PGPfone – vertrauliche Gespräche über das Netz

Studienarbeit am

Lehrstuhl für Rechnernetze und Internet
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften
Universität Tübingen

von

Alexander Rosenstiel

Betreuer:
Dipl.-Inform. Heiko Niedermayer
Dr.-Ing. Falko Dressler

Tag der Anmeldung: 1. November 2003
Tag der Abgabe: 1. Mai 2004

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Tübingen, den 12.04.2004

Kurzfassung

Aufgabenstellung und Motivation der vorliegenden Arbeit war das Untersuchen des Voice-over-IP-Programms PGPfone. Mit Hilfe dieses Open-Source-Produktes sollten Tests durchgeführt werden, um eine Aussage über die Qualität des Programms treffen zu können. Performance-Parameter sollten angegeben und das Programm für eine Beurteilung gegebenenfalls erweitert werden.
Zur Auswertung wurden verschiedene Statistiken, die PGP mitliefert, untersucht. Weiter wurden diverse Signalaufzeichnungen und Frequenzanalysen vorgenommen und ausgewertet. Da PGPfone mehrere Koder zur Verfügung stellt, wurden diese näher untersucht und Unterschiede festgestellt. Die Testumgebung wurde erweitert, um ein Netzwerk besser simulieren zu können. Dazu wurde das Programm NistNet verwendet.
Des Weiteren erfolgte im Laufe der Arbeit eine Messung des Delays sowie die Berechnung der daraus resultierenden Standardabweichung. Abschließend wurden die sich ergebenden Ergebnisse mit anderen Voice-over-IP Programmen verglichen und daraus Folgerungen für eine Verwendbarkeit des Programms getroffen.

Inhaltsverzeichnis

1. Einführung

Im Rahmen meiner Studienarbeit habe ich mich entschlossen, das Programm PGPfone näher zu untersuchen, zu testen und zu erweitern. PGPfone ist ein Voice-over-IP-System, das das Telefonieren über Internet möglich macht. Wie auch schon der Name des Programms verrät, handelt es sich um ein Programm, das nicht nur das Telefonieren ermöglichen, sondern mit Verschlüsselungstechniken das Telefonieren auch möglichst abhörsicher machen will.

Als erstes soll eine kurze Einführung gegeben werden, was man unter VoIP-Programmen zu verstehen hat.

"Voice-over-IP", wörtlich übersetzt "Sprache über Internet-Protokoll", wird schon seit längerer Zeit entwickelt, um eine neue Möglichkeit zu finden, das Telefonieren einfacher und preiswerter zu machen.
Doch die erste Euphorie verflog schnell: Am Anfang wurden VoIP-Systeme mit Werbung finanziert, die erhoffte Wirksamkeit blieb aber aus, so dass sich werbefinanzierte VoIP-Angebote nicht mehr kostendeckend halten konnten. Die meisten Angebote verschwanden schnell wieder vom Markt oder wurden kostenpflichtig.
Inzwischen scheint aber das Telefonieren über das Netz jetzt tatsächlich vor dem Durchbruch zu stehen. Große Unternehmen wickeln ihren Telefonverkehr zunehmend über VoIP ab, verschiedene Anbieter wie QSC und broadnet mediascape haben nun auch in Deutschland VoIP-Produkte auf dem Markt.
Die Festnetzsparte der Deutschen Telekom, T-Com, kündigte mittlerweile an, ihr Netz langfristig auf die Internet-Übertragungstechnologie umzustellen. In weiterer Zukunft soll es mit solchen Programmen auch möglich werden, normale Festnetz- oder Mobilfunkanschlüsse erreichen zu können. Das wäre dann wohl letztendlich der Durchbruch von VoIP-Programmen.

Eines der kostenlosen Tools, um über das Internet zu telefonieren, ist PGPfone. Dieses soll nun hier näher betrachtet werden. PGPfone unterscheidet sich von anderen angebotenen Programmen im Wesentlichen dadurch, dass ein Schwerpunkt auf die Sicherheit und Abhörsicherheit der Gespräche gelegt wird.

Erster Teil der Studienarbeit ist das Installieren und Testen des Programms. Dabei wird eine Beschreibung des Programms angegeben und die Funktionsweise erläutert. Für den zweiten Schritt wird der Quelltext des Programms wichtig, hier soll zunächst eine Einarbeitung erfolgen und danach der Quelltext um entsprechende Funktionen erweitert werden, die den dritten und letzten Schritt, nämlich die Bewertung der Qualität des Programms, vereinfachen sollen. Die Qualität soll möglichst objektiv in Messwerten erfasst werden, um nicht auf subjektive Eindrücke angewiesen zu sein. Auch visuelle Auswertungen sollen Ziel der Arbeit sein.

2. Die Aufgabenstellung

Mit Hilfe des Source-Code soll die Fragestellung der Funktionsweise und der Benutzbarkeit geklärt werden. Des Weiteren ist die Aufgabe die Verbindungsqualität durch entsprechende Performance-Parameter auszudrücken. Die Sicherheitsziele sollen näher untersucht und die Auswirkungen der Verschlüsselungstechniken auf die Qualität betrachtet werden.

Die erste Aufgabe war das Installieren des Programms und das anschließende Testen. Danach sollte der Quelltext kompilierfähig gemacht werden. Eine Einarbeitung in den Quelltext war der nächste Schritt sein, um das Programm zu durchschauen und Fragen, wie zum Beispiel an welcher Stelle die Verschlüsselung ansetzt, zu klären.

Mit dem laufenden Programm sollte nun eine Testumgebung geschaffen werden, die verschiedene Tests möglich macht. Hierzu musste man auf einem weiteren dritten Rechner das Programm NistNet installieren. Bei diesem Programm handelt es sich um eine Simulation eines Netzes. Hier sind Parameter wie einen Delay oder eine Drop- bzw. Dub-Rate einstellbar.

Die Statistiken, die PGPfone zur Verfügung stellt, sollten näher untersucht und anhand des Quelltexts die Berechnungsmethoden gefunden werden. Eine möglichst gute Analyse dieser Daten sollte möglich sein. Um Tests leichter auswertbar zu machen, musste man PGPfone so erweitern, dass es auch möglich war vorgefertigte Audiodateien zu übertragen, denn nur so sind verschiedene Tests miteinander vergleichbar.
Die Tests sollten möglichst visuell zu sehen sein, wie zum Beispiel mit Hilfe von Signalaufzeichnungen und Frequenzbändern. Die verschiedenen Koder sollten genauer untersucht und Unterschiede festgestellt werden. Ebenso sollten die verschiedenen Verschlüsselungsverfahren untersucht und die Frage beantwortet werden, wie hoch der Qualitätsverlust bei eingeschalteter Verschlüsselung ist.

Wenn möglich war auch noch eine Messung des Delays und der Standardabweichung bei verschiedenen Einstellungen durchzuführen.

Endziel der Studienarbeit soll es sein, eine ausführliche Qualitätsanalyse des Programms durchzuführen und daraus resultierende Folgerungen über die Einsetzbarkeit von PGPfone zu treffen.

3. Das Programm im Überblick

3.1 Das Hauptfenster

PGPfone ist ein Voice-over-IP – Programm, bei dem die Sicherheit eine wichtige Rolle spielt. Es handelt sich um ein Programm, welches den Input eines Mikrofons aufnimmt, digitalisiert, komprimiert, verschlüsselt und anschließend über das Internet versendet. Ausgelegt ist das Programm schon ab einem Modem mit der Übertragungsrate von 14,4 kbps.

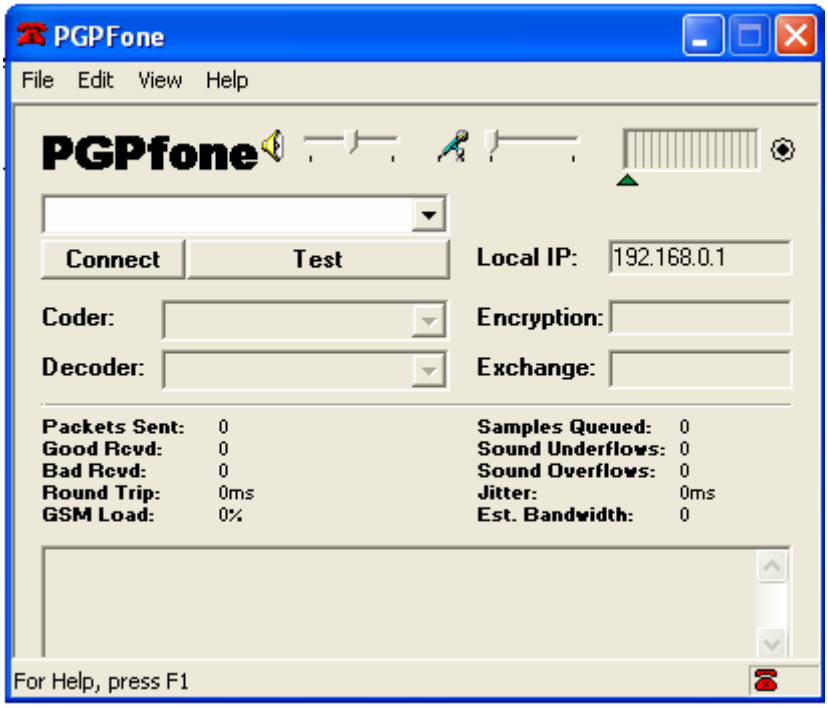

Abbildung 1: PGPfone

Die Oberfläche des Programms ist recht übersichtlich, wie sich in Abbildung 1 sehen lässt. Es gibt die Möglichkeit, eine IP-Adresse einzugeben und sich mit dem „Connect-Button“ zu verbinden. Ab dann besteht eine Verbindung mit dem angewählten Partner, vorausgesetzt dieser ist erreichbar, das heißt, er muss auf seinem Rechner auch PGPfone geöffnet haben. Wird der eingegebene Rechner ermittelt und es gibt keinen Fehler, so wird vom Programm angezeigt, dass nun auf Antwort des Gesprächspartners gewartet wird. Antwortet dieser, sieht man, dass eine Verbindung besteht, und weitere Einstellungen werden möglich. So kann man im Hauptfenster bei bestehender Verbindung den Koder oder Dekoder wählen, was später näher erläutert wird. Der Test-Button ermöglicht einen Sound-Test ohne eine vorhandene Verbindung. Man kann so zum Beispiel testen, ob der Sound funktioniert oder Lautstärkeregelungen vornehmen. Dieses ist mit zwei Reglern im oberen Bereich einmal für das Mikrofon und einmal für den Lautsprecher möglich.

Im Hauptfenster bekommt man auch weitere Informationen zu seiner eigenen IP-Adresse und dem jeweiligen Koder und Dekoder, der verwendet wird, angezeigt. Beim Umstellen des Koders während einer laufenden Verbindung, wird entsprechend der Dekoder auf der anderen Seite angepasst. So kann also jeder auf seiner Seite sowohl Koder als auch Dekoder ändern, was beim jeweiligen Gesprächspartner automatisch mit geändert wird. Als weitere Einrichtung bei PGPfone kann man die aktuelle Verschlüsselung und die Bitlänge des Schlüsselaustausches sehen.
Sehr hilfreich sind die mitgelieferten Statistiken, wie zum Beispiel die Anzahl der gesendeten Pakete, die man im Experten-Modus, den man im Menü einstellen kann, sieht. Es handelt sich hierbei um Werte, die PGPfone berechnet und ausliest. Diese werden im Laufe der Studienarbeit noch eine wichtige Rolle spielen, da man damit das Programm gut analysieren kann. Zu den Statistiken und deren Berechnung wird es ein extra Kapitel in dieser Arbeit geben, weswegen hier die einzelnen Werte nicht vorgestellt werden.

PGPfone liefert auch eine „Silence Detection“ mit, was viele andere VoIP-Programme nicht haben. Man erkennt dies an dem Balken im oberen rechten Eck. Über das kleine Dreieck unterhalb der grünen Fläche kann der entsprechende Lautstärke-Wert geändert werden. Mit welcher Lautstärke der ins Mikrofon eingesprochene Ton erkannt wird, ist durch den auf dem Balken eingezeichneten Wert zu sehen. Hier blinken immer mehr Balken auf, je lauter der Ton wird. Im rechten Bereich werden die Balken rot und man hat eine Übersteuerung und sollte die Lautstärke für das Mikrofon nach unten stellen. Es ist so sehr gut zu entscheiden, ab welcher Lautstärke man eine Übermittlung des Signals möchte. Dies kann man auch während eines Gespräches beliebig einstellen und verändern. [DOCP]

3.2 Die Menüeinstellungen

Im Menü von PGPfone hat man nun einige Einstellungsmöglichkeiten:

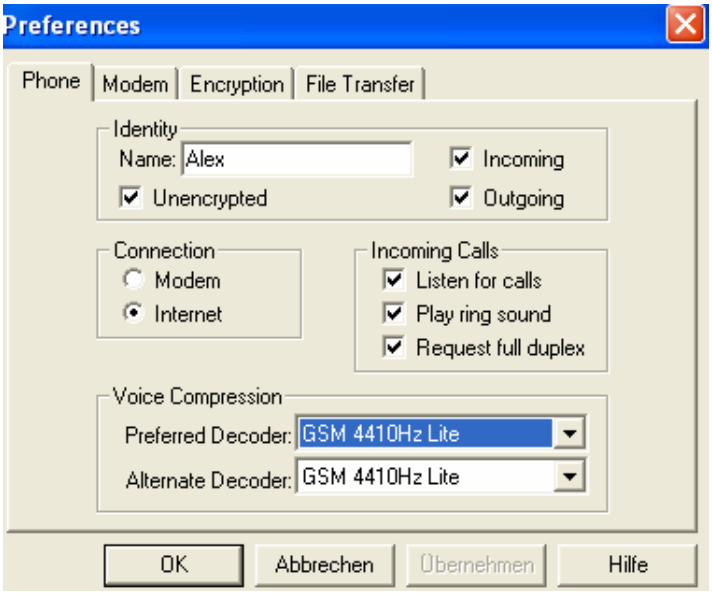

Abbildung 2: Das Menü von PGPfone

In Abbildung 2 lassen sich verschiedene Einstellungsmöglichkeiten erkennen.

Zunächst gibt es die Möglichkeit, seine Identität einzugeben, die dann auch dem Gesprächspartner angezeigt wird. Man kann des Weiteren wählen, ob die Identität bei ankommenden und/oder bei abgehenden Gesprächen angezeigt werden soll. Auch eine verschlüsselte Übermittlung der Identität ist möglich. Des Weiteren gibt es Einstellungsmöglichkeiten für die Verbindung, bei der Modem oder Internet zur Auswahl stehen.
Auch eine Reihe anderer Einstellungen sind möglich, wie man im oberen Bild erkennen kann, zum Beispiel das bevorzugte Benutzen eine Koders bzw. Dekoders. Man hat die Wahl, den „Full-Dublex-Modus“ zu benutzen oder nicht. „Full-Dublex“ bedeutet, dass man die Möglichkeit hat, jederzeit in beide Richtungen zu übertragen. Ohne diesen Modus ist PGPfone um einiges schneller, und man muss zum Sprechen immer einen Button betätigen. Da aber in der weiteren Studienarbeit vom Programm als ernsthafter Ersatz für Telefone ausgegangen wird, wurde zumeist der „Full-Dublex-Modus“ gewählt. Nur so lassen sich relevante Qualitätsaussagen treffen, denn man möchte schließlich nicht immer vorher ankündigen müssen, wenn man etwas zu sagen hat. Auch die Einstellungen, ob man bei eingehenden Anrufen einen Klingelton hören oder überhaupt Anrufe erhalten möchte, sind möglich.

Weitere Einstellungen gibt es zum Modem, die aber aufgrund der direkten Internetverbindung nicht getestet wurden. Die Übertragung einer Datei ist im Programm angedacht, aber nur für bestimmte Betriebssysteme von Apple Macintosh realisiert, so dass auch diese Einstellungen bedeutungslos sind.

Interessant sind dagegen die Einstellungen, die man bei der Verschlüsselung vornehmen kann. Ein Bild zu diesem Einstellungsfenster sieht man in Abbildung 3.

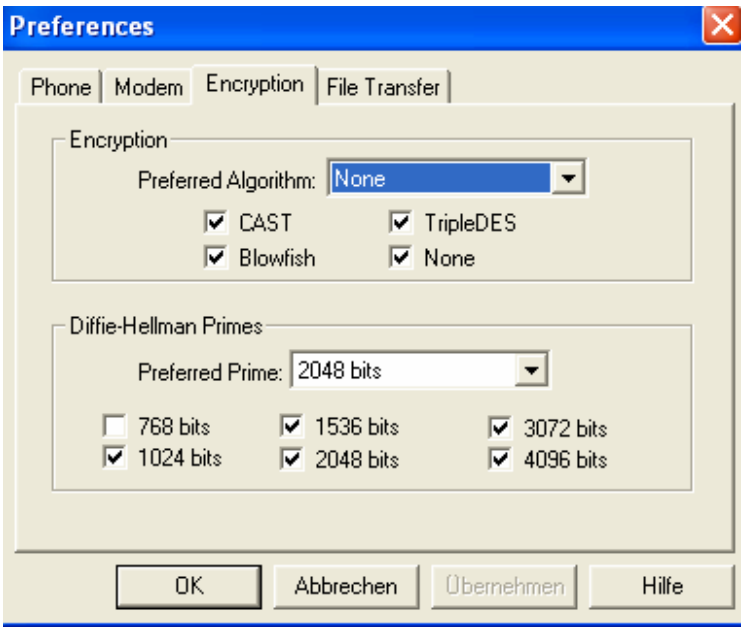

Abbildung 3: Verschlüsselungseinstellungen von PGPfone

Zunächst kann man, wie in Abbildung 3 zu sehen ist, einen bevorzugten Algorithmus zur Verschlüsselung wählen. Mit dem sich darunter befindlichen Haken wählt man aus, welchen Algorithmus man überhaupt zulassen will.

Nun versucht das Programm bei einem Anruf denjenigen Algorithmus zu wählen, der zum einen von beiden Partnern akzeptiert wird und zum anderen möglichst der bevorzugte eines oder am besten sogar beider Teilnehmer ist. Kann keine passende Verschlüsselung gefunden werden, weil die vorgenommenen Einstellungen zu unterschiedlich sind, kann auch keine Verbindung zustande kommen, und beim Abheben des Gesprächspartners bricht die Verbindung danach unmittelbar ab.

Weiter lässt sich im Einstellungsfenster eine Auswahl der Bitlänge, mit der der Schlüsselaustausch von statten gehen soll, vornehmen. Spätere Tests ergaben, dass eine hohe Bitlänge einiges an Zeit kostet, jedoch natürlich auch mehr Sicherheit bringt. So bleibt abzuwägen, was einem wichtiger ist.
Bei den weiteren Tests wurde nun eine Bitlänge von 2048 gewählt, welche auch zunächst als Standard gesetzt ist.

Die Reihenfolge der einzelnen Funktionen, die bei PGPfone ablaufen, ist in Abbildung 4 zu erkennen. Zunächst wird der Input des Mikrophons verarbeitet, codiert, verschlüsselt und anschließend in Paketen über das Netz versandt. Auf Empfängerseite erfolgen der Empfang der Pakete, die Entschlüsselung und die Dekodierung, bevor die Daten als Output an die Soundkarte übertragen werden.

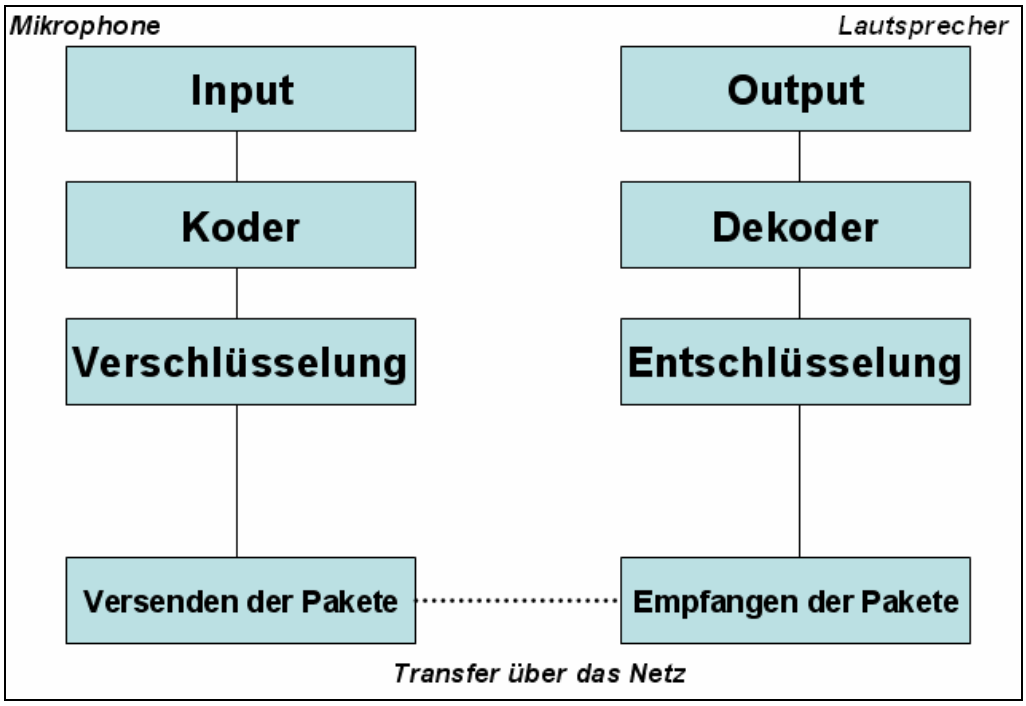

Abbildung 4: Verfahrensweise bei PGPfone

Das Format der Pakete, das PGPfone verwendet, unterscheidet sich, je nachdem ob eine Verbindung über Internet oder eine direkte Modemverbindung besteht. Das Paketformat bei der Internetübertragung besitzt je ein Byte für die Erkennung des Pakettyps, sowie für die Sequenznummer. Daraufhin folgen die Daten, die in ihrer Länge variieren können, jedoch maximal 1024 Bytes groß sein dürfen. Danach folgt der Header CRC mit bis zu 4 Bytes. Alle Pakete bei PGPfone werden als UDP-Pakete über Port 4747 gesendet. [DOCP]

Die Verschlüsselung erfolgt, indem zunächst der Schlüsselaustausch mit dem Diffie-Hellman-Verfahren erfolgt. Ein geheimer Schlüssel wird mit einer Zufallszahl generiert, deren Länge man wie oben beschrieben einstellen kann. Nun erfolgt der Austausch des geheimen Schlüssels an die beiden Parteien. Mit diesem Schlüssel kann nun ver- und auf der anderen Seite wieder entschlüsselt werden. Das Verfahren nennt man symmetrische Verschlüsselung, da der gleiche Schlüssel sowohl zum Ver- als auch zum Entschlüsseln verwendet wird. Die Vorgänge sind in Abbildung 5 zu erkennen.

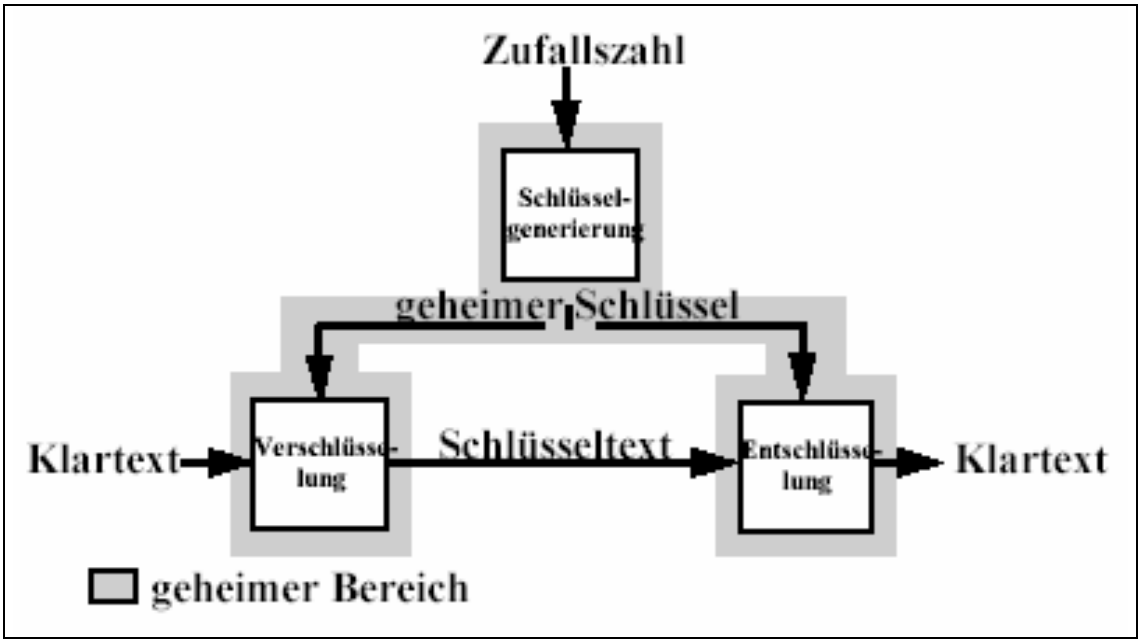

Abbildung 5: Verfahrensweise der Verschlüsselung bei PGPfone

3.3 Installation des Programms

Die Installation verlief ohne Schwierigkeiten, PGPfone läuft nur unter Windows und Macintosh-Betriebssystemen. Der Quelltext steht kostenlos im Internet zur Verfügung. Auf http://www.pgpi.org/products/pgpfone/ kann er heruntergeladen werden, wobei der Quelltext nicht komplett ist und wichtige Dateien, in denen die Verschlüsselung steckt, nicht zur Verfügung gestellt werden.
Das Programm gibt es in zwei verschiedenen Versionen 1.0 und 2.1. Im Verlauf der Studienarbeit wurde mit der aktuelleren 2.1 Version gearbeitet. Die Versionen unterscheiden sich durch die Aufmachung. Version 2.1 ist farbenreicher als ihr Vorgänger, und es sind einige Buttons hinzugekommen, wie zum Beispiel der Test-Button. Näheres dazu findet sich in der Vorstellung des Programms. Angeblich wurden einige Bugs in der neuen Version behoben. PGPfone wurde seit 1997 entwickelt. Die aktuelle Version kam 1998 auf den Markt. Die weiteren Arbeiten an diesem Projekt wurden mittlerweile eingestellt, so dass es keine neueren Versionen mehr geben wird. Die Entwickler waren Philip Zimmermann und Will Price mit der Firma Network Associates. Zu dem Programm wird eine Anleitung angeboten, die auch auf seine Entstehung etwas näher eingeht.

3.4 Testen des Programms

Der erste Test war nicht zufrieden stellend, es wurde eine sehr hohe Übersteuerung festgestellt und die übertragene Sprache war kaum verständlich. Zudem zeigte sich eine sehr hohe Prozessorauslastung während der gesamten Zeit, in der das Programm lief. Um das Programm einigermaßen weiter testen zu können, wurde die Windows-Sound-Regelung stark nach unten gestellt, so dass man nun immer besser verstehen konnte, was vom anderen Ende der Leitung kam. Trotzdem sollte sich eine Untersuchung anschließen, wie die Übersteuerung zustande kommt und ob sie gegebenenfalls behoben werden kann.
Mit den veränderten Einstellungen, das heißt einer stark nach unten korrigierten Lautstärke der Windows-Sound-Einstellungen, konnte man sich mit Hilfe des Programms ziemlich gut unterhalten. Eine geringe Verzögerung war spürbar, aber noch akzeptabel. So konnte man insgesamt doch von einer recht guten Telefonqualität sprechen. Trotzdem wurde versucht, die Übersteuerung zu minimieren, weil das Übertragen von größeren Lautstärken noch starke Probleme machte und in diesem Fall nicht mehr viel zu verstehen war.
Danach wurden sämtliche Einstellungen des Programms getestet und verschiedene Verschlüsselungsalgorithmen, die im Programm zur Verfügung stehen, ausprobiert. Man hat die Wahl zwischen drei verschiedenen Kodierungsverfahren: GSM, GSM Lite und ADPCM. Diese werden wiederum mit unterschiedlichen Frequenzen angeboten, so dass die Auswahl für das Kodier- bzw. Dekodierverfahren ziemlich groß ist.

Die zur Verfügung stehenden Möglichkeiten sind in Abbildung 6 zusammengefasst.

Übersicht über unterstützte Koder:

GSM	GSM Lite	APDCM
GSM 4410 HZ	GSM Lite 4410 HZ	ADPCM 8000 HZ
GSM 6000 HZ	GSM Lite 6000 HZ	
GSM 7350 HZ	GSM Lite 7350 HZ	
GSM 8000 HZ	GSM Lite 8000 HZ	
GSM 11025 HZ	GSM Lite 11025 HZ	

Abbildung 6: unterstützte Koder von PGPfone

Auch bei den Verschlüsselungsverfahren gibt es drei verschiedene Möglichkeiten. Hier kann zum einen die Art der Verschlüsselung gewählt werden, zum anderen hat man eine Auswahl zwischen den verschiedenen Bit-Zahlen, auf die der Schlüsselaustausch dann aufbauen soll. Die Möglichkeiten sind in Abbildung 7 zu erkennen.

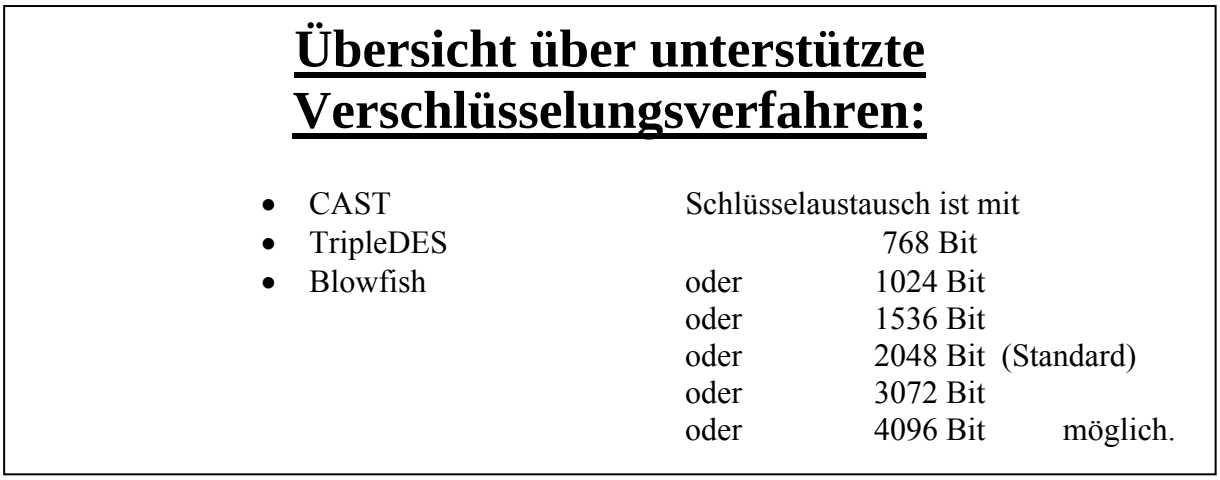

Abbildung 7: Verschlüsselungsverfahren von PGPfone

Beim Testen mit der doch recht guten Qualität und dem Testen der unterschiedlichen Einstellungen ließen sich Unterschiede zwischen den verschiedenen Kodern erkennen, und beim Einstellen der Verschlüsselung entstand ein etwas größerer Delay. Weiter wurde festgestellt, dass GSM mit der höchsten Frequenz die beste Art der Codierung war. So war die Sprache am anderen Ende der Leitung am Besten zu hören. ADPCM hinterließ auch einen guten Eindruck. Man hörte einen deutlichen Unterschied zwischen GSM und GSM Lite. GSM Lite ergab eine deutlich schlechtere Qualität. Auch bei unteren Frequenzzahlen der verschiedenen Kodern war eine wesentlich schlechtere Qualität zu erkennen.

Positiv fällt auf, dass man verschiedene Einstellungsmöglichkeiten für die Verschlüsselung und den Koder bzw. Dekoder hat. Es ist auch praktisch, dass man die Verschlüsselung komplett ausschalten kann. Zudem erscheint die Testfunktion, bei der man den Sound testen kann und sich selbst hört, als recht sinnvoll. Zusätzlich gibt es eine übersichtliche Einstellung der Lautstärke sowohl für das Mikrofon als auch für die Ausgabe. Die mitgelieferten Statistiken verschaffen einen guten Eindruck, was mit den Paketen passiert, wie viele gesendet werden und wie viele Daten in der Warteschlange stehen. Praktisch sind die Worte, die man sich bei einer verschlüsselten Verbindung vorlesen kann um sicher zu gehen, dass der andere auch wirklich der erwartete Gesprächspartner ist.

Es funktioniert folgendermaßen: Beim Aufbau einer Verbindung bekommt man einen Kasten mit verschiedenen Wörtern zu sehen, die PGPfone aus einer großen Liste zufällig zusammenstellt. Die Wörter beider Gesprächspartner sind identisch. Nun kann man sich gegenseitig abwechselnd die Wörter vorlesen und überprüfen. Stimmen die Wörter überein und man kann eventuell sogar noch die Stimme des anderen erkennen, kann man nun davon ausgehen, dass man sicher telefoniert und nicht abgehört wird.
Auch positiv auffallend ist das mitgelieferte Handbuch im PDF-Format, in dem ein Überblick über die Funktionen gegeben wird und auch Ideen und Hintergründe erläutert werden.

Negativ ist, dass nicht bei allen Einstellungen eine Verbindung erstellt werden kann. Die Idee, die bereits in der Vorstellung des Programms erläutert wurde, eine jeweils bevorzugte Codierung oder ein bevorzugtes Verschlüsselungsverfahren zuzulassen, jedoch gleichzeitig anzugeben, welche Verfahren man überhaupt zulassen will, ist gut, funktioniert aber in der Praxis nicht optimal. So passiert es ab und zu, dass keine Verbindung erstellt werden kann

oder sofort wieder beendet wird, wenn die Einstellungen der beiden Gesprächspartner unterschiedlich sind.
Auch die Verzögerung bei eingestellter Verschlüsselung ist spürbar, und auch wenn es sich um keinen besonders großen Delay handelt, ist er doch größer als beim gewohnten Telefonieren und fällt damit negativ auf. Näheres zum Thema Delay wird später erläutert. Auch weniger gut ist, dass der Quelltext relativ schlecht auskommentiert ist, was ein Nachvollziehen des vorhandenen Quelltextes wesentlich schwerer macht.

3.5 Kompilieren des Programms

Schwierigkeiten machte zunächst das Kompilieren des Quelltextes. Einige Dateien, besonders die GSM-Dateien, fehlten beim Quelltext.
Der Zugriff auf die entsprechenden Bibliotheken ist nur für amerikanische Staatsbürger unter Anforderung eines Zugangscodes gestattet. So musste auf die Suche nach diversen Dateien gegangen werden.
Über das Internet konnten einige Dateien gefunden und eingebunden werden. Danach traten einige LINK-Fehler auf, die behoben werden konnten, in dem eine fehlende Datei neu geschrieben wurde. Da das Programm auch komplett für Macintosh-Betriebsysteme geschrieben ist und die entsprechende Funktion dort aufzufinden war, konnte man sich daran orientieren und die entsprechende Macintosh-Funktion so umschreiben, dass sie nun auch unter Windows funktionierte.
Eine weitere Stelle im Programm ergab einen Fehler. Ein Funktionsaufruf wurde mit einer falschen Anzahl von Parametern aufgerufen. Dieser wurde dann angeglichen und getestet. Nach diesen Arbeiten konnte nun der Quelltext kompiliert werden. Eine Reihe von auftretenden Assertion-Fehlern konnte in einigen „Debug-Sessions“ behoben werden. Danach begann das Einarbeiten und Einlesen in den Quelltext. Nächstes Ziel war es, die vorhandene Übersteuerung zu beseitigen.

Zunächst musste die Stelle im Quelltext gefunden werden, an der der Input verarbeitet wird und Einstellungen gesetzt werden. Als diese gefunden war, konnte man sich an den vorhandenen Macintosh-Einstellungen orientieren und die Einstellungen für Windows so abändern, dass nun bei weiteren Versuchen die Übersteuerung nicht mehr auftrat. Eventuell war es ein Problem, dass betriebsystemspezifisch ist.
Als nächstes war es nun von Bedeutung, den Quelltext zu verstehen und zu begreifen, was mit dem Input passiert. Wichtig ist hierbei, dass die Verschlüsselung nach dem Koder ansetzt, und dass danach die Pakete auf die Pipe gesetzt werden.
Der Quelltext ist ziemlich übersichtlich angeordnet, obwohl es hin und wieder doch an Kommentaren mangelt. Eine Übersicht über die Dateien, aus denen der Quelltext besteht und aus deren Namen sich meist leicht die jeweilige Funktion ableiten lässt, sieht man in Abbildung 8.

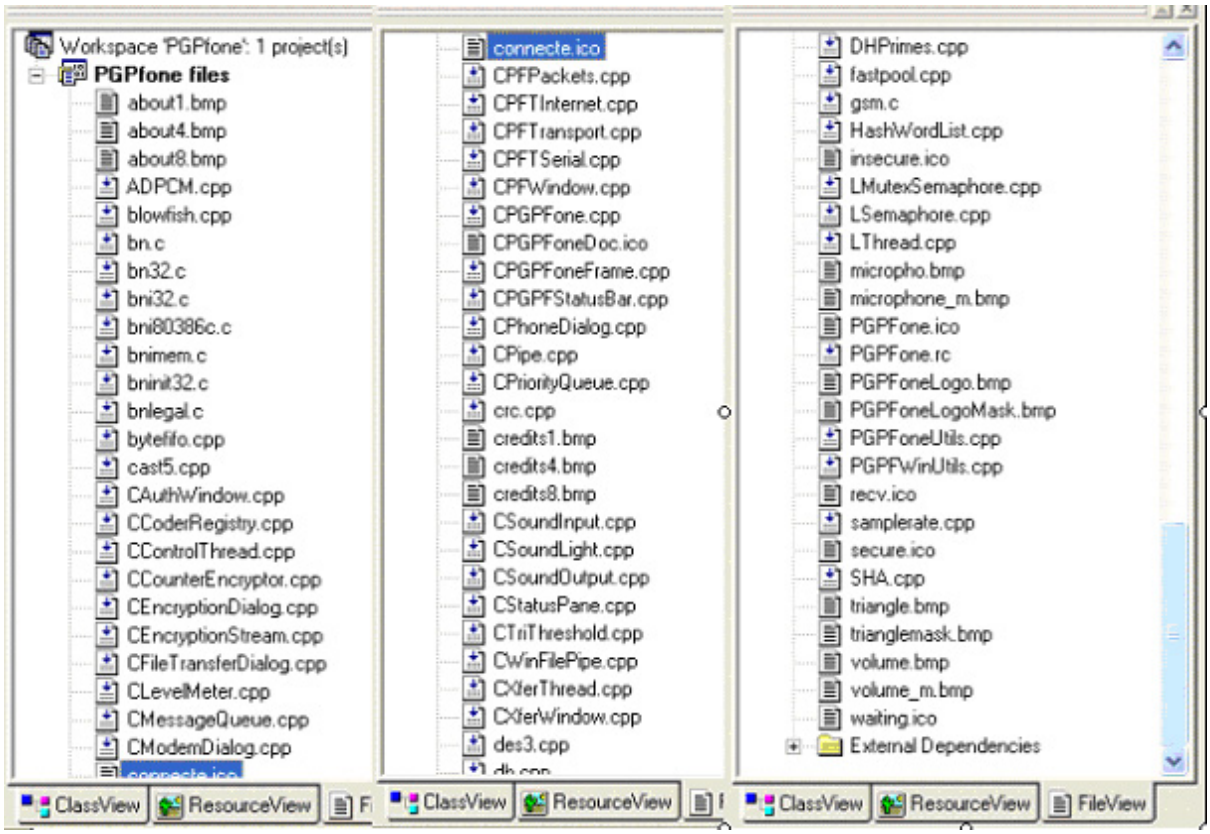

Abbildung 8: Dateiübersicht des Quelltextes

3.6 Was ist NistNet?

Die Rechner wurden nun vom Netz getrennt und nur noch direkt miteinander über einen dritten PC verbunden. Dazu wurden die IP-Adressen des dritten Rechners als Gateway bei den beiden anderen eingetragen. Der dritte Rechner zwischen den beiden anderen bekam also zwei IP-Adressen und sollte zwischen den beiden Netzen von erstens Rechner 3 und Rechner 1 und zweitens Rechner 3 und Rechner 2 das Routing übernehmen.

Dazu wurde NistNet auf dem dritten Rechner installiert. Im Programm NistNet kann man verschiedene Dinge einstellen, wie zum Beispiel einen bestimmten Delay, eine Bandbreite, einen Verlust oder eine Verdoppelung von Paketen. Zusätzlich liefert NistNet Statistiken, so dass man erkennen kann, was für ein Datenverkehr über diesen dritten Rechner abgewickelt wird. NistNet läuft nur unter LINUX.

Zunächst gab es noch einige Probleme, bis feststellt wurde, dass das Routing unter Linux extra einzuschalten ist. Danach funktionierte das Ganze gut. Es konnten so die Verbindungen zwischen den beiden anderen PCs verändert werden. Ein künstlicher Delay konnte eingebracht, oder Paketen konnte eine Verlustrate zugeordnet werden. Bei den ersten Tests sah man genau die erwarteten Veränderungen: Ab einer gewissen Verlustrate war eine Anwahl nicht mehr möglich, und man merkte sofort den eingestellten Delay. Spätere Messungen hierzu folgen. [DOCN]

Unten stehend sieht man in Abbildungen 9 und 10 zwei Sreenshots von NistNet, bei denen man die Funktionen und möglichen Einstellungen erkennen kann. Hier sieht man, wie ein Delay, eine Bandbreite, ein Paketverlust und weitere Einstellungen vorgenommen werden. Im Feld „Source“ und „Dest“ können die jeweiligen IP-Adressen der beiden Rechner, zwischen denen eine Verbindung hergestellt werden sollte, eingegeben werden.

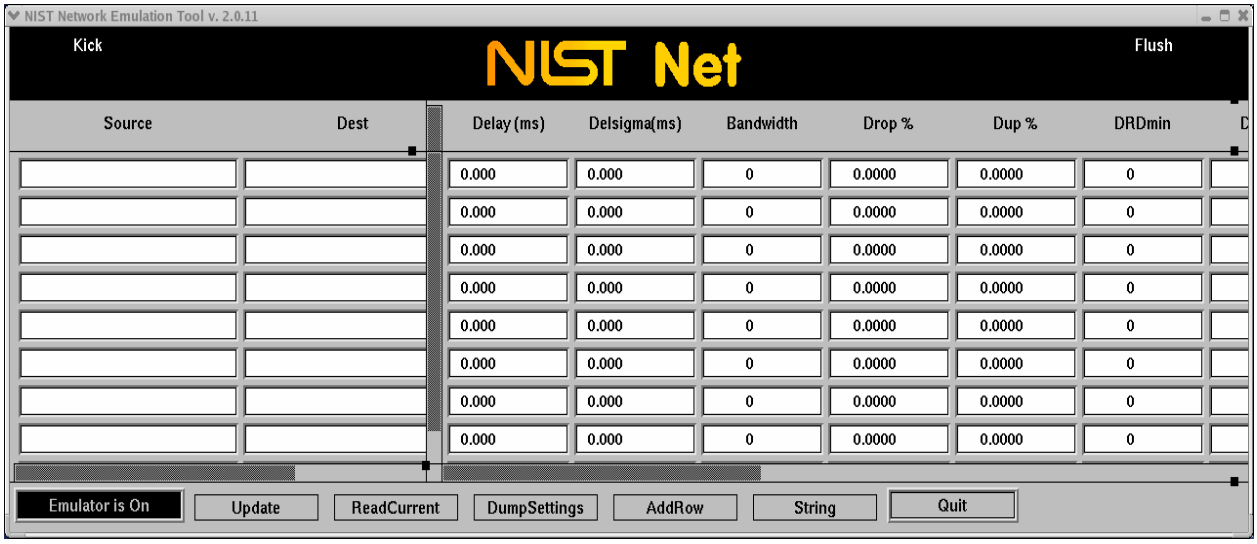

Abbildung 9: Screenshot Nistnet

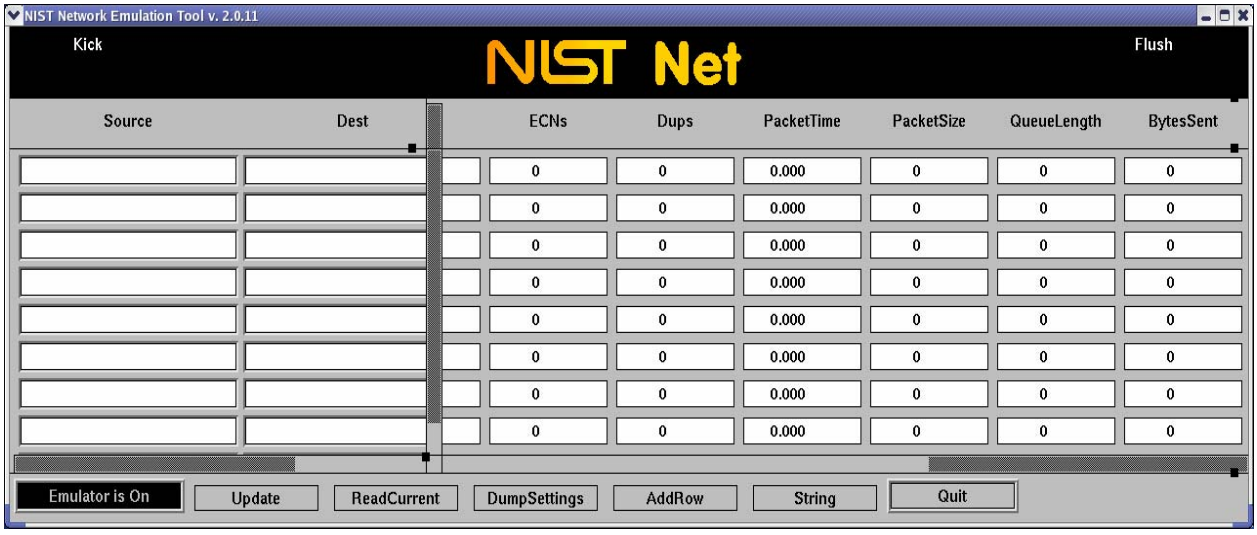

Abbildung 10: Screenshot Nistnet

4. Eigene Implementierungen

4.1 Senden fest definierter Sound-Dateien

Um ein Vergleichen von Tests möglich zu machen, war es notwendig, eine immer gleich bleibende Audiodatei zu übertragen. Nur so können Veränderungen durch verschiedene Einstellungen dokumentiert werden. Da diese Funktion in PGPfone nicht vorgesehen ist, war dies eine sinnvoll durchzuführende Erweiterung. Ziel war es, eine Sounddatei abzuspielen und diese auch ohne weitere Störgeräusche des Mikrofons zu übermitteln. Die Datei sollte möglichst leicht durch andere Audiodateien ausgetauscht werden können, um einer Übertragung mit verschiedenen Dateien zu testen. So wird es dann möglich, das Programm mit verschiedenen Audiodateien zu testen. Das wird zum Beispiel bei der Bestimmung der Grenzfrequenzen angewandt. Dazu mehr im weiteren Verlauf der Arbeit.

Die Ergänzungen wurden im Quelltext vorgenommen, indem ein zusätzlicher Button auf der Oberfläche des Programms erstellt wurde. Beim Drücken dieses Buttons wird nun ein Windows Sound abgespielt. Dieser wird dann übertragen, wenn man vorher den Windows-Input auf WAVE OUT anstatt dem Headset umstellt. Um den Button möglichst auch an das Programm anzugleichen, wechselt er seine Beschriftung, wenn eine Verbindung mit einem Partner besteht und der Sound übermittelt werden kann. In diesem Fall wird dann beim Drücken des Buttons die Datei 3D_Loopy.wav abgespielt, die im entsprechenden Verzeichnis liegen muss. Um das Abspielen der Datei zu ermöglichen, wurden verschiedene Varianten getestet. Am einfachsten ist jedoch die Verwendung der Funktion PlaySound aus der Windows-Bibliothek. Um den Quelltext nicht jedes Mal neu zu kompilieren, wurden im Weiteren eigene Sound Dateien, die übermittelt werden sollen, in das Verzeichnis kopiert und bekamen den Namen 3D_Loopy.wav.

Wie die Erweiterung im Quelltext gelöst wurde, sieht man in Abbildung 11. Es wurden nur relevante Bruchstücke des Quelltextes ausgewählt, die aber die Arbeit ausreichend dokumentieren.

```
const RECT sAudioButtonRect =      {14,0,120,20};
…
#define IDI_AUDIOBUTTON            2021
…
mAudioButton.ShowWindow(SW_SHOW);
…
mAudioButton.SetWindowText("Audio Start");
mAudioButton.ShowWindow(SW_SHOW);
case _cs_disconnecting:
…
mAudioButton.SetWindowText("Audio Start");
…
mAudioButton.Create("Audio",WS_CHILD|WS_VISIBLE|WS_TABSTOP,
sAudioButtonRect,this,IDI_AUDIOBUTTON);
mAudioButton.SendMessage(WM_SETFONT, (ulong)(HFONT)mPBFont,
TRUE);
…
case IDI_AUDIOBUTTON:
{
     char s[256];
     if((wParam >> 16) == BN_CLICKED)
     PlaySound("3D_Loopy.wav",NULL,SND_ASYNC|SND_FILENAME);
}
break;
}
…
```

Abbildung 11: Quellcode um Audiodateien zu senden

4.2 Bedeutungen von Statistiken

Bei PGPfone werden einige Statistiken erfasst, und diese können, wenn dies erwünscht ist, ausgegeben werden. Um sich die Statistiken anzeigen zu lassen, muss man im Menü Ansicht in den Expertenmodus wechseln.
Zunächst gibt es eine Statistik über gesendete Pakete, hier wird jeweils beim endgültigen Senden eines Paketes die Anzahl um eins erhöht. Weiter gibt es eine Statistik über „Good und Bad Receives Packets“. Über eine Funktion „GotPacket“ wird jedes Paket, das empfangen wird, eine dieser beiden Statistiken zugeordnet. „Good Received“ ist das Paket, wenn kein Fehler auftritt und die Decodierung und Entschlüsselung problemlos funktioniert. Dies sollte der Normalfall sein und ist es in der Praxis auch. „Bad Received“ ist das Paket, wenn beim Empfangen ein Fehler auftritt. Ein Fehler ist zum Beispiel, wenn das Paket zu alt, das heißt seine Sequenznummer zu klein ist. Es wird also immer eine aktuelle Spanne berechnet, die angibt, welche Pakete ankommen dürfen. Ist ein Paket älter, als die Spanne dies zulässt, wird es verworfen und der Statistik Bad Received zugezählt. Ein anderer möglicher Fehler tritt auf, wenn das Paket nicht korrekt decodiert werden kann, dass heißt, es wurde zum Beispiel ein Koder verwendet, dessen Dekoder nicht zur Verfügung steht.

Auch bei etlichen weiteren Fehlern, wie zum Beispiel, wenn die Prüfsumme des Headers falsch ist, wird die Statistik für Bad Received um eines erhöht. In der Praxis sieht es so aus, dass man kaum schlecht empfangene Pakete hat und beim Laufen des Programms sehr lange auf solch ein Paket warten muss. In den getesteten Fällen lag die Anzahl von schlecht empfangenen Paketen meistens unter 0,1 %.

Die als nächstes angegebene Statistik ist der „Round Trip“. Er misst die Zeit, die vergeht, bis ein entsprechendes Paket an- und wieder zurückkommt. Es ist also die doppelte Strecke, die hier gemessen wird. Nähere Messungen zum Delay, also der einfachen Strecke, wurden später noch hinzugefügt, doch dies wird an späterer Stelle in dieser Arbeit näher erläutert. Die Zeit wird jedoch erst gemessen, wenn der Input verarbeitet und komplett zu Paketen geformt wurde. Das heißt, der Koder und die Verschlüsselung spielen bei dieser Messung keine Rolle, im Gegensatz zu späteren selbst versuchten Delay-Messungen, wie sie am Ende erläutert werden. In der Praxis ist deshalb dieser Wert des RoundTrip auch meistens bei 0 ms und steigt nur bei sehr intensivem Reden von beiden Seiten aus leicht an. Dabei ist als Ursache die Wartezeit der Pakete in der Warteschlange anzusehen. Wenn sehr viel gesprochen und empfangen wird, können nicht alle Pakete sofort gesendet werden. Dazu wurde in PGPfone eine Warteschlange erstellt. Bis die Pakete gesendet werden können, kommt es dadurch zu einer Verzögerung, die dann zum Round Trip hinzugezählt wird. Bei sehr vielen Paketen kann es einen Round Trip von bis zu 60 ms geben. Das ist jedoch ein Ausnahmefall und wird beim normalen Telefonieren nicht erreicht.

GSMload ist eine Statistik, die Aufschluss darüber gibt, wie viel Zeit der Koder bzw. der Dekoder jeweils benötigt. Hier sind die Werte allein vom gewählten Koder abhängig, unterscheiden sich jedoch kaum. Bei GSM bekommt man einen etwas höheren Wert, weil es auch die beste Codierung ist, ADPCM liegt in der Mitte und GSM Lite ist die schnellste Codierung.

„Samples Queued“ gibt Auskunft darüber, wie lange die schon oben angesprochene Warteschleife der Pakete ist. Sie wird mit der Funktion „GetNumPlaying“ abgefragt. In der Praxis ist die Anzahl der Pakete in der Warteschleife solange 0, bis man laut und schnell anfängt zu sprechen. In diesem Fall kann die Anzahl der Pakete in der Warteschlange dann schnell auf einige Hundert, sogar mehr als Tausend anwachsen. Wenn man nun die Warteschlange im Vergleich zum Round Trip betrachtet, ergibt sich ein direkter Zusammenhang, wie auch schon bei der Funktion des Round Trip erläutert wurde.

Die Statistik Jitter ist eine Varianz, die hier von PGPfone berechnet wird. Zunächst wird mit pgp_milliseconds ein Zeitstempel gesetzt und mit baskew der Unterschied der Zeitstempel ermittelt. Die Berechnung des Jitters erfolgt dann abhängig davon. Wie die Berechnung und ihre Formeln genau aussehen, sieht man in Abbildung 12, in der ein Ausschnitt der Berechnung, der das Prinzip der Vorgehensweise erläutern soll, aufgeführt ist.

```
…
dlen-=4;
*len = (short) dlen;
pgp_memcpy(*buffer, mInputBuffer + 7, dlen);
BUFFER_TO_LONG(mInputBuffer + 3, timestamp);
timestampnow = pgp_milliseconds();
baskew = timestampnow-timestamp;
if(mJitterCount > 3)
     sSkew += (baskew-sSkew)/8;
else
     sSkew = baskew;
baavg = labs(baskew-sSkew);
sJVariance += (baavg-sJVariance)/16;
sJitter = sJVariance * 2;
…
```

Abbildung 12: Berechnung des Jitters

Auch die Bandbreite als letzten Wert zu übermitteln, ist in PGPfone angedacht. Der Wert wird zunächst immer auf 0 gesetzt. Bei jedem Aufruf wird nun der Wert, falls sich die Bandbreite ändert, aktualisiert und neu angepasst. Aufgrund der Bandbreite wird bei PGPfone die MTU geregelt, also die maximale Länge der Pakete. Sie wird im Wesentlichen notwendig, falls ein Modem verwendet und die Bandbreite extrem eingeschränkt wird. Bei der Einstellung Internet wird die Bandbreite sonst nicht weiter benutzt und der Wert bei Tests war immer 0. Deshalb spielt diese Statistik im Verlauf der Arbeit keine weitere Rolle. Die vom PGPfone bei einer Übertragung verwendete Bandbreite wird später noch durch einen Versuch ermittelt.

4.3 Abspeichern der Statistiken

Um besser die Veränderungen der mitgelieferten Statistiken zu erkennen, wurde nun das Programm so erweitert, dass alle vorhandenen Statistiken jeweils in eine Textdatei (.txt) geschrieben wurden. Das ist sinnvoll, da Veränderungen im laufenden Programm damit gesehen und besser ausgewertet werden können. So ist es zum Beispiel möglich, die Übermittlung von Daten zu ändern oder auch den Koder zu wechseln und anschließend die Ergebnisse, wie sich die Statistik-Werte entwickeln, anhand der Textdatei anzusehen. Die Textdateien werden in dem Ordner erstellt, in dem sich auch das Programm befindet. In diesem Fall war es der Ordner Debug und der Ordner andere_Seite. Um nicht nur auf die Textdatei angewiesen zu sein, sondern die Ergebnisse visuell sehen zu können, wurde nach Programmen gesucht, die die Textdatei, möglichst sogar mehrere von ihnen, gleichzeitig in einem Diagramm darstellen können und so Auswertungen noch einfacher machen.
Dies lässt sich mit Microsoft Excel umsetzen. In einem ersten Versuch mit Excel wurde festgestellt, dass die dort zur Verfügung gestellten Funktionen ausreichen, weshalb das Programm im Weiteren zum Erstellen von Diagrammen verwendet wurde. Hier kann man die entsprechenden Textdateien importieren und sich alle importierten Dateien auch als Diagramm darstellen lassen.

Die Erweiterungen im Quelltext, um nun die Statistik-Werte in eine Textdatei zu schreiben, finden sich in Abbildung 13. Auch hier sind nur Bruchstücke des Quelltextes aufgeführt.

```
                // erstellen von .txt-Dateien
     ofstream dataFile1("Psend.txt");      //gesendete Pakete
     ofstream dataFile2("GoodP.txt");      // fehlerfreie Pakete
     ofstream dataFile3("BadP.txt");       // fehlerhafte Pakete
     ofstream dataFile4("GSM.txt");        // Zeit, die Koder
                                           einnimmt
     ofstream dataFile5("RoundT.txt");     // Round Trip
     ofstream dataFile6("Underf.txt");     // Sound Underflows
     ofstream dataFile7("Overf.txt");      // Sound Overflows
     ofstream dataFile8("Jitter.txt");     // Jitter
     ofstream dataFile9("Bandw.txt");      // Bandbreite
     ofstream dataFile0("Queued.txt");     // Länge der
                                           Warteschleife
               …
               InterlockedExchange(&mDirty, 0);
               _itoa(mRTTms, t, 10);
               sprintf(s, "%d\r%d\r%d\r%sms\r%d%%",
                    mPacketsSent, mGoodPackets, mBadPackets,
                    t, mGSMfull);
               …
               dataFile1 << mPacketsSent << endl;
                                 //einlesen der Werte
                                 in die Datei
               dataFile5 << t << endl;
               dataFile2 << mGoodPackets << endl;
               dataFile3 << mBadPackets << endl;
               dataFile4 << mGSMfull << endl;
               …
               mInfo1.SetWindowText(s);
               _itoa(mSoundOutput->GetNumPlaying(), t, 10);
               sprintf(s, "%s\r%d\r%d\r%dms\r%d",
                    t, mOutputUnderflows, mOverflows, mJitter,
                    mBandwidth);


               …
               dataFile6 << mOutputUnderflows << endl;
               dataFile7 << mOverflows << endl;
               dataFile8 << mJitter << endl;
               dataFile9 << mBandwidth << endl;
               dataFile0 << t << endl;
               …
```

Abbildung 13: Quelltext um Statistikwerte zu speichern

Eine Zuordnungstabelle mit den jeweiligen Statistiken und den Dateien, in die sie gespeichert wurden, ist in Abbildung 14 zu sehen.

Beschreibung der Statistik	Name der Statistik in PGPfone	Dateiname der gespeicherten Statistiken
Gesendete Pakete	Packets Sent	Psend.txt
Gut erhaltenen Pakete	Good Rcvd	GoodP.txt
Fehlerhafte Pakete	Bad Rcvd	BadP.txt
Zeit dir Koder einnimmt	GSM Load	GSM.txt
Round Trip	Round Trip	RoundT.txt
Länge der Warteschlange	Samples Queued	Queued.txt
Sound Underflows	Sound Underflows	Underf.txt
Sound Overflows	Sound Overflows	Overf.txt
Jitter	Jitter	Jitter.txt
Bandbreite	Est. Bandwidth	Bandw.txt

Abbildung 14: Zuordnungstabelle der einzelnen Statistikwerte

Aus den Daten lassen sich nun Diagramme mit Hilfe von Microsoft Excel darstellen. Hier ein Beispielsdiagramm, um zu sehen, wie eine Auswertung in Form eines Diagramms aussehen kann.
Bei diesem Test wurde versucht, die Werte möglichst stark nach oben zu treiben, in dem von beiden Seiten aus viele Pakete gesendet wurden. Das Ergebnis lässt sich in Abbildung 15 erkennen.

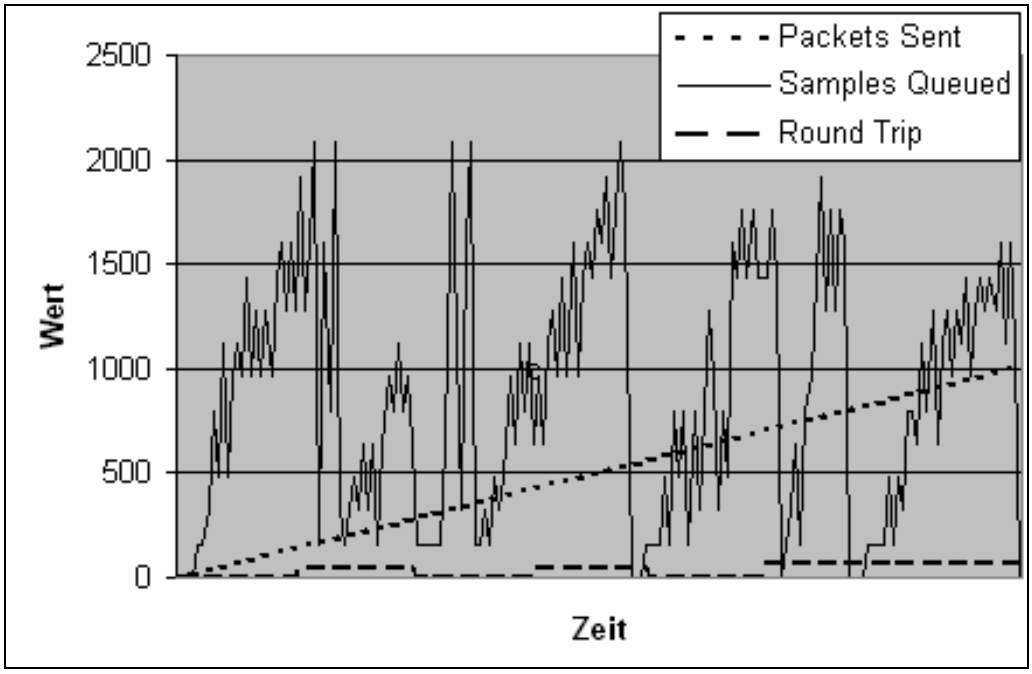

Abbildung 15: Diagramm der Statistiken

4.4 Überlegungen zur Delay-Messung

Im Vordergrund stand zunächst eine passende Synchronisation der Uhren. Es musste nun nach Tools gesucht werden, die eine Synchronisation unter Windows möglich machen. Ein Rechner muss dabei als Server dem anderen die Uhrzeit freigeben. Um die Windows-Uhrzeit über das Internet anzupassen, gibt es unzählige Tools. Jedoch ist die Genauigkeit dieser Tools zwar für die Uhrzeit im normalen Gebrauch ausreichend. Da für die Anwendung, wie sie hier angedacht ist, eine große Genauigkeit, möglichst auf Millisekunden, erreicht werden sollte, musste weiter nach anderen Möglichkeiten gesucht werden.

Wenn die Synchronisation der Uhren beider Systeme gelungen ist, muss man eine Möglichkeit finden, einen präzisen Zeitstempel zu setzen. Dabei müssen genaue sinnvolle Stellen im Quelltext gefunden werden, damit der Zeitstempel auch eine entsprechende Aussagekraft besitzt. Nach erfolgreichem Setzen, wäre es sinnvoll, den Zeitstempel auch gleich in einer Textdatei festzuhalten, um ihn anschließend besser vergleichen und auswerten zu können. Außerdem sollte PGPfone dabei den Unterschied der einzelnen Zeitstempel in die Textdatei mit hineinschreiben, so dass eine spätere Varianzberechnung leichter möglich ist. Auch sollte, wenn möglich, die Paketnummer mit ausgegeben werden, damit man eindeutig sieht, welches Paket wann ankommt. Die Ausgaben und Messungen müssen natürlich auf beiden Seiten des Programms durchgeführt werden.

4.5 Uhrsynchronisation

Von den unterschiedliche Tools, die frei zur Verfügung stehen, und die sich durch unterschiedliche Genauigkeiten unterscheiden, hat sich beim Testen der Tools „AboutTime“ hervorgetan. Die Bedienung ist sehr einfach, die Abweichung beim Stellen der Uhr liegt zwischen 0ms und 10 ms. Sehr wichtig war, dass die Abweichung, um die die Uhr jeweils korrigiert wird, in ms angezeigt wird. So kann also nun die Uhr synchronisiert, eine Messung vorgenommen und anschließend festgestellt werden, wie groß die Abweichung war. So war es möglich, nur Messungen auszuwerten, wenn eine Abweichung von kleiner als 2 ms bestand.

Das Programm funktioniert so, dass unter Time Hosts ein NTP-Server eingetragen werden kann, von dem die Zeit aktualisiert wird. Hier sind auch mehrere Eingaben möglich, wobei in diesem Fall nur die Zeit von dem anderen Rechner synchronisiert werden durfte. Weitere Optionen, in welcher Zeitspanne die Aktualisierungen vorgenommen werden sollen, bietet das Programm auch an. Diese wurden aber nicht verwendet, sondern die Uhrzeit immer direkt vor einer Messung synchronisiert.

Einen Screenshot des Programms sieht man in Abbildung 16.

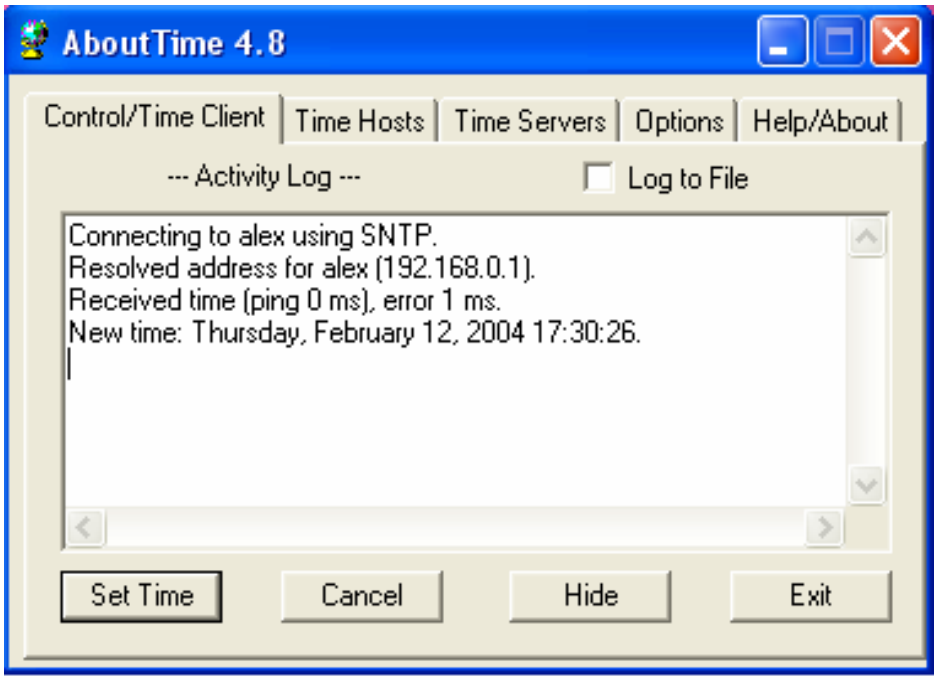

Abbildung 16: Screenshot AboutTime

4.6 Setzen von Zeitstempeln

Nun ging es daran, an vielen verschiedenen Stellen im Programm Zeitstempel zu setzen, so dass der Weg vom Input zum Output möglichst genau nachvollzogen werden kann. Ziel hierbei ist es herauszufinden, welche Teile des Programms wie viel Zeit in Anspruch nehmen, wie viel Zeit die Verschlüsselung kostet und wie groß der Unterschied zwischen den verschiedenen Verschlüsselungsalgorithmen ist.

PGPfone wurde dabei um eine Funktion erweitert, die für das Messen der Zeit zuständig ist. Sie schreibt jeweils die aktuelle Zeit in eine Textdatei und berechnet sofort die vergangene Zeit zum vorherigen Setzen des Zeitstempels. Beim Senden und Empfangen der Pakete wird gleichzeitig noch die Paketnummer ausgegeben.

Die kleinste Auflösung der Windows-Systemzeit sind Millisekunden, da es aber gleich um eine Größenordnung von etwa 50 ms gehen wird, ist es nicht weiter wichtig, Zeitstempel in Mikrosekunden zu bekommen. Um den Zeitstempel in Millisekunden zu erhalten, kann direkt auf die Windows-Systemzeit zugegriffen werden. Zeitstempel wurden an folgenden Stellen gesetzt: Zuerst beim Verarbeiten der Daten und vor der Übergabe an den Koder, nach Ende des Koders, beim Paket schicken (also dem wirklichen Setzen auf die Pipe), beim Empfangen eines Paketes und vor der anschließenden Ausgabe des Paketes nach dem Dekodieren.

Die Funktion zum Setzen der Zeitstempel ist in den Abbildungen 17.1, 17.2 und 17.3 dargestellt. Der Quelltext enthält auch weitere Zeitstempel, die gesetzt wurden, um den Quelltext und die einzelnen Funktionsaufrufe besser zu durchschauen. Es sind nur Ausschnitte des Quelltextes zu sehen.

```
…
            ofstream dataFile12("Koder_Anfang.txt");
            ofstream dataFile13("Koder_Ende.txt");
            ofstream dataFile14("Paket_empfangen.txt");
            ofstream dataFile15("Paket_schicken.txt");
            ofstream dataFile16("Thread_Anfang.txt");
            ofstream dataFile17("Thread_Ende.txt");
            ofstream dataFile20("stream_send.txt");
            ofstream dataFile21("stream_get.txt");
            ofstream dataFile99("gesamtDelay.txt");
            ofstream dataFile88("ack.txt");
…
```

Abbildung 17.1: Anlegen der Dateien für die Zeitstempel

Stelle des Zeitstempels	**Dateiname**
Nach dem Input, noch vor dem Koder	Koder_Anfang.txt
Nach Ende des Koders	Koder_Ende.txt
Vor Setzen des Paketes auf die Pipe	Paket_schicken.txt
Beim Empfangen des Paketes auf der anderen Seite	Pakete_empfangen.txt
Ein Tests um den Quellcode besser zu durchschauen, im Wesentlichen um festzustellen, wann und wie oft einzelne Threads aufgerufen wurde	Thread_Anfang.txt Thread_Ende.txt stream_send.txt Stream_get.txt
Nach Verarbeitung des Paketes vor Übergabe an die Soundkarte	gesamtDelay.txt
Zum Feststellen, wann „Acknowledge-Pakete“ gesendet werden	ack.txt

Abbildung 17.2: Zuordnungstabelle der einzelnen Zeitstempel

```
void
CSoundInput::Zeit(int i, int seq)
{
                  int start;
                  // verschiedene Tests die Zeit zu messen
                  //int a = timeGetTime();
                  //start = clock();
                  //start = timeGetTime();
                  //start=GetTickCount();
                  //start = GetCurrentTimeMillis();

                  SYSTEMTIME st;           // Systemzeit in struct holen
                  GetSystemTime(&st);

// auslesen der aktuellen, vorher synchronisierten Systemzeit mit einem
// Wert, der die Zeit bis zu den Stunden berücksichtigt

// umrechnen der Zeiten, Berücksichtigung bis zu Stunden
start = st.wMilliseconds + st.wSecond * 1000 + st.wMinute * 60000 +
st.wHour * 3600000;
switch(i)
                  // Unterscheidung der Aufrufe und Zuordnung der Dateien
            {
            case 1:
                  dataFile12 << " Zeitunterschied:  " << (start -
                  letzter1) << " absolute Zeit:   " << start << "
                  Übergabe an Koder " << endl;
                  letzter1 = start;
                  break;
            case 2:
                  dataFile13 << " Zeitunterschied:  " << (start -
                  letzter2) << " absolute Zeit:   " << start << "
                  alles gecodet " << endl;
                  letzter2 = start;
                  break;
            case 3:
                  dataFile15 << " Zeitunterschied:  " << (start -
                  letzter3)  << " absolute Zeit:  " << start << "
                  Packet geht raus mit der Nummer:  "   << seq << endl;
                  letzter3 = start;
                  break;
…
            case 99:
                  dataFile99 << " Zeitunterschied:  " << (start -
                  letzter99)  << " absolute Zeit:  " << start << endl;
                  letzter99 = start;
                  break;
            case 88:
                  dataFile88 << " Zeitunterschied:  " << (start -
                  letzter88)  << " absolute Zeit:  " << start << "
                  Nummer:   " << seq << endl;
                  letzter88 = start;
                  break;
            }
}
…
```

Abbildung 17.3: Quellcode zum Setzen der Zeitstempel

Ein Aufruf an den unterschiedlichen Stellen im Programm ist in Abbildung 18 dargestellt. Auch hier sieht man nur einen Funktionsaufruf, die anderen erfolgen analog.

```
// Der Aufruf der Zeitstempel-Funktion an unterschiedlichen
// Messpunkten
…
     CSoundInput    *mSoundInput;  // im Header der Datei
…
          mSoundInput->Zeit(4, *seq);
          // hier wird die Sequenznummer mit übergeben
…
```

Abbildung 18: Aufruf der Funktion zum Setzen von Zeitstempeln

Wichtig für die Delay-Messung ist erstens die Textdatei, in die sofort nach dem Input hineingeschrieben wird, und zweitens, diejenige, die kurz vor dem Output auf der anderen Seite erzeugt wird. Der Vergleich dieser beiden jeweiligen Werte liefert den Delay, zur besseren Veranschaulichung sieht man in den Abbildungen 19 und 20 die jeweilig ausgegebenen Werte beim Setzen der Zeitstempel:

Anfang Input vor der Übergabe an den Koder:

Zeitunterschied	Absolute Zeit
78	36343012
63	36343075
78	36343153
63	36343216
140	36343356
79	36343435
62	36343497
77	36343574

Abbildung 19: Zeitstempel am Anfang der Verarbeitung auf Senderseite

Ende Output vor der Übergabe an die Soundkarte:

Zeitunterschied	Absolute Zeit
62	36343074
78	36343152
63	36343215
78	36343293
140	36343433
63	36343496
78	36343574
63	36343637

Abbildung 20: Zeitstempel am Ende der Verarbeitung auf Empfängerseite

Aus den beiden oberen Kästen ergeben sich nun die Delay-Messungen, wie sie in Abbildung 21 zu sehen sind. Die Messungen werden hier nur von den Werten, die oben aufgelistet wurden, berechnet. Um die späteren Auswertungen vorzunehmen, wurden wesentlich mehr Werte betrachtet, von denen dann wiederum der Durchschnittswert gebildet wurde.

Delay von:

62
77
62
77
77
61
77
63

Abbildung 21: Auswertung der Delays

5. Analysen

5.1 Erstellen und Auswerten von Signalen

Nun standen einige Überlegungen an, wie man wohl getestete Ergebnisse mit dieser einen Audiodatei, die nun übertragen werden kann, visualisieren könnte. Am besten ist dies wohl mit Aufzeichnung der Signale möglich. Man könnte so das Originalsignal mit dem am anderen Ende aufgezeichneten Signal vergleichen und daraus Schlüsse über Änderungen ziehen, um damit Aussagen über die Qualität des Programms treffen zu können.

Hierzu wurden dann einige Tools getestet, mit denen man in der Lage ist, Signale der vordefinierten Audiodatei aufzuzeichnen und am Besten gleich miteinander zu vergleichen. Am Besten gefiel das Programm Audacity, welches als Freeware erhältlich ist. Hier können Signale aufgezeichnet werden. Man hat die Möglichkeit, diese untereinander zu vergleichen, hinein oder heraus zu zoomen und die Zeitachse zu verschieben. Das Tool erschien als sehr geeignet für die Durchführung der Analysen. Des Weiteren gibt es eine Funktion, bei der selbst Töne erstellt werden können, was später noch großen Nutzen hatte. [HIAUD]

Einen Screenshot, wie dieses Programm aussieht und arbeitet, kann man in Abbildung 22 sehen:

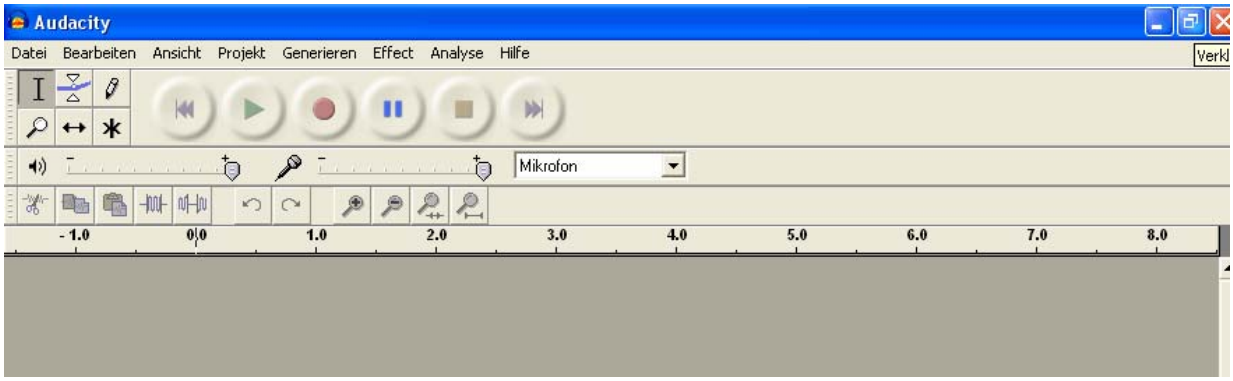

Abbildung 22: Screenshot Audacity

Nach dem problemlosen Installieren dieses Programms wurde nun eine ganze Reihe Tests durchgeführt. Es wurde zunächst die Originaldatei abgespielt und dabei von Audacity aufgezeichnet. Nun wurde die Audiodatei 3D_Loopy.wav immer wieder mit verschiedenen Einstellungen der Koder, mit oder ohne Verschlüsselung und verschiedenen Arten der Verschlüsselung, getestet und am anderen Ende mit Audacity die ankommenden Signale erneut aufgezeichnet. So hatte man die verschiedenen Audiosignale direkt unter der Originaldatei und konnte so Unterschiede untersuchen.

Mit den aufgezeichneten Signalen konnten nun Auswertungen vorgenommen werden. Es gab doch einige sichtbare Unterschiede zwischen den einzelnen Kodern und Dekodern. Diese wurden nun genauer untersucht.

Einen Überblick über die verwendeten Ausschnitte der Signale sieht man in Abbildung 20. Hier kann man erkennen wie sich die Frequenzbänder mit Hilfe von Audacity darstellen lassen. Um Auswertungen durchführen zu können, muss man näher an die Bänder heranzoomen, wie das in Abbildung 23 gezeigt wird.

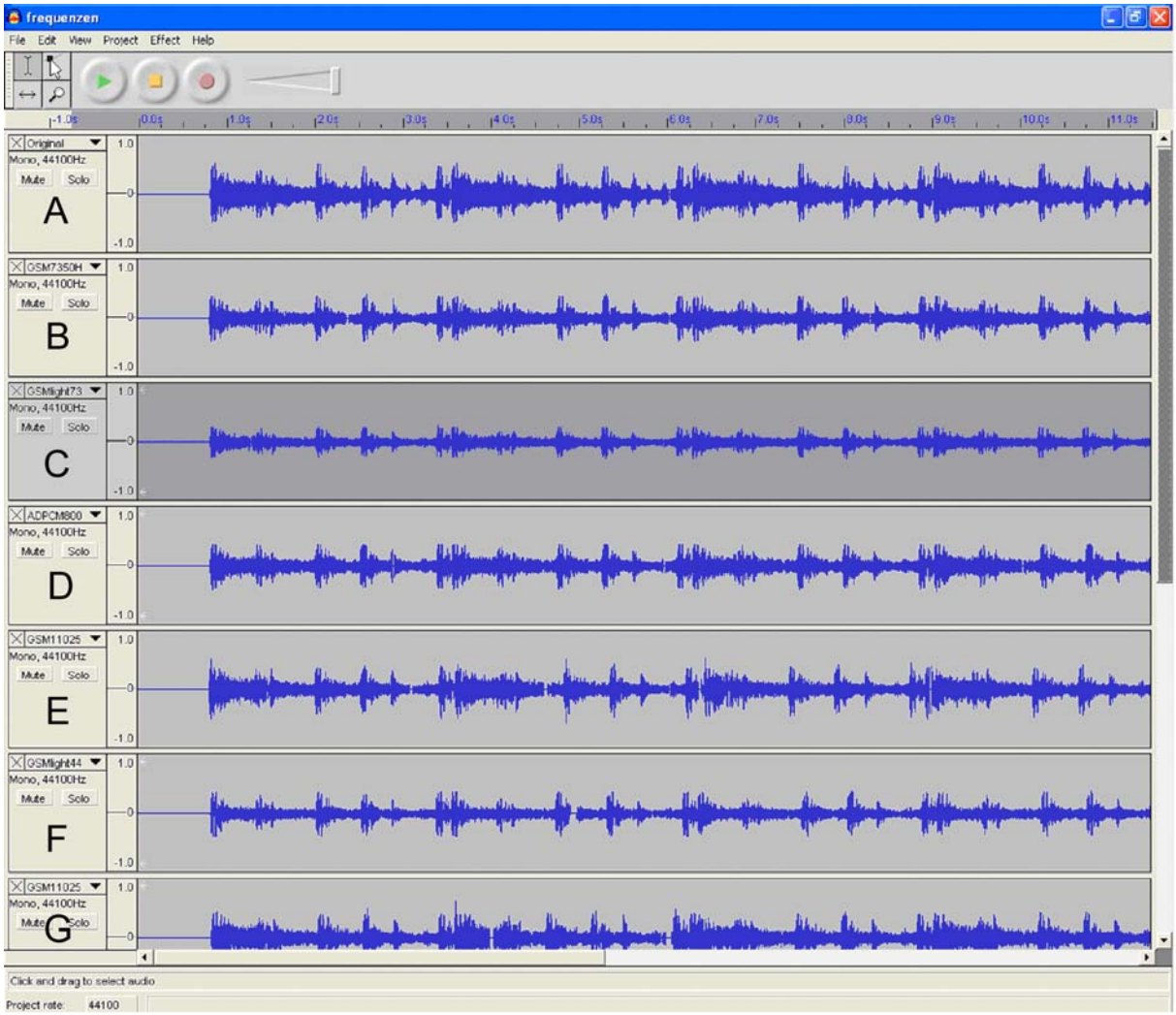

Abbildung 23: Überblick der Signale

Zum näheren Untersuchen der Signale lässt sich, wie auch bei der Programmbeschreibung von Audacity bereits beschrieben, beliebig nahe an die einzelnen Aufzeichnungen heranzoomen. Eine Zoomeinstellung, bei der sinnvolle Auswertungen der einzelnen Signale vorgenommen werden konnten, kann man in Abbildung 24 erkennen. Bei dem ersten Signal handelt es sich um das Original (A), es folgen GSM 7350 HZ (B), GSM Lite 7350 HZ (C), ADPCM 8000 HZ (D), GSM 11025 HZ (E), GSM Lite 4410 HZ(F), bei denen jeweils die Verschlüsselung CAST eingesetzt wurde und ganz unten die Übertragung mit GSM 11025 HZ ohne das Wählen einer Verschlüsselung (G).

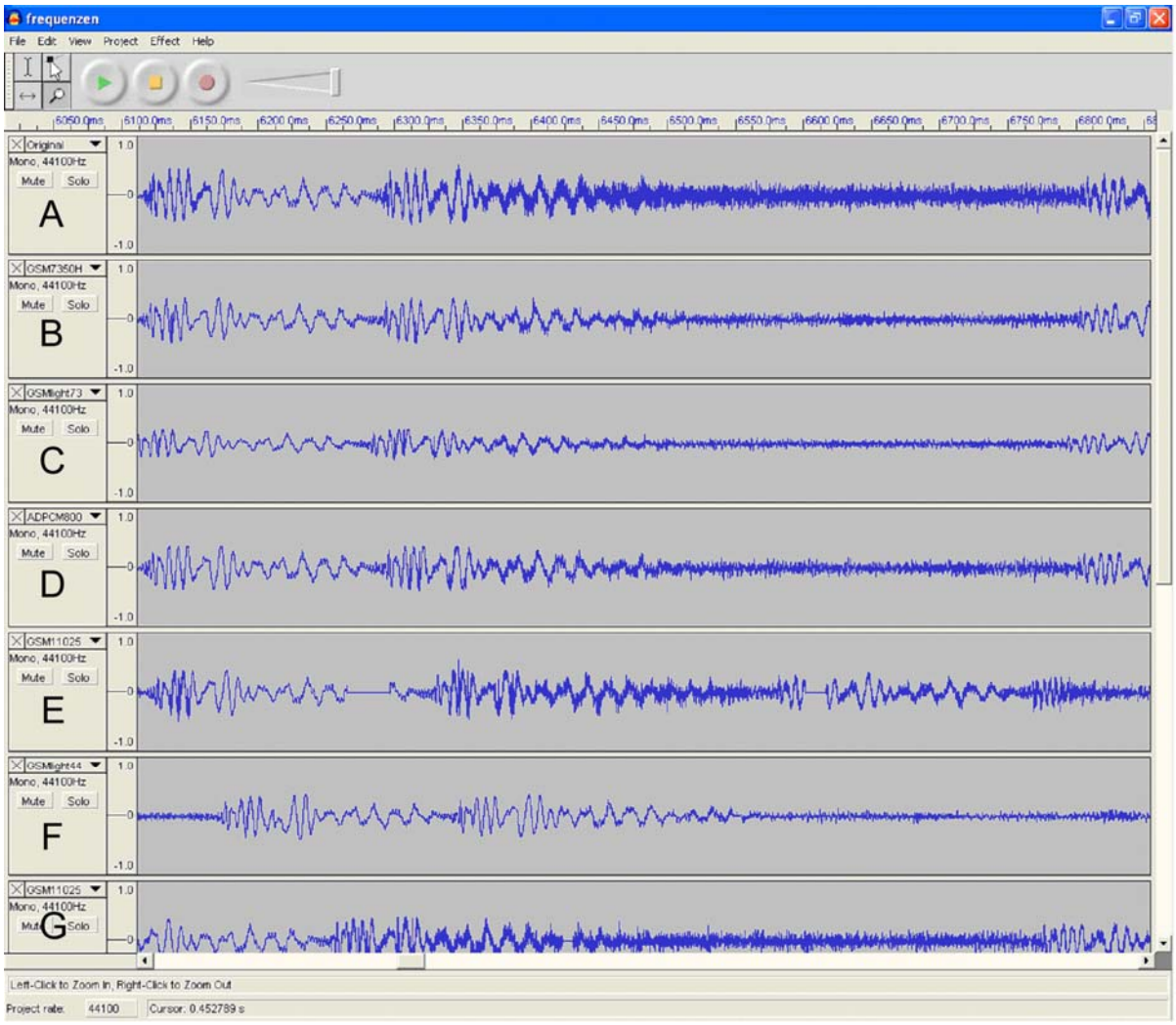

Abbildung 24: Zoom der Signalaufzeichnungen

Auswertungen zu den einzelnen Bändern wurden nun in Textform gebracht und untenstehend beschrieben. Zu beachten ist dabei, dass die oben stehende Grafik nur einen sehr kleinen Ausschnitt zeigt, die vorgenommenen Auswertungen sich jedoch über den gesamten Bereich erstrecken, so dass nicht alle unten stehenden Auswertungen auf dem obigen Bild zu erkennen sind. Der Vergleich bezieht sich hierbei immer auf die Originaldatei. Zunächst werden die Unterschiede der einzelnen Koder näher betrachtet. Alle Tests wurden hier mit der Verschlüsselung CAST getestet. Weitere Tests mit verschiedenen Verschlüsselungen folgen.

GSM 7350 HZ:

Bei diesem Koder lässt sich erkennen, dass das Signal insgesamt gröber wird. Man sieht auch, dass kleine schmale Peaks vollständig verschwinden. Es gibt kurze Aussetzer. An manchen Stellen hat man den Eindruck, als ob das Programm mit dem Codieren, dem Verschlüsseln und Versenden nicht zeitnah übermitteln kann. Man sieht, dass nach einem Aussetzer die Daten meist doch noch übermittelt werden und dann komprimiert innerhalb einer kürzeren Zeit auf der Aufzeichnung des Signals erscheinen. Kleine Amplituden verschwinden häufiger, und man kann erkennen, dass sehr hohe und tiefe Werte weniger extrem hoch bzw. tief übertragen werden.

GSM Lite 7350 HZ:

Im Unterschied zu GSM 7350 HZ ist die zur Verfügung stehende Version GSM Lite laut Anleitung von PGPfone deutlich schneller. Dies merkt man auch daran, dass hier so gut wie keine Aussetzer mehr vorhanden sind. Deutlicher als bisher ist zu sehen, dass nur gering unterschiedliche Werte stärker als bei GSM auf einen gleichen Wert gerundet werden. Die Struktur wird damit etwas gröber als beim Verfahren von GSM.

ADPCM 8000 HZ:

Bei dieser Methode gibt es mehr Aussetzer als bei den bisherigen Verfahren. Insgesamt kann man sagen, dass ADPCM eine gröbere Struktur als GSM besitzt, aber fast besser als GSM Lite ist. Die Rundungen von bestimmten Werten sind auch deutlich größer, als dies bei GSM der Fall ist. Peaks werden durch diesen Koder deutlich abgeschwächt.

GSM 11025 HZ:

Das Signal wird bei diesem Koder wohl am besten angenähert. Negativ zu sehen ist, dass es bei dieser Methode sehr viele Aussetzer gibt. Zum Teil sind nach diesen Aussetzern große Zeitverschiebungen sowohl nach vorne als auch nach hinten zu erkennen. Es kommt auch vor, dass komplette Teile fehlen und einfach durch einen waagrechten Strich bei der Signalaufzeichnung dargestellt werden. Diese Methode kostet wohl am meisten Zeit, da sie mit 11025 HZ die Genauste ist. Das ist wohl auch der Grund, warum das Codieren oft nicht richtig nachkommt und es zu besagten Aussetzern kommt. An den Stellen, an denen es keine Aussetzer gibt, wird das Ausgangssignal sehr gut angenähert, was oben im Signal zu erkennen ist.

GSM Lite 4410 HZ:

Hier sieht man deutlich, dass bei diesem Koder mit einer ziemlich groben und eher schlechten Abtastung gearbeitet wird. Peaks werden zum Teil vollständig verschluckt und kaum noch vernünftig dargestellt. Aussetzer gibt es wie erwartet bei dieser Methode keine. Die Codierung ist mit Abstand die schnellste, was auch in der Anleitung von PGPfone zu lesen ist. Also kann diese Codierung, bei der es auf Geschwindigkeit und weniger auf Qualität ankommt, sinnvoll sein.

Im folgender Abbildung 25 sind noch mal drei der oberen Signalaufzeichnungen genauer gezeigt, damit man sich ein Bild davon machen kann, wie Auswertungen vorgenommen wurden. Es handelt sich um die Audiosignale von der wohl besten Codierung GSM 11025 HZ (B) und darunter die wohl schlechteste GSM Lite 4410 HZ (C), zum Vergleich ganz oben die Originaldatei (A).

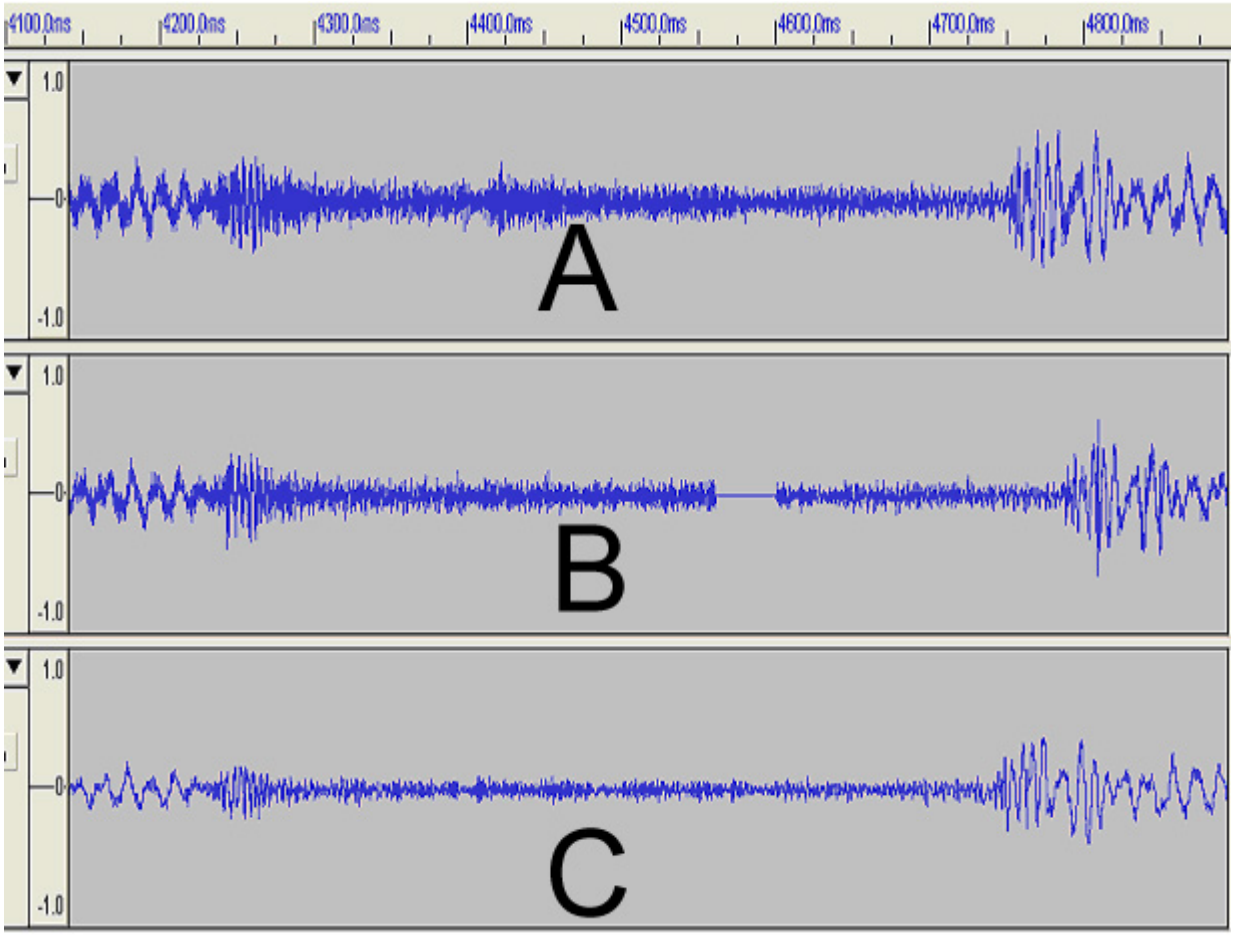

Abbildung 25:Nahaufnahme der Originaldatei gefolgt von GSM 11025 HZ gefolgt von GSM Lite 4410 HZ

Als nächstes wurden nicht nur die Einstellungen der Koder verstellt, sondern verschiedene Einstellungen der Verschlüsselung getestet.
Es gab nun keine besonderen Auffälligkeiten an den Signalen bei ihrer Übermittlung. Was man aber deutlich sah, ist, dass die Zahl und Länge der Aussetzer, die bei den Signalaufzeichnungen deutlich zu erkennen sind, im Wesentlichen von der Verschlüsselung abhängen.

Man kann sagen, dass es bei TRIPLEDES durchschnittlich mehr Aussetzer gibt als bei CAST. CAST ist auch die schnellste Methode, um die Daten zu komprimieren. Näheres dazu gibt es später bei der Auswertung des Delays.
Bei der Verschlüsselung Blowfish gibt es mehr Aussetzer als bei CAST, die Anzahl und Länge ist in etwa mit der von TRIPLEDES zu vergleichen. Dieser Vergleich der Verschlüsselungen sollte sich später bei den Delay-Auswertungen belegen lassen.

Wenn man die Verschlüsselung ganz ausschaltet, sieht man wie erwartet weniger Aussetzer. Die Übertragung ist dann logischerweise am schnellsten. Allgemein lässt sich damit festhalten, dass die Verschlüsselung kaum Einfluss auf die Signale hat, sondern das Band, abgesehen von den Aussetzern, alleinig vom Koder abhängt.

5.2 Erstellen und Auswerten von Frequenzbändern

Unter dem gleichen Programm Audacity gibt es auch die Möglichkeit, sich Frequenzanalysen von den aufgezeichneten Signalen erstellen zu lassen, welche in diesem Kapitel näher unter die Lupe genommen werden sollten. Es wurde von den aufgezeichneten Bändern jeweils eine Analyse erstellt und diese dann miteinander verglichen.

Bilder der Frequenzanalysen finden sich in den Abbildungen 26 -34, hier steht zuerst das Original, es folgen ADPCM 8000 HZ, GSM 7350 HZ, GSM11025 HZ, GSM Lite 4410 HZ, GSM Lite 7350 HZ, bei denen jeweils die Verschlüsselung CAST eingesetzt wurde und weiter unten die Übertragung mit GSM 11025 HZ mit der Verschlüsselung Blowfish, GSM 11025 HZ ohne das Wählen einer Verschlüsselung gefolgt von GSM11025 HZ mit TRIPLEDES:

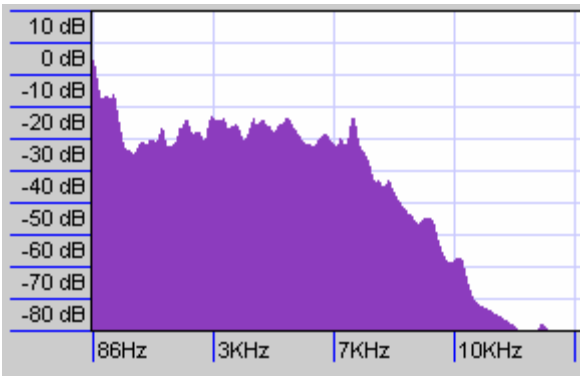

Abbildung 26: Frequenzband des Originals

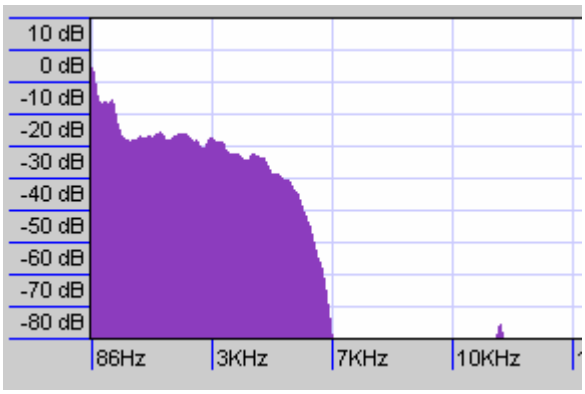

Abbildung 27: Frequenzband mit ADPCM 8000 HZ

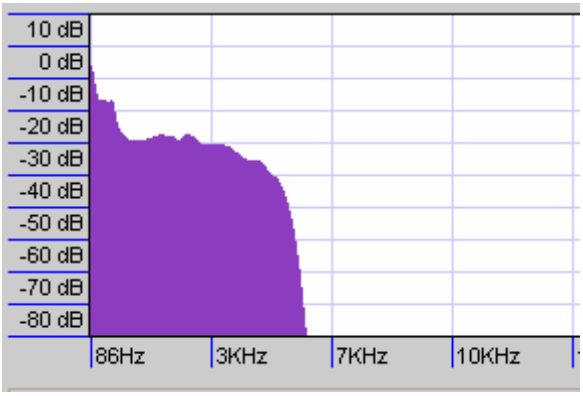

Abbildung 28: Frequenzband mit GSM 7350 HZ

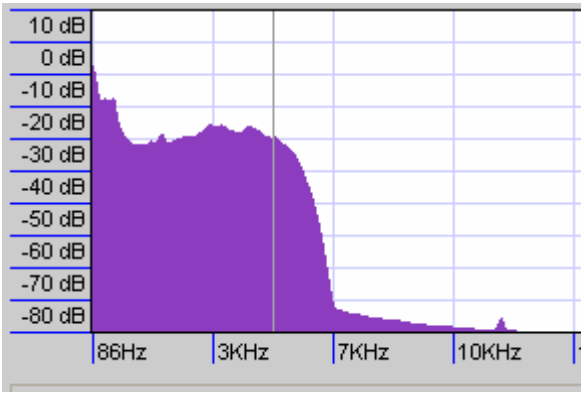

Abbildung 29: Frequenzband mit GSM 11025 HZ

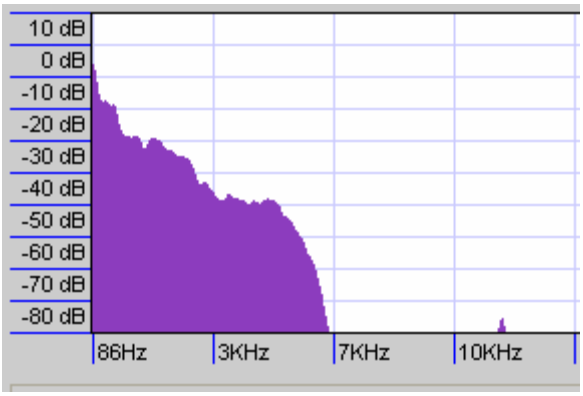

Abbildung 30: Frequenz mit GSM Lite 4410 HZ

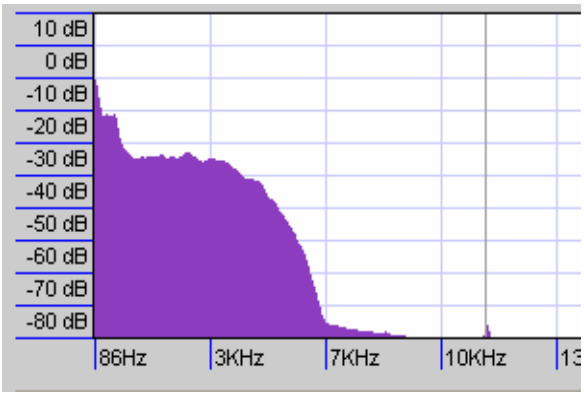

Abbildung 31: Frequenzband mit GSM Lite 7350HZ

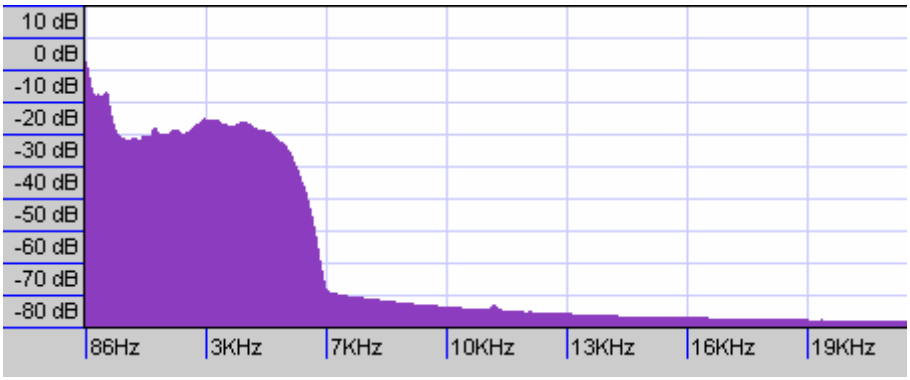

Abbildung 32: Frequenzband mit GSM 11025 HZ und Verschlüsselung Blowfish

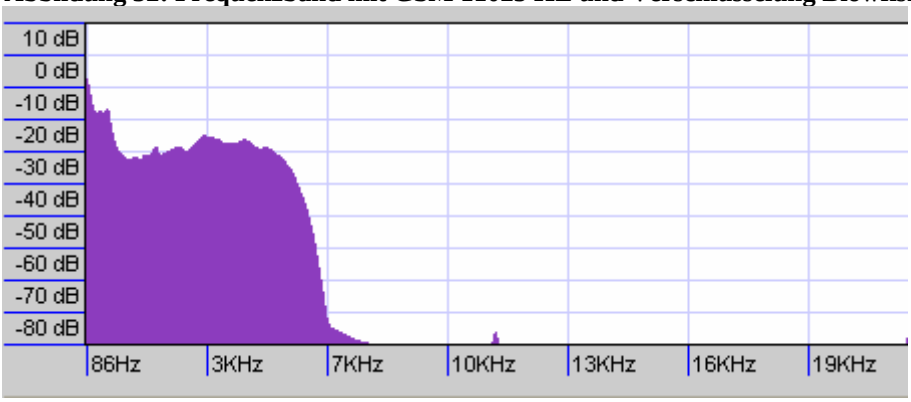

Abbildung 33: Frequenzband mit GSM 11025 HZ ohne Verschlüsselung

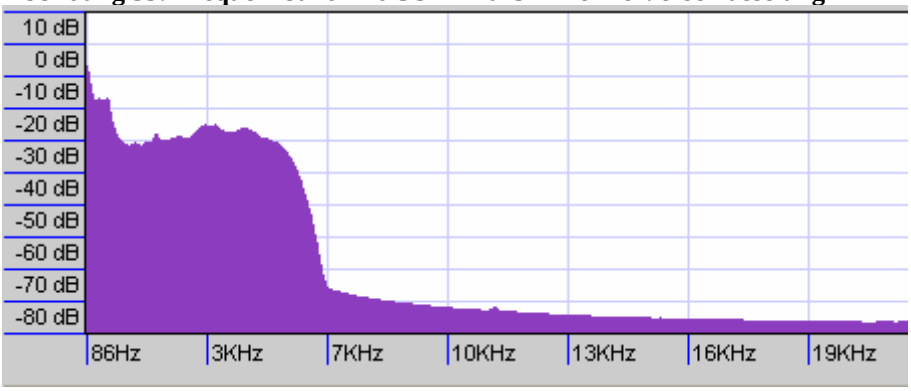

Abbildung 34: Frequenzband mit GSM 11025 HZ und Verschlüsselung TRIPLEDES

Es folgt nun eine Auswertung zu den einzelnen Frequenzanalysen:

ADPCM 8000 HZ CAST:

Ein Abschneiden ist bei ca. 7000 HZ zu sehen, bei ADPCM ist die Originaldatei bei den unteren Frequenzen gut angenähert. Es wird aber nur etwas mehr als die Hälfte des Frequenzbereiches der Originaldatei abgedeckt. Kleinere Unterschiede im Frequenzband der Originaldatei verschwinden, und es entsteht eine recht grobe Struktur.

GSM 7350 HZ CAST:

Hier erfolgt ein Abschneiden sogar schon bei etwa 6000 HZ. Durch die niedrigere Frequenz wirkt das Bild etwas verschwommener, ansonsten sind sich aber die Bänder von ADPCM und GSM recht ähnlich.

GSM11025 HZ CAST:

Bei dieser Methode ergibt sich ein etwas genaueres Abbild der Originaldatei, die einzelnen Verläufe sind klarer zu sehen. Das Abschneiden bei 7000 HZ ist weniger stark, und es ist noch ein sehr abgeschwächtes Signal über 7000 HZ zu erkennen, jedoch wird bei 12 kHZ endgültig abgeschnitten. Dieses abgeschwächte Signal hängt mit der Verschlüsselungsmethode zusammen, aber dazu später mehr.

GSM Lite 4410 HZ CAST:

Bei diesem Koder erfolgt die wohl schlechteste Abtastung, hier sind die unteren Bereiche von etwa 1000 HZ noch einigermaßen klar zu erkennen, doch schon darüber in den mittleren Frequenzbereichen sieht man ein deutliches Abfallen der Werte, ein Abschneiden erfolgt knapp vor 7000 HZ.

GSM Lite 7350 HZ CAST:

Bei größerer Frequenz sieht man eine Steigerung der Qualität bei GSM Lite, dennoch ist die Abweichung noch sehr groß und die Abtastung weit aus gröber, als dies bei GSM zu sehen ist. Vor allem im mittleren Frequenzbereich sind deutliche Verbesserungen zum vorherigen Band zu sehen. Trotzdem wirkt das Bild etwas verwaschen.

GSM11025 HZ Blowfish, CAST, TRIPLEDES und keine Verschlüsselung:

Interessant zu sehen ist, dass sich im Bereich der niedrigen Frequenzen bei der Art der Verschlüsselung nichts ändert. Es erfolgt bei GSM 11025 Hertz die wohl genauste Abtastung des Ausgangssignals, die PGPfone anbietet. Bei 7000 HZ schneidet der Koder ab.
Was nach diesem Bereich sehr gut zu sehen ist, ist, dass bei eingeschalteten Verschlüsselungsalgorithmen auch geringe Frequenzen oberhalb dieser Grenze zu sehen sind.

Ein vermutlicher Grund für diese Werte liegt in den Aussetzern, die die Verschlüsselungen ergeben und die bei den Signalen zu erkennen waren. So wird vermutlich die Frequenz, die Audacity bei einem Aussetzer feststellt, in diesen hohen Frequenzen liegen. So treten, wie auch im letzten Kapitel zu erkennen, nach einem Aussetzer verstärkt Zeitverschiebungen auf. Auch diese können ein Grund für das Auftreten dieser hohen Frequenzen sein. Im Gegensatz dazu finden sich diese hohen Frequenzwerte bei ausgeschalteter Verschlüsselung nicht. Bei Blowfish und TRIPLEDES sind die Ausprägungen am stärksten. Das ist damit zu erklären, dass hier die Zahl der vorkommenden Aussetzer am größten ist. Hier sind Werte größer als 20 kHZ zu erkennen, wohingegen CAST keine Frequenzen mehr in Bereiche über 12 kHZ erkennen lässt.

Um dieser Vermutung nachzugehen, wurden nun weitere Frequenzanalysen vorgenommen, bei denen die Untersuchung nur über kleine Teilbereiche ohne Aussetzer vorgenommen

wurde. Das Ergebnis bestätigte die Vermutung. Die verschiedenen Analysen an der gleichen Stelle der Aufzeichnung, an welcher keine Aussetzer zu erkennen waren, ergaben Frequenzanalysen, bei denen keine Unterschiede festgestellt werden konnten. Die Verschlüsselung hat folglich keinen Einfluss auf die Sprachqualität.
In Abbildung 35 lässt sich die Frequenzanalyse, wie sei beim Koder GSM 11025 und den Verschlüsselungsverfahren CAST, TRIPLEDES, Blowfish und dem Übertragen ohne Verschlüsselung ergeben hat, sehen. Beim Vergleichen waren keine Unterschiede zu sehen, so dass nur eine Frequenzanalyse dargestellt ist.

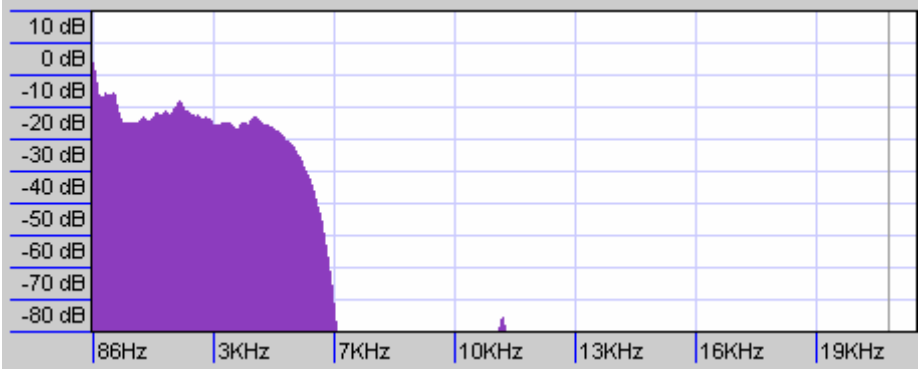

Abbildung 35: Frequenzband mit GSM 11025 HZ

Bei fast allen Frequenzanalysen ist ein kleiner Peak bei 12 kHZ zu sehen. Warum dies der Fall ist konnte nicht festgestellt werden.

5.3 Untersuchungen der Grenzfrequenzen des Koders

Wie in den gerade getesteten Frequenzbändern zu sehen ist, schneidet der Koder die Frequenzen ab einem gewissen Bereich ab, um den Sound-Input besser komprimieren zu können. Folglich war das nächste Ziel, herauszufinden, wo genau die Grenzfrequenz liegt. Hierzu wurden zunächst einige Tools getestet, mit denen man entsprechende Töne auf Basis von unterschiedlichen Frequenzen erstellen konnte. Ein Tool, bei dem automatisch ein Ton über einen gewissen Frequenzbereich erstellt werden kann, wurde nicht gefunden. Nach dem Testen einiger Programme wurde wieder auf das bisher schon verwendete Programm Audacity zurückgegriffen. Mit der neueren Version des Programms ist es möglich, selbst Töne mit einer bestimmten Frequenz zu erstellen. Es ist bei dem Programm so möglich, eine bestimmte Frequenz einzugeben, woraus dann ein Ton generiert wird.

Die oberen Frequenzen, bei denen der Koder abschneidet, liegen wie oben gut zu sehen, im Bereich von 6000 HZ bis 7500 HZ. Auch sollte untersucht werden, ob es eine untere Grenzfrequenz gibt, was bei den Analysen nicht direkt zu entnehmen war. So wurde eine Audiodatei erstellt, die in den unteren Bereichen von 1 HZ bis 100 HZ und anschließend den Bereich von 6000 HZ bis 7500 HZ in 10 HZ-Schritten abdeckte. Das bedeutet, jede Sekunde wurde die Frequenz innerhalb dieser Bereiche erhöht. So sollte dann zu erkennen sein, ab wann keine Daten mehr übermittelt werden.

Also wurde ein neuer Versuch gestartet, bei dem Frequenzbänder aufgezeichnet wurden und diese selbst erstellte Sounddatei übermittelt wurde.

Ergebnis zu den unteren Frequenzen:

Hier war die niedrigste gemessene Frequenz 33 HZ, wobei Audacity nicht weiter aufschlüsselt und keine niedrigeren Frequenzen erkennen lässt. Damit kann man davon ausgehen, dass der Koder wohl die unteren Frequenzen nicht abschneidet. Die Tests wurden mit allen Kodern durchgeführt, jedoch keine Koder-spezifischen Unterschiede festgestellt. Erstaunlich war aber das Signal, welches sich bei den tiefen Frequenzen ergab. Hier sieht man einen deutlichen Unterschied der verschiedenen Koder. Während man im ersten Teil des folgenden Signalausschnittes in Abbildung 36 den Koder GSM sieht, wurde in der zweiten Hälft zu ADPCM gewechselt. Die Signalaufzeichnung wurde bei der Frequenz von 1 HZ vorgenommen.

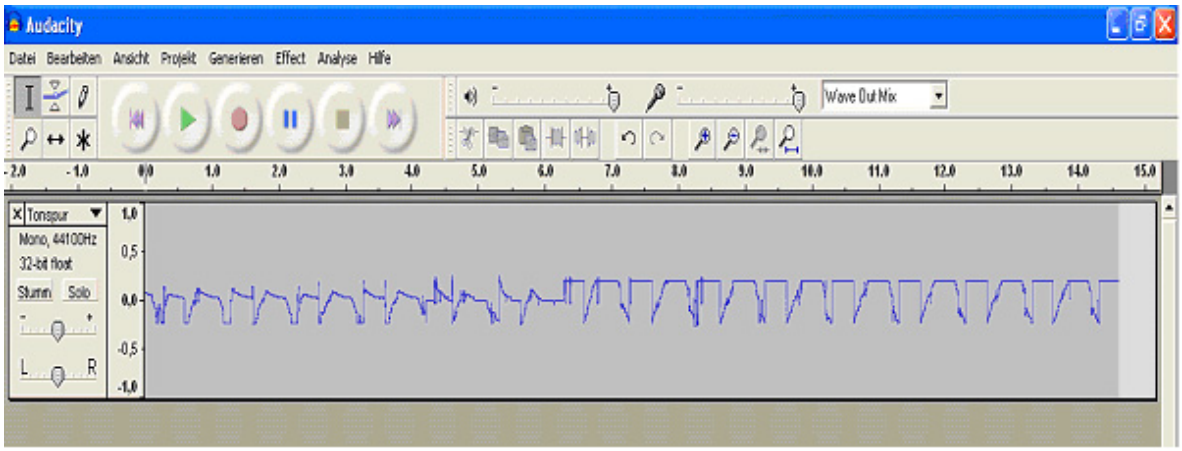

Abbildung 36: Signal bei der Frequenz 1 kHZ

Ergebnis zu den hohen Frequenzen:

Hier war festzustellen, dass alle Koder zwischen 6500 HZ bis 7200 HZ abschnitten. Leichte Differenzen der einzelnen Koder konnten festgestellt werden, so schnitt ADPCM bei ziemlich genau 7000 ab, wohingegen GSM eine etwas weitere Spanne bis ungefähr 7200 zuließ. Abhängig war die Messung jedoch stark von der einzelnen Frequenz, mit der die Verschlüsselung durchgeführt wurde. GSM mit geringer Frequenz schnitt wesentlich früher, nämlich bei 6500 HZ ab. Bei ADPCM und GSM Lite wurde kein sehr großer Unterschied festgestellt, sie schnitten beide ziemlich genau bei 7000 HZ ab.

5.4 Tests mit NistNet

Nun ging es daran einige Tests mit NistNet durchzuführen, um ein Netzwerk mit Hilfe von verschiedenen Parametereinstellungen, wie zum Beispiel einem Delay, zu simulieren. Der Testaufbau wurde bereits bei der Installation von NistNet beschrieben. Es wurden die Werte für die Verlustrate drop geändert, sowie im anschließenden Versuch der Wert für die Verdopplungsrate dub. In einem weiteren Versuch wurde die höchste zulässige Bandbreite geändert. Dabei wurden jedes Mal Signale aufgezeichnet, und es war sehr gut zu erkennen, wie sich PGPfone mit einem Paketverlust arrangierte. Auch bei Verdoppelung der Pakete war das Ergebnis sehr interessant, und beim letzten Versuch konnte die Bandbreite ermittelt werden, mit der PGPfone arbeitet. Die Signale wurden wieder ausgewertet und in Worte gefasst. In Abbildung 37 ist zuerst ein Überblick über aufgenommene Signale zu sehen.
Als erstes ist eine Aufzeichnung des Signals mit eine Rate von 1 % drop (A) zu erkennen, hier sieht man noch keine Unterschiede zur Originaldatei und keine Aussetzer, es folgen 10% drop (B), 20% drop (C) und ganz unten 10% dub (D).

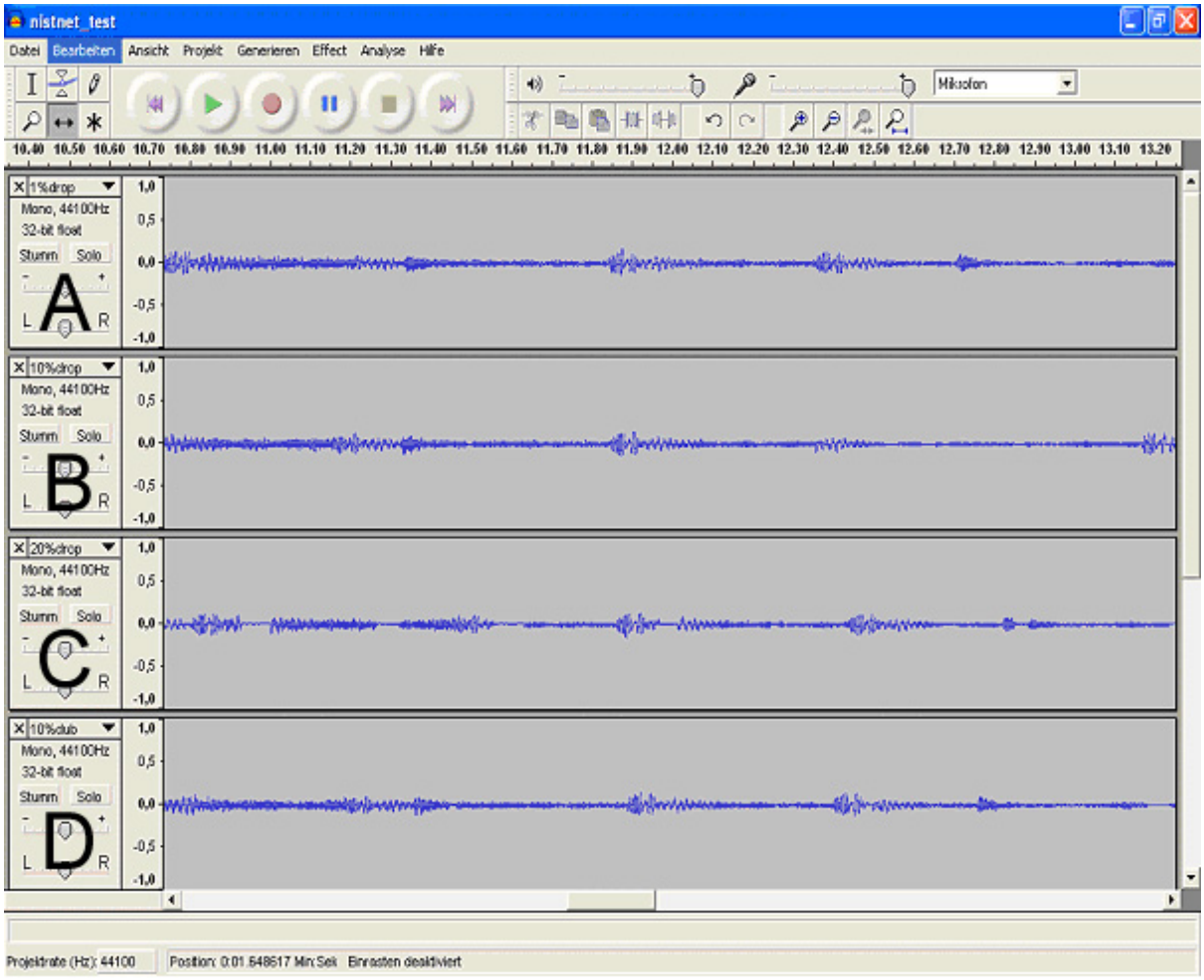

Abbildung 37: Signale des Tests mit NistNet

Im Weiteren folgen nun die Auswertungen zu den einzelnen Aufzeichnungen. Hier gilt, dass oben nur ein Ausschnitt zu sehen ist und des Weiteren nur vier von insgesamt zwölf Signalaufzeichnungen, die zur Auswertung benutzt wurden. Folglich sind nicht alle unten stehenden Beobachtungen in den oben stehenden Aufzeichnungen wieder zu erkennen.

Bei 1% drop:

Hier sind bereits ganz leichte und wenige Aussetzer zu hören jedoch anhand der Signalaufzeichnungen kaum zu sehen. Die Aussetzer sind als ein kleines „Knacken“ beim Anhören wahrzunehmen. Die Verständlichkeit ist aber noch gut gegeben und der Qualitätsverlust ist tolerierbar. Man kann sagen, dass PGPfone den Verlust von 1 % seiner Pakete gut verkraftet.

Bis zu 10% drop:

Bis zu dieser drop-Rate kann man noch von Verständlichkeit sprechen. Man sieht zwar deutlich die Ausfälle anhand der Signale und man hört auch längere Pausen. Die Qualität ist bei 10% erwartungsgemäß nicht mehr sehr gut. Bis zu einer Verlustrate von 10% ist der Gesprächspartner aber noch zu verstehen.

Bei 20% bzw. 50%:

Es ist nichts mehr zu verstehen, und es sind nur noch bruchstückhaft einzelne Teile des Originals zu erkennen. In den Aufzeichnungen gut zu sehen sind die Aussetzer, die zum Teil über längere Zeit andauern.

Bei dub:

Hier ist gut erkennbar, wie die jeweils doppelt ankommenden Pakete wieder aussortiert werden, so dass im Prinzip der gleiche Ton zu hören sein sollte. Praktisch entsteht jedoch ein etwas verzerrtes Bild, da das Aussortieren von doppelten Paketen wohl einige Zeit kostet. Dadurch ergeben sich einige Aussetzer, die zu Zeitverschiebungen beim Signal sowohl nach vorne als auch nach hinten führen. So ist bei Verdoppelung der Pakete die Verständlichkeit noch sehr viel länger gegeben. Ab einer Rate von 50% wird es aber auch hier kritisch. Die Verständlichkeit ist hier nicht mehr vorhanden, da es zu großen Zeitverzögerungen und damit zu Aussetzern kommt.

Bei Bandwidth:

Auch war bei NistNet sehr leicht die Bandbreite zu ermitteln, die von PGPfone benötigt wird. Hierzu wurde die von NistNet erlaubte Bandbreite Stück für Stück nach unten gesetzt. Interessant war zu sehen, dass sich nicht die Qualität verschlechterte, sondern ab einer bestimmten Grenzbandbreite, die in diesem Fall bei 5000 Bytes/s lag, keine Pakete mehr übermittelt werden und schon eine Anwahl zu einem anderen Rechner scheitert. Hier wurde folglich ermittelt, dass alle Schritte bei PGPfone, also sowohl die Anwahl als auch die Übertragung einzelner Pakete, die Bandbreite von 5000 Bytes/s benötigen. Unterhalb dieser Bandbreite ist weder eine Anwahl noch bei bestehender Verbindung ein weiteres Kommunizieren möglich.

5.5 Auswertung der durchgeführten Delay-Messungen

Die gemessenen Werte aus Kapitel 4.6 wurden nun ausgewertet. Es wurde festgestellt, dass solange sich keine Pakete in der Warteschlange stauen, sich der Delay immer gleich verhält. Deshalb ist es in diesem Fall egal, was für die durchgeführten Tests übertragen wurde. Des Weiteren wurde noch ein Test durchgeführt, bei dem sehr viel „Traffic“ erzeugt wurde, um die Warteschlange anwachsen zu lassen. Hier stieg der Delay entsprechend an. Zunächst wurden für den Versuch die Uhren der beiden Rechner synchronisiert. Die Abweichung der Uhren lag auf jeden Fall bei maximal 1 ms, so dass die Berechnungen des Delays einen möglichen Fehlerwert von 1 ms aufweisen können. Bei allen hier durchgeführten Messungen wurde der Koder GSM mit 11025 HZ verwendet. Es wurde nun mit den verschiedenen Einstellungen jeweils eine Verbindung aufgebaut, Hintergrundgeräusche übertragen, so dass die Warteschlange leer war, und dann jeweils auf beiden Seiten der Verbindung die jeweiligen Zeitstempel in eine Textdatei geschrieben.

Die Messung vom Beginn des Verarbeitens vom Input, also noch vor dem Koder, und der Vergleich mit der Messung vor dem Output, also nach dem Dekodieren, liegt bei **ca. 49 ms**, sofern keine Verschlüsselung gewählt wird. Die durchschnittliche Abweichung von diesem Wert liegt bei ungefähr **7 ms**. Damit kann dann die Standardabweichung berechnet werden.

Wenn man nun mehr „Traffic“ erzeugt, indem man in beide Richtung relativ viel überträgt, also zum Beispiel viel redet oder die vorgefertigte Sound-Datei von beiden Seiten aus laufen lässt, erhält man schnell einen Delay von **57 ms**, auch hier liegt die Abweichung bei ca. **7ms**.
Das heißt, der Delay steigt an, in den Tests um bis zu 10 ms, wenn sich Pakete in der Warteschlange stauen. Das Anwachsen der Warteschlange kann man sehr schön an den mitgelieferten Statistiken von PGPfone erkennen. Der Wert bei diesem Test stieg auf bis zu 2000 an. Dies ist der Fall, wenn viel übertragen wird. Bei normalen Anwendungen, wie sie beim Telefonieren üblich sind, wird man eher den Wert der Warteschlange von 0 und einen Delay von 49 ms erreichen.

Aus anderen Dateien, in die Zeitstempel geschrieben wurden, ließ sich erkennen, dass das eigentliche Senden des Paketes, also vom Setzen auf die Pipe bis zum Empfangen auf der anderen Seite, im Normalfall weniger als eine Millisekunde dauerte. Der maximal vorkommende Wert lag bei 2 ms, was aber eine Ausnahme war. Hierbei ist natürlich zu beachten, dass die Rechner direkt miteinander vernetzt und damit keine langen Wege zu überbrücken waren.

Nun ging es daran, Tests vorzunehmen, bei der die Verschlüsselung eingeschaltet war. Ziel war es, einen Delay-Unterschied zwischen den einzelnen Verschlüsselungsverfahren zu messen. Eine der Hauptfragen dieser Studienarbeit war es, zu überprüfen, ob durch die zusätzliche Sicherheit ein Qualitätsverlust entsteht. Das kann mit dieser Delay-Messung zumindest ein Stückweit beantwortet werden.
Zu beachten ist, dass der Schlüsselaustausch mit dem Standardwert von 2048 Bit vorgenommen wurde. Beim Ändern dieses Wertes ergeben sich entsprechende Abweichungen, das heißt, dass ein Schlüsselaustausch mit mehr Bits auch mehr Zeit in Anspruch nimmt. Die Änderungen lagen jedoch nur im Bereich von wenigen Millisekunden. Bei den hier folgenden Tests wurde wieder mit wenig „Traffic“ gearbeitet, so dass der Wert der Warteschlange bei 0 lag.

Bei eingeschalteter Verschlüsselung CAST ergibt sich dann eine Gesamtlaufzeit vom Input zum Output von **55 ms**, die durchschnittliche Abweichung, die gemessen wurde, beträgt hier **9 ms**.
TRIPLEDES benötigt anscheinend wesentlich mehr Zeit als CAST, bei dem es sich um eine Verschlüsselungsmethode handelt, die im Wesentlichen Zeit optimiert sein soll. So ergaben sich Messungen mit einem Delay von **70 ms**, die Abweichung betrug hier im Durchschnitt **8 ms**. Ganz ähnlich lief die Messung nun mit der Verschlüsselung Blowfish, hier konnte ein Delay von **68 ms** gemessen werden bei einer durchschnittlichen Abweichung von **9 ms**.

Insgesamt kann man sagen, dass die Verschlüsselung also zwischen 6 und knappen 20 ms kostet. Das heißt, dass ein ganz erheblicher Zeitfaktor der Übermittlung die Verschlüsselung ist. Sie kann bis zu **30%** der Gesamtzeit einnehmen. Es ist folglich durchaus ein beträchtlicher Faktor, den es zu berücksichtigen gilt.

In Abbildung 38 sind die Ergebnisse der Delay-Messungen noch einmal übersichtlich dargestellt.

	Delay
ohne Verschlüsselung	**49 ms**
ohne Verschlüsselung und viel Verkehr	**57 ms**
mit der Verschlüsselung CAST	**55 ms**
mit der Verschlüsselung TRIPLEDES	**70 ms**
mit der Verschlüsselung BLOWFISH	**68 ms**

Abbildung 38: Auswertung des Delays mit Koder GSM 11025 HZ

Die Werte der Delays wurden nun noch mit verschiedenen Kodern getestet. Um die Tests durchzuführen, wurden als einer der schnellsten Koder GSM Lite 4410 HZ, danach APDCM 8000 HZ und als den wohl genausten Koder GSM 11025 HZ, ausgewählt. Man erkennt die Ergebnisse der Tests in Abbildung 39.

	GSM Lite 4410 HZ	**ADPCM 8000 HZ**	**GSM 11025 HZ**
Keine Verschlüsselung	45 ms	47 ms	49 ms
CAST	52 ms	53 ms	55 ms
Blowfish	66 ms	65 ms	68 ms
TRIPLEDES	66 ms	66 ms	70 ms

Abbildung 39: Auswertung des Delays mit verschiedenen Kodern

Gut zu sehen ist, dass ADPCM mit 8000 HZ etwa soviel Zeit benötigt wie GSM Lite mit 4410 HZ. GSM dagegen benötigt etwa 3-4 ms mehr Zeit, als dies bei GSM Lite der Fall ist.

5.6 Berechnen der Standardabweichung

Die Standardabweichung des Delays spielt eine wichtige Rolle. Sie ist fast ebenso wichtig anzusehen wie der Delay, denn von der Standardabweichung hängt ab, wie groß der Puffer sein muss, um Pakete zu speichern. Die Puffergröße steht jedoch direkt wieder im Bezug zur eigentlichen Geschwindigkeit und damit zum Delay.

Die Varianz σ^2 berechnet sich wie folgt:

$$\sigma^2 = \frac{\sum_{i-1}^{n} (\mu - x_i)^2}{n}$$

Daraus folgt die Standardabweichung σ als Wurzelfunktion der Varianz σ^2:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum_{i=1}^{n} (\mu - x_i)^2}{n}}$$

Der Teil in der Klammer, der quadriert wird, ist die schon durchgeführte Berechnung der Abweichung. Anhand dieser kann nun also die Varianz bestimmt werden. Bei der Formel ist zu beachten, dass durch das Quadrieren der Werte nun größere Abweichungen ein wesentlich größeres Gewicht bekommen als kleine Abweichungen.
Bei den durchgeführten Messungen war es jedoch so, dass die Werte der durchschnittlichen Abweichung ziemlich genau denen der Standardabweichung entsprachen. Das war deshalb der Fall, weil die Abweichung nach oben und unten immer relativ gleich groß war.

So ergab sich eine Standardabweichung, wie sie in Abbildung 40 dargestellt ist. Alle Werte wurden jeweils auf die volle Millisekunde auf- oder abgerundet. Zur Berechnung wurden immer mindestens 30 verschiedene Werte herangezogen.

	Standardabweichung
ohne Verschlüsselung	**7ms**
ohne Verschlüsselung und viel Verkehr	**7ms**
mit der Verschlüsselung CAST	**9ms**
mit der Verschlüsselung TRIPLEDES	**9ms**
mit der Verschlüsselung BLOWFISH	**10 ms**

Abbildung 40: Auswertung der Standardabweichung

5.7 Vergleich mit Angaben zum Delay von anderen VoIP-Systemen

Ein durchschnittlicher Wert für die Zeit, die Verschlüsselung bei VoIP-Systemen einnimmt, liegt bei 10-25 ms, somit ist PGPfone in diesem Schnitt.

Die Gesamtlaufzeit liegt bei modernen VoIP-Systemen bei 50 ms, hier ist PGPfone etwas schlechter.

Ein System, welches in der Fachliteratur als brauchbar angesehen wird, sollte die Laufzeit von 150 ms nicht unterschreiten. Hier liegt PGPfone gut in diesem Rahmen, wobei es natürlich immer zu berücksichtigen gilt, dass die Tests mit direkt vernetzen Computern durchgeführt wurden. Ein hinzukommender Delay durch die Netzübertragung lässt sich jedoch in den meisten Fällen nicht vermeiden.

Man spricht allgemein davon, dass ein Delay ab ungefähr 50 ms hörbar ist, was mit den subjektiven Wahrnehmungen übereinstimmt, denn bei eingeschalteter Verschlüsselung ist ein kleiner Delay zu hören. [LANL03]

PGPfone liefert also im Vergleich zu anderen VoIP-Systemen brauchbare Voraussetzungen. Die Werte des Delays befinden sich zwar eher im Mittelfeld bzw. im unteren Bereich, wenn man den Vergleich zu anderen Systemen heranzieht. Sie liegen jedoch im Rahmen der vorgegebenen Toleranz. Mittlerweile gibt es moderne, neue Verschlüsselungsalgorithmen, die einen kleineren Delay ermöglichen. Da PGPfone schon 1998 auf dem Markt erschien, konnten diese Algorithmen nicht eingebaut werden.

6. Fazit

In der Arbeit wurden verschiedene Tests durchgeführt und verschiedene Ergebnisse erzielt. So konnte man zum einen festgestellt werden, dass die Verschlüsselung die Qualität der Sprachübertragung nicht verschlechtert. Sie wirkt sich jedoch auf den Delay aus. In den Ergebnissen der Delay-Messungen konnte man erkennen, dass die Verschlüsselung bis zu 30% der Gesamtlaufzeit ausmachen kann. Zudem ließ sich feststellen, dass die Verschlüsselung der Grund für einige Aussetzer bei der Übertragung mit PGPfone ist. Die Grenzfrequenzen des Koders wurden übermittelt und man konnte feststellen, dass die Datei im Wesentlichen komprimiert wird, in dem die oberen Frequenzen abgeschnitten werden. Es ließen sich auch große Qualitätsunterschiede der einzelnen Koder feststellen. So wurde GSM mit 11025 HZ als der beste Koder ermittelt und GSM Lite 4410 HZ als der schlechteste.

Das Programm scheint recht gut gelungen und auf jeden Fall brauchbar zu sein. Für eine normale Sprachanwendung reicht die Qualität vollkommen aus. Das Programm ist übersichtlich und einfach zu bedienen, was ein großes Plus darstellt. Zudem ist es frei erhältlich und ein Open-Source-Projekt. Es besticht durch seriöse, sichere Übertragung, was bei anderen kostenlosen Produkten, die sich auf dem Markt befinden, meist nicht der Fall ist. Somit macht das Programm Sinn, wenn man Wert auf eine sichere Übertragung legt und einen etwas größeren Delay in Kauf nimmt.

Negativ zu bemängeln ist allerdings, dass der Stand des Programms nicht mehr „Up-to-Date“ ist. Es gibt mittlerweile neuere und schnellere Algorithmen, die jedoch von PGPfone nicht verwendet werden. Der Delay des Programms PGPfone mit eingeschalteter Verschlüsselung liegt bereits deutlich im hörbaren Bereich. Von Vorteil wäre natürlich auch, wenn das Programm eine Konferenzschaltung zulassen würde, oder wenn das Programm für den Datenaustausch genutzt werden könnte. Dies ist leider beides nicht der Fall und ist möglicherweise auch ein Grund, warum sich das Programm nie stark auf dem Markt durchsetzen konnte.

Als Fazit kann man sagen, dass das Programm PGPfone für den privaten Gebrauch ausreichend ist und gutes Telefonieren ermöglicht. Zu kommerziellen Zwecken ist der Delay zu groß, und es wären die Möglichkeit der Konferenzschaltung und modernere Verschlüsselungsalgorithmen nötig.

Literatur

[LANL03] Heft „LANline“ aus dem Jahr 2003, Ausgabe 10, Seite 100 ff.

[DOCP] Dokumentation von PGPfone Philip Zimmermann, 1996

[DOCN] Dokumentation von NistNet, Marc Carson, 2000

[HIAUD] Hilfedokumentation zu Audacity, Free Software Foundation,
Boston, MA, 1991

[ITU107] ITU-T Recommendation G.107, The E-model, a computational model for use in transmission planning,
03/2003

[ASSQU] Assessing the Quality of Voice Communications over Internet Backbones
Athina P. Markopoulou, Fouad A. Tobagi, Mansour J. Karam,
Stanford, CA, 2003

[FITU] Fine-tuning Voice over Packet services
Yuval Boger, VP Business Development, RADCOM Ltd.,
Paramus, NJ, 1999

[VOIPAS] Voice over IPsec: Analysis and Solutions
Roberto Barbieri, Danilo Bruschi, Emilia Rosti,
San Diego, CA, 2002

[MOEF] Modeling the Effects of Burst Packet Loss and Recency on Subjective Voice Quality
A. D. Clark, Ph.D. Fellow
Stanford, CA, 2001

[VOIPPE] Voice over IP Performance Monitoring
Cole, R. G. and Rosenbluth, J.H.,
Middletown, NJ, 2001