wichtiger Teil von Phase I ist das übermitteln von kryptographischen Informationen (wie Algorithmen und Keys), welche für die eigentliche Kommunikation als Absicherung dienen sollen. Deshalb soll ein Basisparametersatz in Bezug auf kryptographische Methoden nur im ersten Schritt der Phase I verwendet werden. Danach, vor dem Senden von ACK oder

NACK, muss zumindest schon auf die neue Kryptographie umgeschaltet werden. Dies erhöht die Sicherheit der Kommunikation schon in Phase I (ab dem zweiten Schritt), wenn sich zumindest die kryptographischen Methoden vom gewählten Basisparametersatz unterscheiden.

4.2.1.2 Kompromittierung des öffentlichen Parametersatzes

Sollte ein Parametersatz, welcher für Phase I eingesetzt wurde, nicht sicher genug gewesen sein, dann kennt ein Angreifer im schlimmsten Fall den verschlüsselten Parametersatz für Phase II. Wenn nun davon ausgegangen wird, dass die eingesetzte Verschlüsselung hinreichend sicher ist, dann hat der Angreifer die steganographische Übertragung auf das Niveau einer kryptographischen Übertragung reduziert (jedoch nur für die erste Phase). Deshalb ist, wie zuvor schon erwähnt, die Sicherheit der Basisparametersätze zusätzlich zur Sicherheit des geheim ausgemachten Parametersatzes sehr wichtig.

Wird für die erste Protokollphase also einer der Basisparametersätze verwendet (welche öffentlich bekannt sind), hat der Man in the Middle theoretisch die Möglichkeit, dass er den ersten Payload abfängt und die Werte im Parametersatz entsprechend abändert. So könnte er eine schwächere Verschlüsselung, eine andere Protokollvariante, etc. bewirken.

Eine Einschränkung für den Man in the Middle bietet auf jeden Fall die Verschlüsselung des Payloads. Durch die stattfindende Randomisierung der Daten (siehe Abschnitt 3.2.2) muss es auch bei einem bekannten Einbettungsverfahren sehr schwer sein den Payload zu lokalisieren wenn man nicht weiß, dass einer vorhanden ist. Des weiteren besteht für beide Kommunikationspartner immer noch die Möglichkeit, nicht gewünschte Werte abzuändern (siehe Abbildung 4.2).

Solange aber keine sicheren Verfahren zur Integritätsprüfung und Authentifizierung herangezogen werden, können die Kommunikationspartner nicht 100%-ig sicher sein, dass die gesendeten Angebote im Parametersatz nicht abgeändert wurden. Deshalb kann bei besonders sicheren Setups eine Vereinbarung mit potentiellen Kommunikationspartnern stattfinden, wo die einzusetzenden Methoden ausgehandelt werden. Alle anderen als die ausgemachten Angebote können somit automatisch abgelehnt werden.

Eine andere (eventuell komplexere) Abhilfe schafft hier auch das sichere Aushandeln eines privaten Parametersatzes für Phase I.

4.2.2 Protokollphase II

In Phase II beginnt die eigentliche Übertragung der geheimen Daten. Zur Einbettung und Absicherung wird hier der zuvor in Phase I ausgemachte Parametersatz herangezogen. In dieser Phase besteht nun wiederum die Gefahr einer Man in the Middle Attacke, welche durch verschiedene Methoden abgewehrt oder zumindest eingeschränkt wer-

den kann. Für genauere Hintergründe über diese Angriffsart siehe Anhang B, für nähere Informationen zu Angriffen auf steganographische Kommunikationen siehe Abschnitt 2.8.2.

4.2.2.1 Man in the Middle in Phase II

Der Man in the Middle ist eine Entität, welche den Kommunikationskanal abhören (Angriffe ausführen) kann. Man muss auch davon ausgehen, dass die verwendete Hard- und Software performat genug ist, um ein Telefongespräch in Echtzeit abzuhören, steganographische Informationen herauszulesen, neue Informationen einzubetten und die Daten weiterzuleiten ohne dass merkliche Verzögerungen bei den Gesprächspartnern auftreten.

Aus dieser Definition lässt sich schließen, dass Angriffe von einem Man in the Middle nur dann erfolgreich verhindert werden können, wenn eine gegenseitige Überprüfung von bekannten (eindeutigen) Authentifizierungsmerkmalen der Kommunikationspartner erfolgt.

Durch ein Verhindern der unter Abschnitt 2.8.2 genannten Angriffsarten ist jedoch noch keine 100%ig unterbrechungsfreie Kommunikation gewährleistet. Der Man in the Middle kann durch Einsatz von Authentifikation zwar erkannt werden. Wenn der Man in the Middle die Kommunikation jedoch nicht nur mithören bzw. ändern, sondern einfach unterbinden möchte, müssen andere Mechanismen herangezogen werden, um die Kommunikation selbst nicht sichtbar zu gestalten (eines der Hauptziele der Steganographie, (vgl. Hübner, 2003)). Eine sehr bekannte Lösung für das Problem des versteckten Kommunizierens beschreibt der *Subliminal Channel* von G.J. Simmons. Hierbei werden innerhalb einer vom Man in the Middle „erlaubten“ Kommunikation redundante Teile verwendet, um Informationen auszudrücken, deren Bedeutung bzw. Lokalisation nur den Kommunikationspartnern bekannt sind (vgl. Simmons, 1984).

4.2.2.2 Nachträgliches Anfordern von Daten

In Abschnitt 3.4.11 wurde die Möglichkeit beschrieben, Parameter in Phase I auszutauschen, die die „Wichtigkeit der Daten“, welche in Phase II versendet werden, definieren. Nachdem der Sender diese Informationen an den Empfänger in Form des privaten Parametersatzes übertragen hat, ist für den Empfänger der Ablauf von Phase II festgelegt. Einerseits muss der Sender eventuell zusätzliche Informationen an die Rohdaten anfügen (wie Codes zur Fehlerkorrektur — siehe Abschnitt 3.6.5). Diese müssen vom Empfänger natürlich entsprechend bei Erhalt des Payloads überprüft werden. Wenn der Sender die Übertragung beendet, steht das Flag *DataEnd* zur Verfügung, über welches dieser Zustand an den Empfänger signalisiert werden kann (siehe Abschnitt 3.6.3). Abhängig vom „Wichtigkeitsparameter“ von Phase I (siehe Abschnitt 3.4.11) entsteht nun eventuell eine weitere Kommunikation:

- Müssen *alle* Daten empfangen werden, hat der Empfänger nicht erhaltene Pay-

loads (anhand der Sequenznummer ersichtlich — siehe Abschnitt 3.6.1) nachzufordern.

- Der Empfänger kann auch adaptiv selbst entscheiden, ob zu einem Zeitpunkt genügend Informationen verfügbar sind und keine weiteren Payloads nachgefordert werden müssen.

Abbildung 4.3 zeigt den zusätzlichen Ablauf nach Ende der Kommunikation, wenn Payloads vom Empfänger nachgefordert werden müssen. Auch dieser Mechanismus ist, wie ACK/NACK in Phase I (siehe Abschnitt 4.2.1.1), nur bei bidirektionaler Kommunikation möglich.

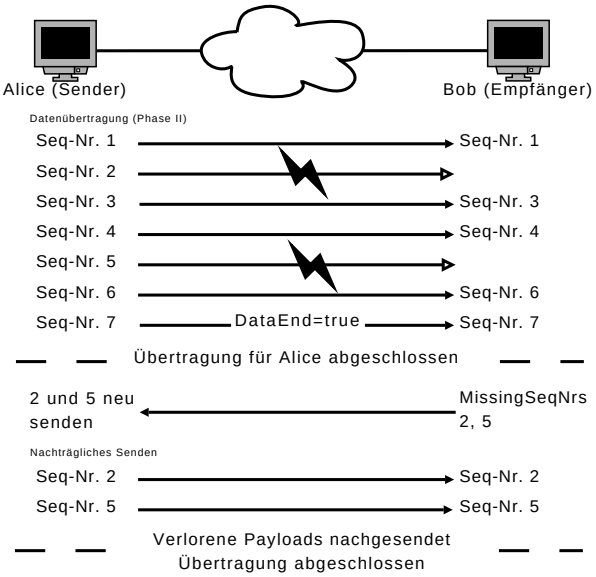

Abbildung 4.3: Nachträgliches Anfordern von nicht erhaltenen Daten in Phase II

4.3 Ausgangslage für die Protokolle

Als Basis für die Protokolle existiert die herkömmliche steganographische Kommunikation. Hierbei stellen sich alle genannten Probleme (siehe Abschnitt 2.8.3), wie fehlende Authentifikation und der Man in the Middle Angriff. Die einzige Sicherheit besteht hier im Einsatz von Sprachkommunikation (Telefonie, z.B. per VoIP) zwischen bekannten Entitäten. Dadurch wird es möglich, dass sich die Gesprächspartner aufgrund der Kenntnis der Stimme des jeweils anderen während des gesamten Gesprächs „authentifizieren“.

Für andere Varianten, wie dem Einbetten von Informationen in Bilddaten, ist die herkömmliche steganographische Kommunikation jedoch völlig unsicher. Ein Man in the Middle kann den Kommunikationskanal abhören und Nachrichten mitlesen oder sogar austauschen.

4.3.1 Existierende Konzepte zur Zusammenführung von Steganographie und Protokollen

Wie schon in Abschnitt 2.9 erwähnt wurde, dient das OSI Referenzmodell als Grundlage für netzwerkbasierte Kommunikationsprotokolle. In diesem Abschnitt werden kurz einige existierende Vorschläge für die Zusammenführung von Steganographie und Protokollen bzw. für die Verbesserung der herkömmlichen Steganographie vorgestellt.

Um den Bezug zum OSI Referenzmodell herstellen zu können, wird bei jedem Lösungsansatz versucht, die entsprechende Schicht im Modell, auf welcher der Ansatz arbeitet, anzugeben. Im nächsten Abschnitt wird dieses wichtige Modell nun kurz erläutert.

Nach dieser Einführung vorhandener Arbeiten werden die Protokolle GSP-I bis GSP-IV im Detail beschrieben.

4.3.1.1 ISO/OSI Referenzmodell

Das ISO/OSI Modell besteht aus den in Abbildung 4.4 dargestellten sieben Schichten. Beim Sender einer Kommunikationsverbindung entstehen die Daten in der obersten Schicht (*Application Layer*). Um diese an einem Empfänger zu übertragen, müssen die Daten bis zum *Physical Layer* durchgereicht werden. Jede Schicht fügt im Verlauf Headerinformationen hinzu und kommuniziert somit „virtuell“ mit der korrespondierenden Schicht des Empfängers (Headerinformationen werden nur in jener Schicht betrachtet, wo sie am anderen System eingefügt wurden). Beim Empfänger findet der umgekehrte Prozess statt und die Daten werden von Schicht eins zu Schicht sieben hochgereicht, wo sie schlussendlich von der empfangenden Applikation (z.B. Webbrowser) verarbeitet werden (vgl. ISO/IEC, 1996).

Beispiele für bekannte Protokolle sind Ethernet auf Schicht zwei, Internet Protocol (IP) auf Schicht drei, TCP und UDP auf Schicht vier. Eine exakte Trennung zwischen den Schichten fünf, sechs und sieben ist nicht immer sinnvoll bzw. möglich. Ein Beispiel für ein Protokoll dieser Schichten ist das HTTP.

Einordnung von GSP-I bis GSP-IV. Die in diesem Dokument vorgestellten Protokolle (Abschnitte 4.4 bis 4.7) arbeiten alle ab „Schicht acht“ des OSI Referenzmodells. D.h. die Protokollabläufe (wie Verschlüsselung, Einbettung, etc.) finden statt, bevor die Applikation (wie e-Mail oder VoIP Clients) die zu versendenden Daten übergeben bekommt und durch die sieben Schichten an das Übertragungsmedium weiterleiten kann. Beim Empfänger findet dieser Vorgang danach genau in der umgekehrten Reihenfolge statt.

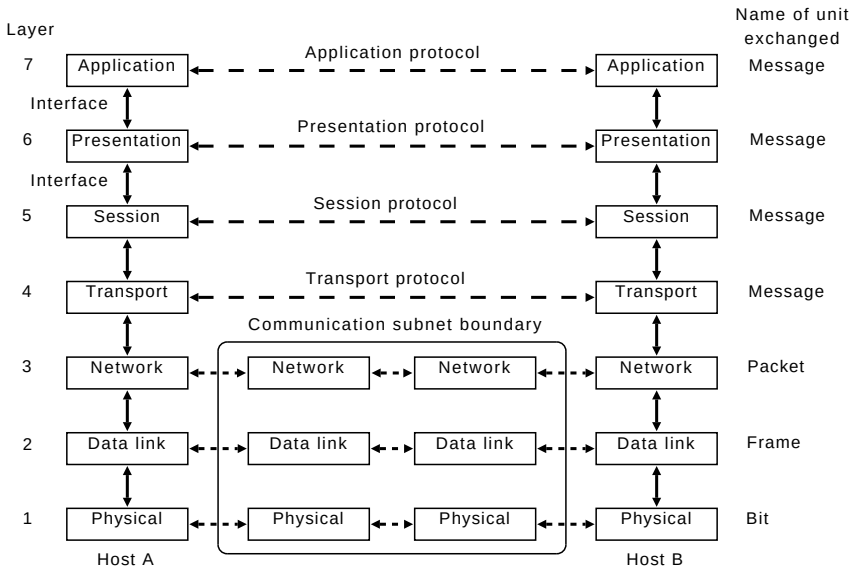

Abbildung 4.4: ISO/OSI Referenzmodell (vgl. Glover und Grant, 2003, S. 696)

4.3.1.2 Protocol Steganography

In bestehender Literatur wird die Zusammenarbeit von Steganographie und Protokollen meist als *Protocol Steganography* in folgender (oder ähnlicher) Form beschrieben:

> „*Protocol Steganography* ist the art of embedding information within messages and network control protocols used by common applications.“ (Lucena u. a., 2004)

Daraus folgt, dass steganographische Nachrichten in existierende Kommunikationsprotokolle versteckt übertragen werden sollen. Ein möglicher Protokollablauf ist hierbei jedoch nur durch das verwendete Cover (ein Protokoll wie z.B. Internet Control Message Protocol (ICMP)) gegeben und hat nichts mit einem eigens für die steganographische Kommunikation entwickelten Protokoll zu tun. Die existierenden Ansätze arbeiten daher eher auf der Ebene von Einbettungsverfahren (siehe Abschnitt 2.7) und können somit alternativ zu Echo Hiding oder LSB Einbettung eingesetzt werden. Ein Vorteil gegenüber der „einfachen“ Einbettung und dem Austausch von Bildern, Audiodaten, etc. wird in der Verwendung von TCP als Transportprotokoll gesehen, welches Mechanismen bereitstellt, um z.B. verloren gegangene Pakete nachzusenden. Es wird jedoch auch hervorgehoben, dass beim Einsatz von UDP zusätzliche Mechanismen implementiert werden müssen, um eine ähnliche Garantie für die Übertragung zu gewährleisten (vgl. Lucena u. a., 2004).

Einzuordnen sind diese Verfahren auf den Schichten zwei bis vier (eventuell auch sieben) des OSI Referenzmodells, je nachdem, welches Protokoll zur Einbettung des Payloads herangezogen wird.

IEEE 802.11 Header und WLAN Kommunikation – Schicht zwei. Bei drahtloser Kommunikation liegt es Nahe, auch die in diesem Medium verwendeten Header zu betrachten und auf ihre steganographische Anwendbarkeit hin zu überprüfen. Beispiele für diese Verfahren sind das HICCUPS System (vgl. Szczypiorsky, 2003), sowie die Empfehlungen aus (Kühne, 2007).

TCP/IP – Schicht drei und vier. In mehreren Ausarbeitungen wurde der TCP/IP Header herangezogen, um Daten in unbenutzten oder nicht vollständig benutzten Headerfeldern zu verstecken (vgl. Kühne, 2007). In diesen existierenden Konzepten werden die Felder des Headers analysiert und deren Relevanz für die Aufnahme von steganographischen Payloads beurteilt. D. Kundur und K. Ahsan beschreiben zumindest den zusätzlichen Einsatz einer symmetrischen Verschlüsselung, was in Richtung eines einfachen Protokollablaufs führt (vgl. Kundur und Ahsan, 2003, Murdoch und Lewis, 2005, Rowland, 1997).

Application Layer Protokolle – Schicht sieben. Auch Protokolle der obersten Schicht des OSI Modells wurden betrachtet, um deren Anwendbarkeit als Träger für steganographische Daten zu bewerten. Hierbei wurde z.B. das Secure Shell (SSH) Protokoll in einer Beispielimplementierung herangezogen. Hierbei wird der eigentliche Inhalt verschlüsselt übertragen, daher ändern sich die statistischen Eigenschaften der Protokolldaten nicht, wenn zusätzlich steganographische Daten eingebettet werden (welche im Idealfall ebenfalls verschlüsselt wurden). Mögliche Punkte zur Einbettung bei SSH sind das Message Authentication Code (MAC) Feld im SSH Paket und das hinzufügen von zusätzlichem Inhalt bei Paketen, welche von bestehenden SSH Verbindungen gesendet werden (vgl. Lucena u. a., 2004).

4.3.1.3 Triple Layer Data Security

Von S. Mukherjee u. a. wurde ein Verfahren zur Absicherung von steganographischer Kommunikation mit dem Namen *Triple Layer Data Security* beschrieben. Hierbei wird der Kommunikationskanal zusätzlich einerseits durch Algorithmen der Kryptographie und andererseits durch ein neu vorgestelltes Verfahren zur LSB Einbettung erweitert (vgl. Mukherjee u. a., 2008).

Diese Arbeit geht in Richtung der hier vorgestellten Protokolle. Es werden die Daten in mehreren Schritten abgesichert, bevor diese über einen Kanal (ein Netzwerk) übertragen werden. Somit kann ebenfalls von einem Protokollablauf gesprochen werden, welcher über die einfache Anwendung der steganographischen Verfahren hinausgeht. Im Folgenden wird jedoch zu sehen sein, dass die hier definierten Protokollabläufe noch umfangreichere Maßnahmen in Richtung einheitlicher Abläufe bieten als dieses Verfahren.

Die Einordnung dieses Ansatzes in das OSI Referenzmodell findet ab Schicht fünf statt, da die Daten vor einer Verarbeitung durch die zu übertragenden Applikationen

und Medien bereits modifiziert werden.

4.4 Protokoll GSP-I

Das erste Protokoll setzt keine Publikation von öffentlichen Schlüsseln der Kommunikationspartner voraus. Um die erste Phase dennoch sicher (verschlüsselt) ablaufen zu lassen, wird hier das *Massey-Omura Kryptosystem* angewandt. Die einzige Voraussetzung hierbei ist es, dass sich beide Kommunikationspartner im Vorfeld auf eine Primzahl geeinigt haben. Diese Primzahl muss jedoch nicht geheim gehalten werden, da das Verfahren auf dem Problem des *diskreten Logarithmus* beruht (wie auch das *Diffie-Hellman Verfahren*, welches diesem Verfahren ähnlich ist) (vgl. Koblitz, 2004, S. 100 f).

Das Protokoll läuft nun folgendermaßen ab (die beiden Kommunikationsteilnehmer seien *Alice* und *Bob*, die gemeinsame Primzahl sei p, der von Alice gewählte Parametersatz sei m):

1. Alice erzeugt eine geheime Zahl $a \mod (p-1)$ und die Inverse $a' \mod (p-1)$. Es gilt
$$a * a' \mod (p-1) = 1.$$

2. Bob erzeugt ebenfalls eine geheime Zahl, $b \mod (p-1)$ und die entsprechende Inverse $b' \mod (p-1)$. Der erwähnte Zusammenhang muss auch hier gelten.

3. Alice berechnet nun
$$x = m^a \mod p$$
und sendet x an Bob.

4. Bob kann nun das empfangene x potenzieren und dadurch y berechnen:
$$y = x^b \mod p.$$
Dieses Ergebnis wird wiederum an Alice geschickt.

5. Alice erhält nun y und berechnet z durch
$$z = y^{a'} \mod p.$$
Durch diese Berechnung „entfernt“ Alice die erste Verschlüsselung wieder (siehe Punkt drei — da a' die Inverse von a ist). z wird an Bob gesandt.

6. Mit z kann Bob nun die ursprüngliche Nachricht berechnen:
$$m = z^{b'} \mod p.$$
Diese Operation entfernt die Verschlüsselung, welche Bob in Punkt vier hinzugefügt hat.

Nach diesen Berechnungen haben Alice und Bob somit einen Parametersatz sicher ausgetauscht ohne zuvor kryptographische Schlüssel ausgemacht oder öffentlich zur Verfügung gestellt zu haben. Des weiteren wurde keine Information unverschlüsselt übertragen, da zumindest immer eine „Schicht der Verschlüsselung“ (repräsentiert durch das Potenzieren mit den geheimen Werte) am Kanal vorhanden war. Für eine genaue Beschreibung dieses Verfahrens siehe (Koblitz, 2004, Schneier, 1995). Abbildung 4.5 stellt diesen Ablauf in Phase I graphisch dar.

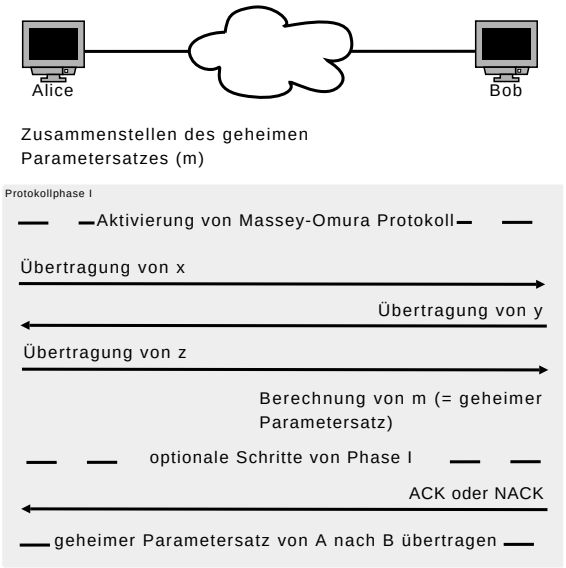

Abbildung 4.5: Ablauf von Phase I bei Protokoll GSP-I

Aus Abbildung 4.5 ist ersichtlich, dass bei dieser Protokollvariante (zumindest) drei Nachrichten in Protokollphase I versendet werden. Diese Payloads (x, y und z) werden gemäß der Beschreibungen unter Abschnitt 4.2.1 mit einem öffentlich verfügbaren Parametersatz eingebettet.

4.4.1 Authentifizierung

Um bei dieser Protokollvariante eine Authentifizierung zu erreichen, müssen beim Parametersatz die optionalen Felder unter *Biometry* herangezogen werden. Hierbei sind biometrische Merkmale sowie die Angabe eines Passworts möglich (siehe Anhang A.1 und Abschnitt 3.4.9).

Um diese Formen der Authentifizierung gewährleisten zu können ist es natürlich eine Voraussetzung, dass die Kommunikationspartner vorab über die verwendeten Merkmale der Biometrie des jeweils anderen (Fingerabdruck, Iris-Scan, Gesicht, etc.) oder über ein gemeinsames Passwort Bescheid wissen.

4.4.2 Man in the Middle Abwehr

Beim Einsatz einer Authentifizierung ist diese Protokollvariante vor Man in the Middle Angriffen insofern geschützt, dass die Kommunikationspartner merken würden, wenn ein Man in the Middle übertragene (steganographische) Inhalte verändern würde. Nimmt man jedoch nur biometrische Merkmale oder das Passwort zur Authentifzierung, hat man im Payload einen statischen Wert eingetragen. Dies ermöglicht dem Man in the Middle das erneute Verschicken eines Payloads nachdem dieser schon zwischen den Kommunikationspartnern ausgetauscht wurde. Der Man in the Middle kann den Inhalt zwar dennoch nicht lesen (aufgrund der Verschlüsselung) oder verändern, jedoch kann es je nach Anwendung ebenfalls zu Problemen führen, wenn Daten mehrfach beim Empfänger ankommen. Diese Angriffsart wird *Replay Attack* genannt.

Die genannten Probleme und Angriffe können zumindest durch zwei Maßnahmen vermindert bzw. unterbunden werden:

- Der Einsatz von *Nonces* im Zuge der Authentifizierung schützt vor einem Replay Attack.
- Beim Einsatz von biometrischen Merkmalen können diese in Hashwerte oder andere Daten (z.B. Einbettungsverfahren) einfließen und so gewährleisten, dass der Empfänger eine Änderung dieser Angaben bemerkt. Somit kann ein Angreifer nicht unbemerkt Authentifizierungsdaten austauschen und somit den Empfänger täuschen.

Wird keine Authentifizierung angewandt, ist diese Protokollvariante nicht geschützt vor Man in the Middle Angriffen. Ein Angreifer kann dann bereits zu Beginn das Massey-Omura Protokoll mit beiden Kommunikationspartnern ausführen und keiner der beiden kann feststellen, ob er nun mit dem Gegenüber oder einem Man in the Middle kommuniziert. Es ist in dieser Variante also von den Benutzern bzw. der Implementierung abhängig, wie sicher gegenüber Angriffen das Protokoll arbeitet.

4.5 Protokoll GSP-II

Diese Protokollvariante setzt voraus, dass beide Kommunikationspartner ein Schlüsselpaar zur Verwendung mit Algorithmen der Public-Key Kryptographie erstellt haben. Des weiteren muss jeder Beteiligte seinen öffentlichen Schlüssel bekannt geben (etwa in Form eines Zertifikats), damit eine Kommunikation initiiert werden kann.

Der Sender kann nun einen geheimen Parametersatz für Phase II erstellen und diesen vor dem Einbetten mit dem öffentlichen Schlüssel des gewünschten Empfängers verschlüsseln. Somit läuft die erste Protokollphase ähnlich dem OpenPGP Protokoll ab. Dies stellt eine Beschleunigung der ersten Protokollphase gegenüber GSP-I dar, da im Idealfall (der Empfänger akzeptiert den gewählten Parametersatz sofort) nur zwei

Payloads (inklusive ACK bei bidirektionaler Kommunikation) ausgetauscht werden müssen. Abbildung 4.6 stellt die erste Phase mit GSP-II graphisch dar.

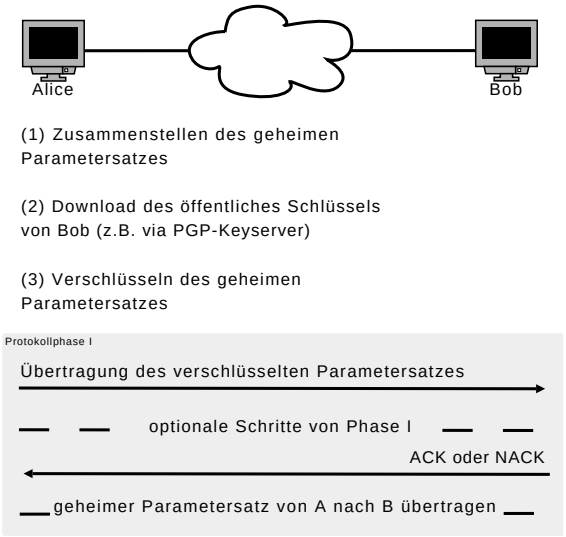

Abbildung 4.6: Ablauf von Phase I bei Protokoll GSP-II

4.5.1 Authentifizierung und Man in the Middle Abwehr

Beim Einsatz von Public-Key Kryptographie ist es naheliegend, dass digitale Signaturen zur Authentifizierung der übertragenen Informationen herangezogen werden. Hierfür ist das Feld *Signature* im Nachrichtenformat (siehe Anhang A.1) verwendbar, wo die berechnete Signatur über die Daten entsprechend eingetragen werden kann. Dadurch wird gewährleistet, dass der Kommunikationspartner weiß, von wem die gesendeten Daten stammen. Ein Man in the Middle kann hierbei keine Änderungen vornehmen, da sonst die Überprüfung der digitale Signatur beim Empfänger einen Fehler ergeben würde – der Man in the Middle wäre somit aufgedeckt.

Es ist jedoch zu beachten, dass die eingesetzten Schlüssel für die Public-Key Kryptographie zwar eventuell von einem Zertifikat stammen, der Einsatz einer PKI für diese Protokollvariante jedoch nicht zwingend erforderlich ist (siehe Abschnitt 4.7). Dadurch kann der Man in the Middle theoretisch wie bei GSP-I ohne Authentifzierung (siehe Abschnitt 4.4.2) mit beiden Kommunikationspartnern kommunizieren, ohne dass diese merken würden, nicht mit dem jeweils anderen direkt verbunden zu sein (außer bei speziellen Szenarien wie einem Telefongespräch, wenn die Kommunikationspartner sich anhand ihrer Stimmen kennen und auf diese Weise authentifizieren können). Abbildung 4.7 zeigt das hier genannte Man in the Middle Problem und den Ablauf.

Wie die Abbildung zeigt ist es für den Man in the Middle relativ einfach, sich gegenüber Alice als Bob und gegenüber Bob als Alice auszugeben, so lange keine PKI und vertrauenswürdigen Zertifikate eingesetzt werden. Schon beim Besorgen des öf-

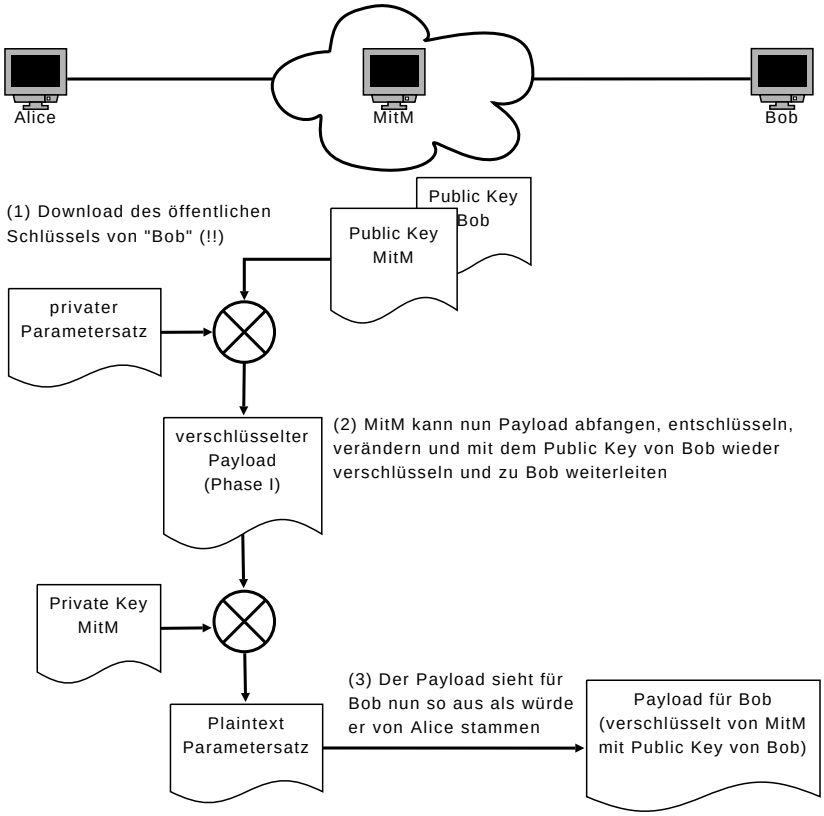

Abbildung 4.7: Mögliches Man in the Middle Problem bei GSP-II, dargestellt aus Sicht von Alice

fentlichen Schlüssels von Bob wird Alice mit dem Schlüssel des Man in the Middle getäuscht. Die weitere Kommunikation ist für den Man in the Middle somit eindeutig lesbar, es ist für ihn sogar möglich, Änderungen am Parametersatz von Alice zu bewirken.

4.6 Protokoll GSP-III

Diese Protokollvariante ist vor allem in Phase I komplexer, da zusätzliche Authentifizierung der Kommunikationspartner über einen eigenen Mechanismus gewährleistet wird. Hierbei sind einige Überlegungen notwendig um den Man in the Middle Angriff auszuschließen. Wie schon unter Abschnitt 4.2.2.1 beschrieben wurde muss davon ausgegangen werden, dass dem Man in the Middle praktisch alle Möglichkeiten zur Verfügung stehen, um die Kommunikation ohne Unterbrechungen (Störungen) abzuhören und auch Daten abzuändern. Der Man in the Middle kann somit immer Veränderungen vornehmen so lange man keine Informationen verwendet, welche zuvor schon vereinbart wurden. Nur später folgende Informationen kann ein Man in the Middle nicht „vorausahnen“.

Diese Protokollvariante verwendet daher das Feld *Hash* des Nachrichtenformats, wo ein Hashwert über Daten der zweiten Protokollphase (der eigentlich zu übertragenden Daten) inkludiert wird. Da somit im Vorfeld (in Phase I) durch den Hashwert Bezug auf „Daten der Zukunft“ (aus Phase II) genommen wird, kann ein Man in the Middle Angriff erfolgreich ausgeschlossen werden. Abbildung 4.8 zeigt den Ablauf von Phase I bei Anwendung dieser Protokollvariante.

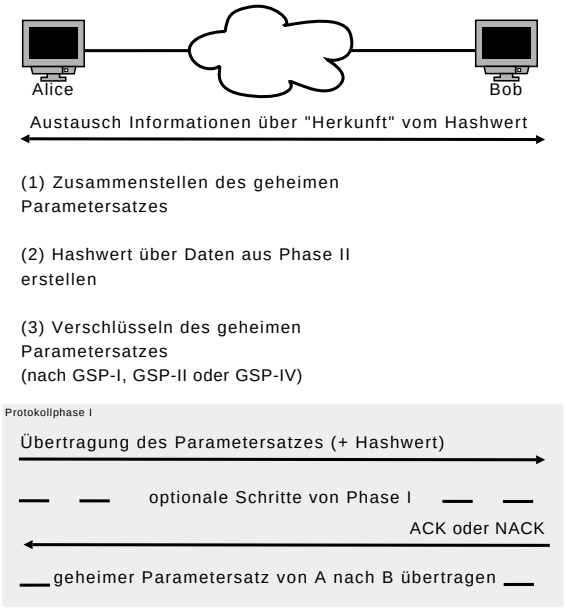

Abbildung 4.8: Ablauf von Phase I bei Protokoll GSP-III

Um den Payload zu Beginn verschlüsseln zu können, kann auf den Ablauf von GSP-I, GSP-II oder GSP-IV zurückgegriffen werden.

4.6.1 Einschränkungen

Das offensichtlichste Problem dieser Protokollvariante ist die Tatsache, dass die Daten, welche zum Hashwert von Phase I führen, beiden Kommunikationspartnern bekannt sein müssen. Der Initiator der Kommunikation muss dem gewünschten Empfänger die Daten oder zumindest eine Information darüber, welche Daten herangezogen werden, im Vorfeld mitteilen, damit dieser den Hashwert später verifizieren kann. Kann der Man in the Middle diesen Austausch abfangen, ist er wiederum in der Lage, den Kommunikationskanal ohne Wissen der Kommunikationspartner zu kompromittieren.

Eine (relativ) sichere Anwendung für diese Protokollvariante ist eine Echtzeitanwendung, z.B. ein Telefongespräch über VoIP, wenn ein Teil des Trägermedium für den Hashwert herangezogen wird (siehe Abschnitt 4.6.2). Für Nicht-Echtzeit Anwendungen wäre eine Man in the Middle Attacke einfacher denkbar, weshalb hier eigene Varianten notwendig sind, um eine sichere Kommunikation gewährleisten zu können (siehe Abschnitt 4.6.3).

4.6.2 Ablauf bei einem Telefongespräch

Ein Gesprächspartner (der Initiator der Verbindung) legt die Quelle für den Hashwert von Phase I fest, bevor das Gespräch begonnen wird. Danach erfolgt die Kommunikation, in welcher steganographische Daten in den Kanal eingebettet werden (Phase II). Zu einem vordefinierten Zeitpunkt (welcher auch den Systemen bekannt sein muss) werden nun die Daten übertragen, welche im Vorfeld schon zum Hashwert führten. Der Gesprächspartner (bzw. das System) muss natürlich vorbereitet sein und wissen, welche Informationen herangezogen wurden.

Somit kann ein Man in the Middle (welcher den Inhalt ändern muss) durch dieses Verfahren erkannt werden. Sobald das System des Gesprächspartners die zugrundeliegenden Informationen erkennt und diese nicht mit den Erwartungen übereinstimmen, kann z.B. die Kommunikation abgebrochen werden. Abbildung 4.9 stellt den beschriebenen Ablauf graphisch dar.

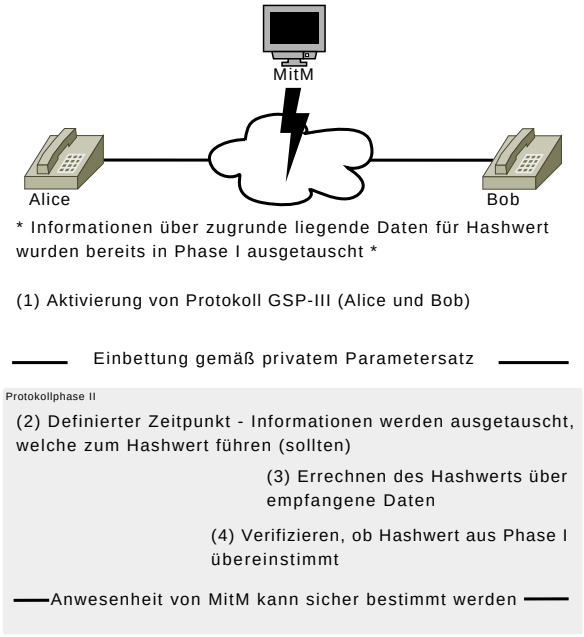

Abbildung 4.9: Ablauf von Phase II bei Protokoll GSP-III und einem Telefongespräch

4.6.3 Ablauf bei e-Mail Verkehr mit Bildern im Anhang

Um bei einer e-Mail Kommunikation mit dieser Protokollvariante eine Abwehr bzw. Erkennung von Man in the Middle Angriffen zu erreichen, sind zusätzliche Aufwände nötig. Die Kommunikationspartner müssen sich innerhalb der Kommunikation auf ein bestimmtes Motiv einigen. Hierbei ist es wichtig, dass der Man in the Middle den Inhalt der e-Mail nicht zu seinen Gunsten verändern kann und dass auch für den Empfänger klar ist, welches Motiv ausgemacht wurde (siehe Abschnitt 4.6.1). In Phase I wird somit der Hashwert des ausgemachten Bildes integriert. Dies gewährleistet, dass

der Man in the Middle diese Information (Parametersatz) nicht verändern kann, ohne dass der Empfänger es (später) bemerkt. Außerdem kann zur zusätzlichen Sicherheit im Text der e-Mail auch eine Information über das im dritten Schritt (zweites e-Mail des Initiators) gesendete Bild erfolgen.

Da man davon ausgehen sollte, dass der Man in the Middle „alle Bilder“ in relativ kurzer Zeit beschaffen und verwenden kann, muss man mit dieser Variante arbeiten um Authentifizierung zu gewährleisten. Dies könnte man umgehen, indem sich die Kommunikationspartner z.B. persönlich kennen und auf ein Bild einigen, welches der Man in the Middle *sicher nicht* beschaffen und verwenden kann (z.B. ein persönliches Foto des Senders in seiner privaten Umgebung). Abbildung 4.10 zeigt den Ablauf dieser Protokollvariante.

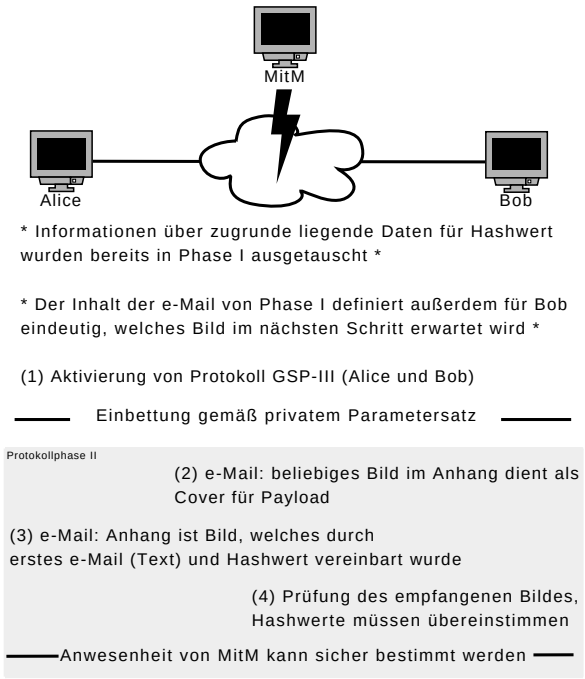

Abbildung 4.10: Ablauf von Phase II bei Protokoll GSP-III und e-Mail Verkehr

4.6.4 Authentifizierung und Man in the Middle Abwehr

In dieser Protokollvariante erfolgt die Authentifizierung über einen Hashwert, welcher in Phase I eingebracht wird, jedoch bereits Daten von Phase II beschreibt (z.B. Bilder, Trägerinformationen, etc.). Da nur der „echte“ Sender die richtigen Informationen in Phase II verfügbar hat und verwendet, ist für den Empfänger dessen Authentizität bestimmbar.

Ein Problem von GSP-III ist es, dass der Empfänger darüber informiert sein muss, welche Daten zum Hashwert führen. Dies ist natürlich notwendig, da er den Hashwert in Phase II mit den entsprechenden Daten nach dem Empfang überprüfen muss. Sofern diese Informationen über einen vertraulichen (und/oder anderen) Kanal ausgetauscht

wurden (z.B. per SMS, Messenger, Telefon, etc.), kann von einer sicheren Protokollvariante gesprochen werden. Auch Man in the Middle Angriffe können somit aufgedeckt werden. Bei diesen Betrachtungen wird davon ausgegangen, dass die dem Hashwert zugrunde liegenden Informationen nicht von einem Man in the Middle „zufällig“ erraten werden können. Dies würde die Integrität des Protokolls wiederum brechen.

Wird jedoch kein Hashwert verwendet (was aber einen Bruch mit der Definition von GSP-III bedeuten würde), kann auch keine Authentifizierung stattfinden und der Man in the Middle hat die Möglichkeit, Daten nach belieben zu ändern (das einzige Hindernis bleibt die Verschlüsselung des Payloads).

4.7 Protokoll GSP-IV

Die letzte Protokollvariante ermöglicht eine Kombination von steganographischer Kommunikation und PKI. Der Initiator der Kommunikation fragt wie bei GSP-II den öffentlichen Schlüssel des gewünschten Empfängers ab. Jedoch ist in dieser Variante durch die PKI gewährleistet, dass der öffentliche Schlüssel wirklich zu der Entität gehört, welche der Sender erwartet. Diese Verbesserung gegenüber GSP-II ermöglicht der Einsatz von vertrauenswürdigen Instanzen, welche auf den öffentlichen Schlüsseln der Kommunikationspartner Zertifikate erstellen und signieren. Dies verhindert die Probleme von GSP-II und fügt außerdem noch eine Möglichkeit zur sicheren Authentifikation hinzu. Das Feld *Signature* des Nachrichtenformats kann dazu verwendet werden, eine digitale Signatur über die gesendeten Daten hinzuzufügen. Dies gewährleistet dem Empfänger die Integrität und Authentizität der Daten. Abbildung 4.11 zeigt die erste Protokollphase mit GSP-IV.

Wie in der Abbildung dargestellt ist, kommuniziert Alice zusätzlich mit einer vertrauenswürdigen Instanz, welche Zertifikate ausstellt und signiert. Von dort wird der öffentliche Schlüssel von Bob bezogen. Diese Variante ist somit komplexer als GSP-II, da die Interaktion mit der PKI eingebaut werden muss. Des weiteren ist auch zu beachten, dass alle gewünschten Kommunikationspartner ihre kryptographischen Schlüssel von einer Zertifizierungsstelle signieren lassen müssen (Erstellung eines Zertifikats). Dieses Vorgehen ist eventuell nicht in allen Situationen möglich.

Zusätzlich zur erhöhten Komplexität kommt noch die digitale Signatur zum Payload hinzu, was auch die Kapazität des Kanals für eigentliche Informationen verringert (was jedoch bei GSP-II ebenfalls der Fall ist). Wird ein Kommunikationskanal mit geringer Kapazität verwendet, kann dieser Faktor mitunter negative Auswirkungen haben.

4.7.1 Authentifizierung und Man in the Middle Abwehr

Durch die Möglichkeit der Verwendung von digitalen Signaturen bietet diese Protokollvariante eine sichere Authentifikation der Kommunikationspartner. Dies wiederum gewährleistet eine effektive Abwehr gegen Man in the Middle Angriffe, da eine Ma-

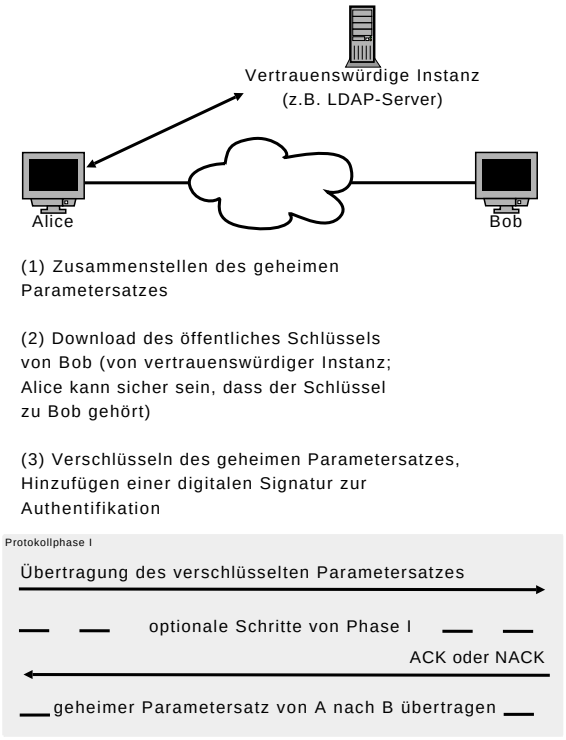

Abbildung 4.11: Ablauf von Phase I bei Protokoll GSP-IV

nipulation der Daten durch den Empfänger erkannt werden würde. Ein Mitlesen von gesendeten Daten wird ebenso durch Verschlüsselung des Payloads bzw. geeigneten steganographischen Verfahren im Vorfeld verhindert.

Ein Vorteil dieser Variante ist die Tatsache, dass man hier nicht die Probleme von GSP-II bekommt (siehe Abschnitt 4.5.1). Lädt sich ein Kommunikationspartner ein Zertifikat mit dem öffentlichen Schlüssel des Empfängers herunter, kann er sicher sein, dass der Schlüssel auch wirklich zu der Person gehört. Ein Man in the Middle kann sich somit nicht mehr mit selbstsignierten und gefälschten Zertifikaten in die Kommunikation einschleusen. Dies setzt natürlich den durchgängigen Einsatz von vertrauenswürdigen Instanzen, Zertifikaten und einer PKI voraus.

4.8 Vergleich der vorgestellten Protokolle

Nachdem nun alle Protokollvarianten definiert und deren Funktionsweise beschrieben wurde zeigt Tabelle 4.1 eine Zusammenfassung der Fähigkeiten der dargestellten Protokolle. „○“ bedeutet, dass das Feature mit Einschränkungen verfügbar ist (siehe Beschreibungen der Protokolle), „●“ bedeutet, dass diese Protokollvariante das Feature unterstützt.

PROTKOLL-VARIANTE	AUTHENTIFIZIERUNG VERFÜGBAR	MAN IN THE MIDDLE ABWEHR
GSP-I	○	○
GSP-II	●	○
GSP-III	●	○
GSP-IV	●	●

Tabelle 4.1: Authentifizierung und Man in the Middle Abwehr bei den Protokollen

4.9 Beispiele für BER-Kodierung eines Parametersatzes

Dieser Abschnitt gibt zwei Beispiele für die Kodierung von Parametersätzen laut BER. Die Werte der Tags wurden bereits in Abschnitt 3.4.14 definiert.

4.9.1 Parametersatz für GSP-II

```
0xe1 0x35                                                    -- GSP
    0x04 0x01 0x01                                           -- Version
    0x04 0x01 0x02                                           -- Protocol
    0xe2 0x03                                                -- Steganography
        0xc3 0x01 0x01                                       -- Algorithm
    0xe4 0x03                                                -- Cryptography
        0xc5 0x01 0x01                                       -- Algorithm
    0xeb 0x1d                                                -- Signature
        0xcc 0x01 0x01                                       -- Algorithm
        0x04 0x18 0x0bbbcbc2ceefcaa2d1893626831ff8a205b69cfdd494a9d3
                                                             -- SignatureValue
    0x04 0x04 /* CRC Code */                                 -- CRC
```

Listing 4.1: BER-Kodierung eines Parametersatzes für GSP-II

4.9.2 Parametersatz für GSP-III

```
0xe1 0x51                                                    -- GSP
    0x04 0x01 0x01                                           -- Version
    0x04 0x01 0x03                                           -- Protocol
    0xe2 0x03                                                -- Steganography
        0xc3 0x01 0x01                                       -- Algorithm
    0xe4 0x18                                                -- Cryptography
        0xc5 0x01 0x04                                       -- Algorithm
        0x02 0x01 0x80                                       -- KeyLength
        0x04 0x10 0xed2e308983f48d7066ffd57f9560da17         -- Key
    0xe8 0x1c                                                -- Hash
        0xc9 0x01 0x01                                       -- Algorithm
        0x03 0x10 0xc1144a106b004a3528bb2e7d368f3127         -- FutureHash
        0x04 0x05 0x2098de9992                               -- HashedData
    0xed 0x06                                                -- Importance
        0xce 0x01 0x01                                       -- Mode
        0x04 0x01 0x03                                       -- Parameter
    0x04 0x04 /* CRC Code */                                 -- CRC
```

Listing 4.2: BER-Kodierung eines Parametersatzes für GSP-III

Kapitel 5

Fazit und Ausblick

Ziel dieser Arbeit war die Definition von Protokollen für steganographische Kommunikation. Wie in Kapitel 2 beschrieben wurde, liegt der Vorteil eines anerkannten und standardisierten Protokollablaufs unter anderem in der Tatsache, dass dies eine bessere (bzw. effektive) Absicherung gegen spezielle Angriffsarten bietet.

5.1 Beantwortung der Forschungsfragen

In Kapitel 1 wurden zwei Forschungsfragen definiert, welche im Verlauf dieser Arbeit betrachtet und beantwortet wurden.

5.1.1 Probleme der herkömmlichen Steganographie

Die erste Frage bezieht sich auf herkömmliche Steganographie, also jener Kommunikationsform, welche ohne definierte Protokollabläufe auskommt. Die Kommunikationspartner sind hierbei nur durch die Protokolle geschützt, in deren Datenstrom die steganographischen Daten eingebettet werden. Diese Protokolle haben aber nicht zwangsweise die Voraussetzung, auch sichere Känale zu verwenden. Beispielsweise TCP und UDP, welche häufig im Local Area Network (LAN) und für Internetverbindungen eingesetzt werden, sind in ihren Standardausführungen nicht für sichere Übertragungen geeignet (siehe Abschnitt 4.3.1.2). Die Anwender haben also, wie auch in Kapitel 2 beschrieben wurde, mehrere Probleme:

- Auf eine herkömmliche steganographische Kommunikation bestehen mehrere Arten des Angriffs (siehe Abschnitt 2.8.2).
- Ohne zusätzliche „Features“ des Kommunikationskanals ist dieser bei herkömmlicher steganographischer Kommunikation das schwächste Glied.

- Dies ermöglicht den Man in the Middle Angriff, eine gefährliche Angriffsart, in welcher ein Dritter den Kommunikationskanal abhören und auch in die Kommunikation einschreiten kann. Bietet das verwendete Protokoll keine geeigneten Mechanismen, ist es auch möglich, Daten zu verändern ohne dass dies die Kommunikationspartner bemerken.

In Abschnitt 2.8.3 wurde auf eine Abhilfe gegen dieses Man in the Middle Problem eingegangen, nämlich der Authentifizierung der Kommunikationspartner. Hierbei wird gleichzeitig ein weiteres Problem der herkömmlichen Steganographie aufgezeigt, da hierfür keine Mechanismen (aus Mangel an Protokollen) allgemein verfügbar sind.

5.1.2 Abwehr von Man in the Middle Angriffen durch geeignete Protokolle

Die zweite Forschungsfrage bezieht sich auf das eigentliche Thema dieser Arbeit, dem Design von Protokollen, welche eine steganographische Kommunikation gegenüber Angriffen, wie dem Man in the Middle, absichern können. Kapitel 3 legt die Grundlagen für die Zusammenführung von Steganographie und Protokollen dar. Neben einer Analyse des Kommunikationskanals und dessen Anforderungen in Bezug auf steganographische Kommunikation beinhaltet dieses Kapitel auch eine Beschreibung des Nachrichtenformats, welches für den Datenaustausch mittels den in Kapitel 4 beschriebenen Protokollen (GSP-I bis GSP-IV) eingesetzt wird.

Der Ablauf der Kommunikation beinhaltet zwei Phasen. In Phase I werden die Sicherheits- und Kommunikationsparameter für Phase II zwischen den Kommunikationspartnern ausgemacht. Phase II wird verwendet, um die eigentlichen, geheimen, Daten zu übertragen. Für Phase I und zur Definition der Sicherheitsparameter, wie Authentifzierung und Verschlüsselungsmethoden, wurden in Kapitel 4 vier verschiedene Protokolle vorgestellt und definiert:

- **GSP-I.** *Protokoll ohne Schlüsselveröffentlichung*
  Mittels des Massey-Omura Kryptosystems werden die Parameter in Phase I ausgetauscht. Die Kommunikationspartner müssen hierfür im Vorfeld keine Schlüsselinformationen zur Verfügung stellen. Dafür benötigt diese Protokollvariante drei Schritte „innerhalb des ersten Schritts“ von Phase I, was einen Overhead darstellt (siehe Abschnitt 4.4).

- **GSP-II.** *Protokoll mit öffentlichem Schlüssel, aber ohne PKI*
  Die Kommunikationspartner müssen im Vorfeld einen öffentlichen Schlüssel austauschen, um die Daten von Phase I mittels dieses Schlüssels kryptographisch abzusichern. Hierbei wird keine PKI verwendet, weshalb eine Man in the Middle Gefahr besteht (siehe Abschnitt 4.5).

- **GSP-III.** *Authentifizierung durch „Zukunftsinformationen“*
  In Phase I werden Daten (Hashwerte) übertragen, welche über andere Daten

berechnet wurden, die jedoch erst in Phase II ausgetauscht werden. Dies ermöglicht die Abwehr des Man in the Middle, da ein Angreifer nicht über Informationen Bescheid wissen kann, welche noch nicht übertragen wurden (siehe Abschnitt 4.6).

- **GSP-IV.** *Verbesserung von GSP-II durch Einsatz einer PKI*
  Durch den Einsatz einer PKI mit vertrauenswürdigen Instanzen und signierten Zertifikaten kann das Man in the Middle Problem von GSP-II in dieser Protokollvariante behoben werden (siehe Abschnitt 4.7).

5.2 Weiterführende Arbeiten

Diese Arbeit beschreibt Probleme der Steganographie und sucht die Lösungen in definierten Protokollen. Diese Protokollabläufe wurden beschrieben und auch das Format der gesendeten Nachrichten wurde exakt definiert (siehe Anhang A).

Ein längerfristiges Ziel dieser Definitionen ist die Anerkennung der Protokolle als einheitliche Varianten für steganographische Kommunikation und eventuell deren Standardisierung. Der erste Schritt ist durch diese Arbeit getan. Ein weiterer Schritt wäre die Entwicklung der Protokolle als Programmbibliothek, um den definierten Ablauf zu testen und auch das Nachrichtenformat durch Erkenntnisse im echten Einsatz weiterentwickeln und verbessern zu können.

Anhang A

ASN.1 Darstellung der Nachrichtenformate

A.1 Nachrichtenformat für Phase I

```
GSP ::= [1] SEQUENCE
{
    Version          OCTET STRING,
    Protocol         OCTET STRING,
    Steganography    StegoData OPTIONAL,
    Cryptography     CryptoData OPTIONAL,
    Syncronisation   SyncronisationMode OPTIONAL,
    Hash             HashData OPTIONAL,
    Biometry         BiometricMode OPTIONAL,
    Signature        SignatureData OPTIONAL,
    Importance       ImportanceData OPTIONAL,
    ACK              BOOLEAN OPTIONAL,
    CRC              OCTET STRING
}

StegoData ::= [2] SEQUENCE
{
    Algorithm ::= [3] CHOICE
    {
        LSB          0x01,
        Echo         0x02,
        Phase        0x03,
        Frequency    0x04
        /* ... */
    },

    Parameter        OCTET STRING
}

CryptoData ::= [4] SEQUENCE
{
    Algorithm ::= [5] CHOICE
    {
        RSA          0x01,
        ElGamalDL    0x02,
        ElGamalEC    0x03,
        AES          0x04
        /* ... */
```

```
    },

    KeyLength       INTEGER OPTIONAL,
    Key             OCTET STRING OPTIONAL
}

SynchronisationMode ::= [6] CHOICE
{
    Pattern ::= [7] SEQUENCE
    {
        HasEmbData BIT STRING,
        StartAt    BIT STRING,
        EndAt      BIT STRING
    }

    /* ... */
}

HashData ::= [8] SEQUENCE
{
    Algorithm ::= [9] CHOICE
    {
        MD5         0x01,
        SHA-1       0x02,
        SHA-256     0x03,
        SHA-512     0x04
        /* ... */
    },

    FutureHash      BIT STRING,
    HashedData      OCTET STRING
}

BiometricMode ::= [10] CHOICE
{
    Password        0x01,
    Face            0x02,
    Voice           0x03,
    Fingerprint     0x04,
    EyeIris         0x05,
    EyeRetina       0x06,
    HandGeometry    0x07,
    Signature       0x08,
    Keystrokes      0x09,
    Lips            0x0a,
    ThermalFaceIm   0x0b,
    ThermalHandIm   0x0c,
    Gait            0x0d,
    Odor            0x0e,
    DNA             0x0f,
    EarShape        0x10,
    FingerGeometry  0x11,
    PalmGeometry    0x12,
    Veins           0x13
    /* ... */
}

SignatureData ::= [11] SEQUENCE
{
    Algorithm ::= [12] CHOICE
    {
        RSA         0x01,
        DSA         0x02,
        ECDSA       0x03
    },
```

```
    Key             OCTET STRING OPTIONAL,
    SignatureValue  OCTET STRING
}

ImportanceData ::= [13] SEQUENCE
{
    Mode ::= [14] CHOICE
    {
        MultipleTransmission 0x01,
        Parity       0x02
        /* ... */
    }

    Parameter       OCTET STRING OPTIONAL,
    Priority        INTEGER OPTIONAL
}
```

Listing A.1: ASN.1 Beschreibung des Nachrichtenformats für Phase I, Aushandlung des Parametersatzes

A.2 Nachrichtenformat für Phase II

```
Data ::= [1] SEQUENCE
{
    Version          OCTET STRING,
    Protocol         OCTET STRING,
    SequenceNumber   INTEGER,
    Payload          OCTET STRING,
    DataEnd          BOOLEAN,
    AuthSig          AuthSigData OPTIONAL,
    ErrCorrection    ErrCorrectionData OPTIONAL,
    ResendInfo       ResendData OPTIONAL,
    CRC              OCTET STRING
}

AuthSigData ::= [2] SEQUENCE
{
    AuthData         OCTET STRING OPTIONAL,
    Signature        OCTET STRING OPTIONAL
}

ErrCorrectionData ::= [3] SEQUENCE
{
    CorrectionCode OCTET STRING
}

ResendData ::= [4] SEQUENCE
{
    MissingSeqNrs   SET OF INTEGER
}
```

Listing A.2: ASN.1 Beschreibung des Nachrichtenformats für Phase II, Übertragung der geheimen Daten

A.3 Beispiele für Tags in Phase I

A.3.1 GSP-I mit LSB-Einbettung und symmetrischer Verschlüsselung

```
TAG ::= SEQUENCE
{
    TAG-NR ::= 0x01,
    KryptoKey OCTET STRING,
    Password  OCTET STRING,
    CRC OCTET STRING
}
```

Listing A.3: Parametersatz Tag 0x01

Wie dargestellt ist, sind für die gesamten Informationen aus Anhang A.1 nur drei Werte erforderlich, welche nicht durch ein Tag vorgegeben werden können. Dies sind ein kryptographischer Schlüssel (*KryptoKey*), ein Passwort (*Password*) zur Authentifikation und die CRC Prüfsumme (*CRC*). Tabelle A.1 zeigt die definierten Werte für Tag 0x01.

	WERT
VERSION	0x01
PROTOCOL	0x01
STEGANOGRAPHY	
ALGORITHM	0x01
CRYPTOGRAPHY	
ALGORITHM	0x04
KEYLENGTH	192
BIOMETRY	0x01

Tabelle A.1: Definierte Werte für Tag 0x01

A.3.2 GSP-II mit LSB-Einbettung und Einsatz von RSA

```
TAG ::= SEQUENCE
{
    TAG-NR ::= 0x02,
    Signature OCTET STRING,
    CRC       OCTET STRING
}
```

Listing A.4: Parametersatz Tag 0x02

Für diese Variante ist neben der CRC Prüfsumme (*CRC*) nur noch die Angabe einer digitalen Signatur (*Signature* – zur Integritätsprüfung und Authentifizierung) nötig. Dies setzt jedoch voraus, dass die öffentlichen RSA-Schlüssel beider Kommunikationspartner untereinander bereits ausgetauscht wurden. Alle anderen Informationen werden über den Tag 0x02 bestimmt (siehe Tabelle A.2).

	WERT
VERSION	0x01
PROTOCOL	0x02
STEGANOGRAPHY	
ALGORITHM	0x01
CRYPTOGRAPHY	
ALGORITHM	0x01
SIGNATURE	
ALGORITHM	0x01

Tabelle A.2: Definierte Werte für Tag 0x02

A.3.3 GSP-III mit LSB-Einbettung und symmetrischer Verschlüsselung

```
TAG ::= SEQUENCE
{
    TAG-NR ::= 0x03,
    KryptoKey           OCTET STRING,
    Syncronisation Pattern,
    HashValue           BIT STRING,
    HashedData          OCTET STRING,
    CRC                 OCTET STRING
}

Pattern ::= SEQUENCE
{
    HasEmbData          BIT STRING,
    StartAt             BIT STRING,
    EndAt               BIT STRING
}
```

Listing A.5: Parametersatz Tag 0x03

Die erforderlichen Informationen für diesen Tag sind umfangreicher, da mehr variable Daten verwendet werden, als bei den vorigen Beispielen. Es muss ein kryptographischer Schlüssel angegeben werden (*KryptoKey*), der Sender hat für die Synchronisation zu sorgen und die Parameter für diese Verbindung einzutragen (*HasEmbData*, *StartAt*, *EndAt*) und der Hashwert über zukünftige Daten ist mitzusenden (*HashValue*, *HashedData*). Tabelle A.3 stellt jene Werte dar, welche durch Tag 0x03 definiert werden.

	WERT
VERSION	0x01
PROTOCOL	0x03
STEGANOGRAPHY	
ALGORITHM	0x01
CRYPTOGRAPHY	
ALGORITHM	0x01
KEYLENGTH	192
HASH	
ALGORITHM	0x02

Tabelle A.3: Definierte Werte für Tag 0x03

Anhang B

Man in the Middle Angriff

B.1 Das Szenario

Ein Man in the Middle ist ein Angreifer, welcher auf einem Kommunikationskanal *physikalisch* oder *logisch* zwischen zwei Entitäten auftritt. In heutigen Arten dieses Angriffs befindet sich der Man in the Middle nahezu immer logisch zwischen den Kommunikationspartnern (vgl. Ferguson und Schneier, 2003, Wikipedia, 2008d). Bruce Schneier beschreibt den Angriff folgendermaßen:

> „The phrase *man-in-the-middle attack* is used to describe a computer attack where the adversary sits in the middle of a communications channel between two people, fooling them both.“ (Schneier, 2004)

Bezogen auf den Handlungsspielraum wird häufig zwischen zwei Arten von Angreifern unterschieden:

- **Passiver Angreifer.** Hat die Möglichkeit, den Datenverkehr zwischen den Entitäten abzuhören und einzusehen.
- **Aktiver Angreifer.** Kann zusätzlich Daten am Übertragungsweg manipulieren und neue (selbst erstellte) Pakete (z.B. mit gefälschter Identität) einbringen (vgl. Franz und Pfitzmann, 1999).

In heutigen Umgebungen muss davon ausgegangen werden, dass einem Man in the Middle alle Ressourcen zur Verfügung stehen, um komplexe Aufgaben effizient durchführen zu können (wie auch schon unter Abschnitt 4.2.2.1 dargestellt wurde). Daraus folgt, dass man nicht davon ausgehen darf, in einem komplexen Aufbau nicht mit der Man in the Middle Problematik konfrontiert zu sein.

B.2 Angriffsarten

Ein Angreifer hat viele verschiedene Möglichkeiten, um sich als Man in the Middle in eine Kommunikation einzuschleusen. Dies hängt auch von der gegebenen Netzwerkstruktur und dem Handlungsspielraum des Angreifers ab. Beispiele für Angriffsarten sind:

- Abhören von Übertragungsmedien (Leitungen, Luft) bei physikalischem Zugriff bzw. Vorhanden sein von nötigem Equipment.
- Konfiguration eines Dynamic Host Configuration Protocol (DHCP) Servers, welcher die Anfragen der Clients zu vom Angreifer kontrollierten Diensten weiterleitet.
- Übernahme eines Routers im Internet (z.B. durch Sicherheitslücke im Betriebssystem des Routers) (vgl. Wikipedia, 2008d).
- Änderung der Identität durch Vorgabe einer falschen Hardwareadresse (MAC Adresse) in einem Netzwerk (*Address Resolution Protocol (ARP) Spoofing*) (vgl. Wikipedia, 2008a).
- Angriff des Domain Name System (DNS) und Vorgabe von falschen Informationen an Clients, damit sich diese zu gefälschten Servern verbinden (*DNS Cache Poisoning*) (vgl. Kaminsky, 2008).

Neben den hier genannten Beispiele existieren natürlich noch viele andere Arten, einen Man in the Middle Angriff auszuführen. Außerdem sind viele Tools erhältlich (vgl. Wikipedia, 2008d), welche Teilaspekte eines Angriffs, oder ganze Angriffe, automatisieren.

B.3 Schutzmaßnahmen

B.3.1 Schutz der Daten vor Einsicht

Eine relativ einfache Maßnahme, um sensitive Daten vor den Augen eines Man in the Middle zu schützen, ist der Einsatz von Verschlüsselung. Wenn der Datenverkehr verschlüsselt zwischen den Kommunikationspartnern versendet wird, weiß ein Angreifer zwar über deren Existenz, kann jedoch (bei geeigneten Verschlüsselungsalgorithmen) den Klartext nicht ohne Kenntnis über den kryptographischen Schlüssel berechnen (vgl. Wikipedia, 2008d).

B.3.2 Schutz der Daten vor Manipulation

Ist der Man in the Middle ein aktiver Angreifer, schützt eine Verschlüsselung zwar davor, dass der Angreifer den Datenverkehr mitliest, jedoch kann er die Daten unentdeckt verändern. Abhängig von den gesendeten Informationen ist diese Tatsache oft schwerwiegender als die Möglichkeit des Mitlesens. Um die Integrität zu gewährleisten, können ebenfalls Mechanismen der Kryptographie eingesetzt werden. Darunter fallen der Einsatz eines MAC bei Secret-Key Kryptographie oder die digitale Signatur bei Public-Key Kryptographie. Durch die Berechnung eines Codes, dessen Ergebnis durch die Daten beeinflusst wird, entdeckt der Empfänger eine Manipulation am Übertragungsweg durch die Neuberechnung des Codes in Abhängigkeit der erhaltenen Daten und einem Vergleich mit dem erhaltenen Code (vgl. Wikipedia, 2008d).

B.3.3 Schutz vor Entdecken von sensiblen Daten

In manchen Situationen kann es Vorkommen, dass keiner über eine Kommunikation Bescheid wissen darf (außer den Kommunikationspartnern). Dies könnte z.B. der Fall sein, wenn Firmen Fusionen vorbereiten, Journalisten in Ländern operieren, welche dem Heimatland feindlich gesinnt ist, etc. In solchen Fällen ist der Schutz vor dem Mitlesen und dem Manipulieren nicht ausreichend, da nicht der Inhalt im Vordergrund steht, sondern einzig die Tatsache, dass eine Kommunikation sattfindet schon zu auffällig wäre. Hierbei kann auf die Steganographie zurückgegriffen werden (siehe Kapitel 2). Die herkömmliche Steganographie hat jedoch wiederum eigene Problematiken in Bezug auf Angriffe, welche in dieser Arbeit behandelt wurden.

Abkürzungsverzeichnis

ARP Address Resolution Protocol
ASN.1 Abstract Syntax Notation One

BER Basic Encoding Rules

CRC Cyclic Redundancy Check

DCT Diskrete Kosinus Transformation
DFT Diskrete Fourier Transformation
DHCP Dynamic Host Configuration Protocol
DNS Domain Name System
DRM Digital Rights Management

FCS Frame Check Sequence

GIF Graphics Interchange Format
GSM Global System for Mobile Communications
GSP General Purpose Steganographic Protocol

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronics Engineers
IP Internet Protocol
ISO Internation Organisation for Standardisation

JPEG Joint Photographic Experts Group

LAN Local Area Network
LSB Least Significant Bit

MAC Media Access Control
MAC Message Authentication Code
MD5 Message Digest 5

OSI	Open Systems Interconnection
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
QoS	Quality of Service
RAID	Redundant Array of Inexpensive Disks
RFC	Request for Comments
SHA	Secure Hash Algorithm
SKE	Steganographic Key Exchange
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP

Glossar

ASCII

American Standard Code for Information Interchange — eine Darstellungsform von Zeichen in 7-Bit Werten

Base-64

Das Base-64 Verfahren stellt eine Möglichkeit dar, 8-Bit Binärwerte in eine Zeichenfolge umzuwandeln, welche exklusiv aus druckbaren ASCII Zeichen besteht. Das Verfahren ist in (Josefsson, 2006) definiert.

BZip2

Zweistufiges, verlustfreies Komprimierungsverfahren welches die *Huffman Kodierung* anwendet (vgl. Seward, 2008).

Counter Mode

Verschlüsselungsmodus welcher einen Block Cipher als Stream Cipher einsetzt. Pro Block der Nachricht wird ein Schlüssel durch Verschlüsselung einer Nonce und einem Zähler (*Counter*) erstellt. Diese Ausgabe wird mit dem Nachrichtenblock XOR verknüpft. Dadurch entstehen keine Abhängigkeiten einzelner Cipher Blöcke (vgl. Ferguson und Schneier, 2003, S. 75 f).

Diffie-Hellman

Das *Diffie-Hellman* Protokoll ist das älteste asymmetrische Kryptosystem. Es ermöglicht zwei Entitäten einen sicheren symmetrischen Schlüssel über einen unsicheren Kommunikationskanal auszutauschen. Dieses Verfahren wurde erstmals in (Diffie und Hellman, 1976) definiert. (Schneier, 1995) enthält eine aktuelle Beschreibung des Protokollablaufs.

Ethernet

Technik zur Kommunikation in Netzwerken (vor allem in LAN Umgebungen), welche auf Schicht zwei des ISO/OSI Referenzmodells arbeitet.

Hop

Der Weg von einem Knotenpunkt im Netzwerk zum nächsten. In Datennetzwerken stellen vor allem Router die „Hops“ auf dem Übertragungsweg dar.

Man in the Middle

Angreifer, welcher am Übertragungsweg zwischen zwei Kommunikationspartnern die Verbindung abhören kann. Wird in der Fachliteratur oft *Eve* oder *Ward* genannt.

Nonce

Number used Once, eine Zahl, welche z.B. bei Authentifzierungsverfahren eingesetzt werden kann um Replay Attacken abzuwehren. Für jede Verbindung wird eine neue Nonce generiert.

OpenPGP

Freie Implementierung von PGP zur verschlüsselten Kommunikation. Die OpenPGP Implementierung unterstützt symmetrische und Public-Key Kryptographie, digitale Signaturen und PKI.

Public-Key Kryptographie

Verschlüsselungsverfahren, welches zum Ver- und Entschlüsseln zwei verschiedene Schlüssel verwendet. Diese asymmetrischen Verfahren benötigen keinen sicheren Schlüsseltausch, sind aber langsamer als symmetrische Verschlüsselungsverfahren. Beispiele sind *RSA* und *ElGamal*.

Radix-64

Eine Erweiterung des Base-64 Verfahrens welches, nachdem die Daten in Base-64 umgewandelt wurden, eine CRC Prüfsumme berechnet und an die Daten anhängt.

Replay Attack

Eine kryptographische Angriffsart mit welcher ein Angreifer versucht, gesendete Daten aufzuzeichnen um diese zu einem späteren Zeitpunkt nachsenden zu können. Dies zielt auf Kompromittierung der Authentizität ab, da der Angreifer durch das Nachsenden von real gesendeten Daten den Empfänger zu täuschen versucht, indem er sich als der ursprüngliche Sender ausgibt. Eine Gegenmaßnahme ist z.B. der Einsatz von Nonces zur Authentifizierung in kryptographischen Anwendungen (vgl. Schneier, 1995, S. 58 f).

Secret-Key Kryptographie

Diese Verfahren wenden denselben Schlüssel zum Ver- und Entschlüsseln an. Somit ist ein sicherer Austausch der Schlüssel zwischen Sender und Empfänger notwendig. Die symmetrischen Verfahren benötigen kleinere Schlüssellängen und sind somit schneller als asymmetrische Verfahren. Beispiele sind *AES* und *DES*.

XOR

Exklusives Oder, eine logische Operation welche folgende Ergebnisse liefert: $1 \otimes 1 = 0; 0 \otimes 0 = 0; 1 \otimes 0 = 1; 0 \otimes 1 = 1$

ZIP

Ein Komprimierungsverfahren für beliebige Daten, arbeitet nach dem *deflate* Prinzip (vgl. Deutsch, 1996).

ZLib

Ein Komprimierungsverfahren ähnlich zu ZIP (vgl. Deutsch und Gailly, 1996).

Literaturverzeichnis

[von Ahn und Hopper 2004] AHN, L. von ; HOPPER, N.J.: Public-Key Steganography. In: *Eurocrypt 2004* (2004)

[Bender u. a. 1996] BENDER, W. ; GRUHL, D. ; MORIMOTO, N. ; LU, A.: Techniques for data hiding. In: *IBM Systems Journal* 35 (1996), Nr. 3&4, S. 313–336

[Cachin 2005] CACHIN, C.: Digital Steganography. In: *IBM Research, Zurich Research Laboratory* (2005)

[Callas u. a. 2007] CALLAS, J. ; DONNERHACKE, L. ; FINNEY, H. ; SHAW, D. ; THAYER, R.: *RFC 4880 — OpenPGP Message Format*. http://ftp.rfc-editor.org/in-notes/rfc4880.txt. November 2007

[Chung 2001] CHUNG, S.-J.: Network architecture: hamming codes and cyclic redundancy for transmission error correction. In: *SIGCSE Bull.* 33 (2001), Nr. 4, S. 48–50

[Contini u. a. 2007] CONTINI, S. ; STEINFELD, R. ; PIEPRZYK, J. ; MATUSIEWICZ, K.: A critical look at cryptographic hash function literature. In: *ECRYPT Hash Workshop*, 2007

[Deutsch 1996] DEUTSCH, P.: *RFC 1951 — DEFLATE Compressed Data Format Specification version 1.3*. ftp://ftp.rfc-editor.org/in-notes/rfc-1951.txt. May 1996

[Deutsch und Gailly 1996] DEUTSCH, P. ; GAILLY, J.-L.: *RFC 1950 — ZLIB Compressed Data Format Specification version 3.3*. ftp://ftp.rfc-editor.org/in-notes/rfc-1950.txt. May 1996

[Diffie und Hellman 1976] DIFFIE, W. ; HELLMAN, M.E.: New Directions in Cryptography. In: *IEEE Transactions on Information Theory*, November 1976, S. 644–654

[Federrath und Wicke 1997] FEDERRATH, H. ; WICKE, E.: Vertrauliche Kommunikation mit Steganographie. In: *Praxis der Informationsverarbeitung und Kommunikation* TU Dresden, Fakultät Informatik (Veranst.), 1997, S. 134–137

[Ferguson und Schneier 2003] FERGUSON, N. ; SCHNEIER, B.: *Practical Cryptography*. John Wiley & Sons, 2003

[Franz und Pfitzmann 1999] FRANZ, E. ; PFITZMANN, A.: Steganography Secure against Cover-Stego-Attacks. In: (Pfitzmann, 2000), S. 29–46

[Fridrich 2004] FRIDRICH, J.J. (Hrsg.): *Information Hiding, 6th International Workshop, IH 2004, Toronto, Canada, May 23-25, 2004, Revised Selected Papers*. Bd. *3200*. Springer, 2004. (Lecture Notes in Computer Science)

[Glover und Grant 2003] GLOVER, I. ; GRANT, P.M.: *Digital Communications*. Pearson Education, 2003

[Hübner 2003] HÜBNER, C.: *Grundlagen der Steganographie*. http://www.ra.cs.uni-tuebingen.de/lehre/ws02/pro_sicherheit_ausarbeitung/Huebner_Steganographie.pdf. 2003

[IEEE 2005] IEEE: *Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specification*. 2005

[ISO/IEC 1996] ISO/IEC: *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. June 1996

[ITU-T 2002a] ITU-T: *X.680 — Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*. July 2002

[ITU-T 2002b] ITU-T: *X.690 — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*. July 2002

[Johnson und Katzenbeisser 2000] JOHNSON, N.F. ; KATZENBEISSER, S.: A Survey of Steganographic Techniques. In: (Katzenbeisser und Petitcolas, 2000), S. 43–78

[Josefsson 2006] JOSEFSSON, S.: *RFC 4648 — The Base16, Base32, and Base64 Data Encodings*. ftp://ftp.rfc-editor.org/in-notes/rfc4648.txt. October 2006

[Kaminsky 2008] KAMINSKY, D.: *Why So Serious*. http://www.doxpara.com/?p=1204. 2008

[Katzenbeisser und Petitcolas 2000] KATZENBEISSER, S. (Hrsg.) ; PETITCOLAS, F.A.P. (Hrsg.): *Information Hiding Techniques for Steganography and Digital Watermarking*. Norwood, MA, USA : Artech House, Inc., 2000

[Koblitz 2004] KOBLITZ, N.: *A Course in Number Theory and Cryptography*. Springer Verlag, 2004

[Kundur und Ahsan 2003] KUNDUR, D. ; AHSAN, K.: Practical Internet Steganography: Data Hiding in IP. In: *Proceedings of the Texas Workshop on Security of Information Systems*, April 2003

[Kühne 2007] KÜHNE, T.: *Steganographie in WLANs – Design und Implementierung*. VDM Verlag Dr. Müller, 2007

[Lucena u. a. 2004] LUCENA, N.B. ; PEASE, J. ; YADOLLAHPOUR, P. ; CHAPIN, S.J.: Syntax and Semantics-Preserving Application-Layer Protocol Steganography. In: (Fridrich, 2004), S. 164–179

[Menezes u. a. 1996] MENEZES, A.J. ; OORSCHOT, P.C. van ; VANSTONE, S.A.: *Handbook of Applied Cryptography*. CRC Press, 1996

[Mittelholzer 1999] MITTELHOLZER, T.: An Information-Theoretic Approach to Steganography and Watermarking. In: (Pfitzmann, 2000), S. 1–16

[Mukherjee u. a. 2008] MUKHERJEE, S. ; BHATTACHARYA, S. ; CHAUDHURY, A.: Triple layer data security. In: *Ubiquity* 9 (2008), Nr. 17

[Murdoch und Lewis 2005] MURDOCH, S.J. ; LEWIS, S.: Embedding Covert Channels into TCP/IP. In: *Information Hiding, Seventh International Workshop, IH 2005, Barcelona, Spain, June 6-8, 2005*, Springer, 2005, S. 247–261

[Patterson u. a. 1988] PATTERSON, D.A. ; GIBSON, G. ; KATZ, R.H.: A case for redundant arrays of inexpensive disks (RAID). In: *SIGMOD Rec.* 17 (1988), Nr. 3, S. 109–116

[Petitcolas u. a. 1999] PETITCOLAS, F.A.P. ; ANDERSON, R.J. ; KUHN, M.G.: Information Hiding - A Survey. In: *Proceedings of the IEEE*, 1999, S. 1062–1078

[Pfitzmann 2000] PFITZMANN, A. (Hrsg.): *Information Hiding, Third International Workshop, IH'99, Dresden, Germany, September 29 - October 1, 1999, Proceedings*. Bd. *1768*. Springer, 2000. (Lecture Notes in Computer Science)

[Pfitzmann 1996] PFITZMANN, B.: Information Hiding Terminology – Results of an Informal Plenary Meeting and Additional Proposals. In: *Information Hiding, First International Workshop, Cambridge, U.K., May 30 - June 1, 1996*, Springer, 1996, S. 347–350

[Piller 2007] PILLER, E.: *StegIT — Machbarkeitsstudie für Anti-Steganographie-Lösung für VoIP und GSM. Endbericht des KIRAS Forschungsprojektes StegIT*. 2007

[Rowland 1997] ROWLAND, C.H.: *Covert Channels in the TCP/IP Protocol Suite*. http://ebipol.p.lodz.pl/Content/1081/issues/issue2_5/rowland/. May 1997

[Schneier 1995] SCHNEIER, B.: *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, 1995

[Schneier 2004] SCHNEIER, B.: *Crypto-Gram Newsletter*. http://www.schneier.com/crypto-gram-0404.html. April 2004

[Seward 2008] SEWARD, J.: *The Bzip2 and libbzip2 home page*. http://www.bzip.org. 2008

[Simmons 1984] SIMMONS, G.J.: The Prisoners' Problem and the Subliminal Channel. In: *Advances in Cryptology: Proceedings of Crypto 83*, Plenum Press, 1984, S. 51–67

[Struif 2002] STRUIF, B.: Standardisierung im Bereich SmartCards & Biometrie. In: *12. SIT-SmartCard Workshop Tagungsband* Fraunhofer-Institut für Sichere Telekooperation - SIT -, Darmstadt (Veranst.), February 2002

[Szczypiorsky 2003] SZCZYPIORSKY, K.: HICCUPS: Hidden communication system for corrupted networks. In: *Proceedings of the Tenth International Multi-Conference on Advanced Computer Systems ACS'2003, October 22-24, 2003 Międzyzdroje*, 2003, S. 31–40

[Vogel u. a. 2006] VOGEL, T. ; DITTMANN, J. ; HILLERT, R. ; KRÄTZER, C.: Design und Evaluierung von Steganographie für Voice-over-IP. In: *Sicherheit*, 2006, S. 131–142

[Wikipedia 2008a] WIKIPEDIA: *ARP-Spoofing — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/wiki/ARP-Spoofing. 2008

[Wikipedia 2008b] WIKIPEDIA: *Cyclic redundancy check — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/Cyclic_redundancy_check. 2008

[Wikipedia 2008c] WIKIPEDIA: *Kryptologische Hashfunktion — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/wiki/Kryptologische_Hashfunktion. 2008

[Wikipedia 2008d] WIKIPEDIA: *Man-in-the-middle-Angriff — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/wiki/Man-in-the-middle-Angriff. 2008

[Wikipedia 2008e] WIKIPEDIA: *Paritätsbit — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/wiki/Paritätsbit. 2008

[Wikipedia 2008f] WIKIPEDIA: *Steganographie — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/wiki/Steganographie. 2008