



BACHELOR - ARBEIT

Mobiler, persönlicher Assistent

Patrick Hornecker

betreut durch

Klaus Rechert

an der

Technischen Fakultät
der Albert-Ludwigs-Universität Freiburg

Inhaltsverzeichnis

1	Einleitung	2
2	Tutorial	3
2.1	CeGCC	3
2.2	Enlightenment	3
2.2.1	evil	5
2.2.2	eina	6
2.2.3	eet	6
2.2.4	embryo	7
2.2.5	evas	7
2.2.6	ecore	9
2.2.7	edje	9
2.2.8	elementary	10
2.2.9	Weitere Schritte	10
3	Friend Finder	11
3.1	Features	11
3.1.1	Nachrichten versenden	11
3.1.2	Eigene Position senden	11
3.1.3	Position anderer Teilnehmer anzeigen	11
3.1.4	2D-Barcode	12
3.2	Verwendete Verfahren und Bibliotheken	12
3.2.1	Versenden der Nachrichten	12
3.2.2	Verschlüsselung der Daten	12
3.2.3	Erzeugen eines 2D-Barcodes	13
3.2.4	Grafisches Benutzeroberfläche	13
3.3	Implementierung	13
3.4	Analyse	15
4	Ausblick	16
4.1	Plattformunabhängigkeit	16
4.1.1	Windows Mobile	16
4.1.2	Android	16
4.1.3	WebOS	17
4.2	Kryptographische Verfahren auf Mobilien Plattformen	17
4.2.1	Symmetrische Verschlüsselungsverfahren	18
4.2.2	Asymmetrische Verschlüsselungsverfahren	18
4.3	Alternative <i>location awareness</i> Verfahren	18
4.4	Zusammenfassung	18

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum

Unterschrift

1 Einleitung

In der heutigen, vernetzten Gesellschaft wird es den Leuten relativ einfach gemacht sich untereinander, auch über große Entfernungen, auf dem digitalen Weg zu verständigen. Die Spannweite dieser Vernetzung reicht von E-Mails, Personal Messenger bis hin zu sozialen Netzwerken wie zum Beispiel *facebook*, *StudiVZ* oder *twitter*.

Seit einiger Zeit wird diese Vernetzung um eine neue Möglichkeit erweitert. Die Masse an neuen Modellen und der Erfolg von *Smartphones* ermöglicht es dem Nutzer immer und so gut wie überall über das Internet verfügbar zu sein. So existieren schon Unmengen an Kommunikationssoftware und Applikationen die speziell für diese Mobilen Geräte entwickelt wurden.

Mit dem Aufkommen dieser neuen und mobilen Internetplattform muss auch hier wieder die Frage nach der Sicherheit von Daten und Persönlichen Informationen gestellt werden. So gelten für *Smartphones* andere Voraussetzungen wie für Desktop Computer. Der Energieverbrauch sollte möglichst gering gehalten werden, um den Akku nicht allzuschnell zu entladen. Es muss auch berücksichtigt werden, dass die Mobilen Geräte nicht an die Leistungsfähigkeit regulärer Computer heranreichen und somit nicht die aufwändigsten Verfahren gewählt werden können.

Der Inhalt dieser Bachelor-Arbeit behandelt eben dieses Thema: Das verschlüsselte versenden von Informationen auf *Smartphones* und des weiteren die Möglichkeiten von *location awareness*. Für diesen Zweck wurde ein Programm namens "*Friend Finder*" implementiert, welche einfache Chat Nachrichten sowie die eigene Position versenden kann. Diese Positionen können dann von anderen Benutzern auf ihrem Mobilen Gerät angezeigt werden. Des weiteren beinhaltet diese Arbeit eine Analyse des Datenverkehrs, welcher durch diese Software erzeugt wird. Es werden auch andere Möglichkeiten von *location awareness* und Verschlüsselung aufgezeigt und besprochen.

Ein weiterer Punkt der aktuellen Generation der *Smartphones* ist, das es verschiedene Betriebssysteme gibt. Diese unterscheiden sich je nach Hersteller. Diese Ausarbeitung beinhaltet auch ein kurzes Tutorial, um das Softwarepaket *Enlightenment*[efl] von einer Linux-Plattform nach Windows Mobile zu konvertieren.

2 Tutorial

Beim folgenden Abschnitt handelt es sich um ein *Tutorial* in welchem Schritt für Schritt erklärt was notwendig ist um *Enlightenment* [?] und das Programm *Ueberall* von Linux nach *Windows Mobile 6.1* [Windows] zu portieren. Bei dem genutzten Linux handelt es sich um ein Ubuntu Version 9.10. Für dieses Vorhaben werden mehrere Tools benötigt, welche auch im folgenden kurz vorgestellt werden. Zum einen wird das *Enlightenment* Paket benötigt, welches aus mehreren Unterprogrammen besteht. Diese wurden allesamt aus dem *Subversion Repository* der Entwickler heruntergeladen. Ein weiterer wichtiger Rolle in diesem Vorhaben spielt der *CeGCC-Compiler* [?], welcher für das kompilieren von Programmcode von Linux nach Windows Mobile benötigt wird.

Diese Programme bilden die Grundlagen für diese Aufgabe. Um *Ueberall* kompilieren ist es auch nötig noch ein paar *Libraries* zu portieren. Auf diese wird im Abschnitt *Ueberall* genauer eingegangen.

Als erstes wird nun auf den *CeGCC* näher eingegangen, danach auf das Erstellen von Enlightenment für Windows Mobile und schliesslich wird auch das portieren von *Ueberall* genauer besprochen.

2.1 CeGCC

Der CeGCC ist ein *Open-Source* Projekt, welches ein *Crosscompiler* von Linux nach Windows Mobile entwickelt hat. Dieser Compiler basiert auf dem standard Unix C-Kompiler, dem GCC. Eine aktuelle Version des CeGCC's kann auf der Projekthomepage gefunden und heruntergeladen werden. Es wird hierbei prinzipiell zwischen zwei verschiedenen Arten des CeGCC's unterschieden. Zum Einen *CeGCC*, zum Anderen *mingw32ce*. Der Unterschied zwischen diesen beiden Compilern besteht darin, dass ersterer nur dann benutzt wird, wenn man nur Linux APIs nutzt. Im Unterschied dazu wird der *mingw32ce*-Compiler dann gebraucht, wenn man auch *Windows Mobile* APIs nutzen möchte.

Für diese Aufgabe wird der *mingw32ce* für den ARM-Prozessortyp benötigt. Dieser kann auf der Homepage des CeGCC-Projekts heruntergeladen und in das passende Systemverzeichnis entpackt werden.

Hat man diese zwei Schritte erledigt, hat man auch schon den CeGCC erfolgreich auf seinem System installiert.

2.2 Enlightenment

Unter dem Namen *Enlightenment* werden mehrere Programme zusammengefasst, welche zusammen einen kompletten *Window-Manager* bilden. Auch hier handelt es sich um ein *Open-Source* Projekt. Um ein *Frontend*, welches mit einer Enlightenment Bibliothek erstellt wurde, auf einem

Smart Phone starten zu können müssen mehrere Programme mit dem *CeGCC* kompiliert werden. Diese sind: *evil, eina, eet, embryo, evas, ecore, edje und elementary*. Diese Programme werden mit dem von den Entwicklern bereitgestelltem Source Code kompiliert.

Bevor man allerdings damit beginnen kann, müssen noch ein paar benötigte Pakete aus dem Ubuntu-Repository installiert werden.

```
sudo apt-get install build-essential make gcc bison flex subversion
autoconf libtool gettext libfreetype6-dev libpng12-dev zlib1g-dev
libjpeg-dev libtiff-dev libungif4-dev librsvg2-dev xorg-dev
libltdl3-dev libcurl4-dev cvs subversion git-core doxygen proj
libsqlite3-0 libsqlite3-dev
```

Nachdem diese Pakete installiert wurden kann man sich nun die einzelnen Pakete aus dem *Subversion-Repository* der Entwickler herunterladen.

Nun muss man sich noch ein Verzeichniss anlegen, in welchem die für Windows Mobile kompilierten Dateien abgelegt werden. Des weiteren muss noch eine Datei angelegt werden, in welcher die Pfade zu dem genutzten Kompiler liegen und welche dann einmalig exportiert werden müssen, damit die benötigten *Header-Files*, *textitLibraries* und *Binaries* auch vom Betriebssystem gefunden werden. Diese Datei wird im folgenden "mingw32ce.env" benannt.

```
touch mingw32ce.env
```

Nun müssen noch in diese Datei die zu exportierenden Pfade geschrieben werden.

```
export CEGCC_PATH=/opt/cegccc
export MINGW32CE_PATH=/opt/mingw32ce
export WINCE_PATH=$HOME/workspace/wince

export PATH=$CEGCC_PATH/bin:$MINGW32CE_PATH/bin:$PATH
export CPPFLAGS="-I$WINCE_PATH/include -I$WINCE_PATH/zlib-1.2.3-dev/include
-I$WINCE_PATH/libjpeg-6b-dev/include -I$WINCE_PATH/win_iconv-dev/include
-I$WINCE_PATH/freetype-2.3.7-dev/include
-I$WINCE_PATH/libpng-1.2.33-dev/include/libpng12
-I$WINCE_PATH/win_iconv-dev/include -I/opt/mingw32ce/arm-mingw32ce/include/"
export LDFLAGS="-L$WINCE_PATH/lib -L$WINCE_PATH/zlib-1.2.3-dev/lib
-L$WINCE_PATH/libjpeg-6b-dev/lib
-L$WINCE_PATH/win_iconv-dev/include -L$WINCE_PATH/freetype-2.3.7-dev/lib
-L$WINCE_PATH/libpng-1.2.33-dev/lib -L$WINCE_PATH/win_icon-dev/lib
-L$CEGCC_PATH/lib"
export LD_LIBRARY_PATH="$WINCE_PATH/bin"
export PKG_CONFIG_PATH="$WINCE_PATH/lib/pkgconfig"
```

Der Inhalt dieser Datei muss nun in jeder neu geöffneten Shell neu exportiert werden, da sie durch die hier gewählte Methode nur in eben diesen Shell's gelten wo sie exportiert wurden. Bei den Variablen "CEGCC_PATH" und "MINGW32CE_PATH" ist der Pfad zum Verzeichniss des *cegcc*, beziehungsweise des *mingw32ce* Kompilers einzutragen. Unter "WINCE_PATH" muss der Pfad, zu dem Verzeichniss in dem die kompilierten Daten gespeichert werden sollen, eingetragen werden. Mit "PATH" werden die *Binaries*, der zwei Kompiler, in den Systempfad aufgenommen. Des weiteren werden unter "CPPFLAGS" die *include*-Pfade und unter "LD-FLAGS" die *Librarie* Pfade abgelegt. "LD_LIBRARY_PATH" zeigt auf den Ordner in welchem die kompilierten *Binaries* liegen. "PKG_CONFIG_PATH" zeigt schliesslich noch auf den Ordner der die Paketinformationen der installierten Dateien beinhaltet. Dieses exportieren geschieht mit dem folgenden Aufruf.

```
source <Pfad-zu-der-Datei>/mingw32ce.env
```

Im nächsten Schritt muss nun noch ein Ordner angelegt werden, in welchem der *Enlightenment Source-Code* abgelegt wird. Nun muss noch in dieses Verzeichniss gewechselt werden und es kann mit dem ersten Programm begonnen werden.

2.2.1 evil

Als erstes ist es nötig das Programm *evil* aus dem *SVN*, welches von den Entwicklern bereit gestellt wurde, herunterzuladen. Das herunterladen geschieht mit:

```
svn co http://svn.enlightenment.org/svn/e/trunk/evil
```

Nachdem alle Dateien erfolgreich heruntergeladen wurden muss, falls nicht schon geschen, die Datei mit den *Umgebungsvaribalen* eingelesen werden. Nachdem dies geschehen ist, kann man nun das Konfigurationsskript starten

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce
```

Durch diesen Aufruf wird der Installationspfad auf den Wert der Variable "WINCE_PATH" gesetzt und als Zielsystem ein *ARM-Prozessor* gesetzt und der *mingw32ce*-Kompiler als Kompiler gewählt.

Nachdem dieses Skript erfolgreich durchgeführt wurde, kann man im nächsten Schritt das Programm erstellen.

```
make
```

Ist auch dies erfolgreich durchgelassen, so muss man nun noch in einem letzten Schritt die erstellten Dateien im Zielordner installieren.

```
make install
```

Nun sollte *evil* erfolgreich im Zielordner installiert worden sein.

2.2.2 eina

Auch hier ist es auch wieder nötig die Dateien aus dem Entwickler-Repository herunterzuladen.

```
svn co http://svn.enlightenment.org/svn/e/trunk/eina
```

Danach wird auch hier wieder das “autogen.sh” Skript aufgerufen.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce --disable-pthread
```

Es werden bei diesem Aufruf die gleichen Parameter wie bei *evil* übergeben. Hinzu kommt noch “--disable-pthread”. Mit diesem Parameter wird *threading* beim erstellen von *eina* deaktiviert, da *ARM-Prozessoren* dies nicht unterstützen.

Nachdem das Skript durchgelaufen ist, muss man nun auch wieder das Programm erstellen und im Zielverzeichnis installieren.

```
make ; make install
```

2.2.3 eet

Bevor man *eet* erstellen kann, muss man noch vier vorgefertigte *tar-Archive* im Verzeichnis, welches in der Variable “WINCE_PATH” gespeichert wurde, entpacken. Diese Archive kann man unter den Links, welche in Anhang 2 zu finden sind, herunterladen. Nach dem herunterladen müssen diese nur noch in das “WINCE_PATH”-Verzeichnis kopiert und entpackt werden. Nun kann man den Quellcode für *eet* herunterladen.

```
svn co http://svn.enlightenment.org/svn/e/trunk/eet
```


Nachdem die Dateien heruntergeladen sind, muss wieder das “autogen.sh“-Skript aufgerufen werden.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce
```

Im Anschluss muss nun auch wieder kompiliert und installiert werden.

```
make ; make install
```

2.2.4 embryo

```
svn co http://svn.enlightenment.org/svn/e/trunk/embryo
```

Nachdem die Dateien heruntergeladen sind auch wieder das Skript aufrufen.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce
```

Im Anschluss nun auch wieder kompilieren und installieren.

```
make ; make install
```

2.2.5 evas

Auch für *evas* müssen mehrere *tar-Archive* heruntergeladen werden. Auch sollen diese in das gleiche Verzeichniss, wie die vorhergegangenen Archive, entpackt werden. Nun müssen noch die Dateien, welche die Paketinformationen beinhalten für die heruntergeladen Dateien ergänzt werden:

```
cp $WINCE_PATH/cp libpng-1.2.33-dev/lib/pkgconfig/libpng* $WINCE_PATH/lib/pkgconfig/  
cp $WINCE_PATH/freetype-2.3.7-dev/lib/pkgconfig/freetype2.pc $WINCE_PATH/lib/freetyp
```

Nun müssen diese Paketinformationen noch bearbeitet werden. Dazu müssen diese mit einem beliebigen Editor geöffnet werden und in beiden Dateien der Wert von ”prefix“ auf ”WINCE_PATH“ gesetzt werden.

Nachdem dies durchgeführt wurde kann nun *evas* heruntergeladen werden.

```
svn co http://svn.enlightenment.org/svn/e/trunk/evas
```

Nun muss auch hier, wie bei allen anderen Programmen das "autogen.sh"-Skript aufgerufen werden.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce --disable-async-events
```

Als nächster Schritt muss nun das Programm kompiliert werden.

```
make
```

Sollte hierbei die Datei "ft2build.h" nicht gefunden werden, so muss diese an die richtige Stelle kopiert werden. Eigentlich liegt die Datei an folgendem Ort:

```
$WINCE_PATH/freetype-2.3.7-dev/include/freetype2/ft2build.h
```

Allerdings wird sie im Ordner "freetype2" nicht gefunden. Um dies zu umgehen muss "ft2build.h" einfach eine Ordner Ebene nach oben kopiert werden.

```
cp $WINCE_PATH/freetype-2.3.7-dev/include/freetype2/ft2build.h $WINCE_PATH/freetype-
```

Um einen weiteren Fehler von vorneherein zu umgehen, muss man noch den Pfad eines eingebundenen Headers in "ft2build.h" abändern. Hierzu öffnet man "ft2build.h" mit einem beliebigen Editor und ändert folgendes

```
#include <freetype/config/fthead.h>
```

zu

```
#include <freetype2/freetype/config/fthead.h>
```

ab. Anschliessend zu dieser Lösung muss nun einfach der "freetype"-Ordner um eine Ebene nach oben kopieren werden, da die *include*-Pfade in den Headern von "freetype2" stellenweise nicht korrekt sind.

Falls man nun noch *evas* mit *DirectX-Support* kompilieren möchte, muss man das *DirectX-SDK* herunterladen und "ddraw.h" in die Verzeichnisse "/opt/cegcc/arm-cegcc/include/w32api/" und "/opt/mingw32ce/arm-mingw32ce/include/" kopieren.

2.2.6 *ecore*

Um *ecore* zu erstellen muss zu allererst eine Änderung im “winnt.h”-Header vorgenommen werden. Dieser liegt im *include*-Verzeichniss des *mingw32ce*-Kompilers.

```
#define PROCESS_SET_QUOTA          0x0100
#define PROCESS_SET_INFORMATION    0x0200
#define PROCESS_QUERY_INFORMATION  0x0400
+#define PROCESS_SUSPEND_RESUME     0x0800
#define PROCESS_ALL_ACCESS          (STANDARD_RIGHTS_REQUIRED|SYNCHRONIZE|0xfff)

#define THREAD_TERMINATE           0x0001
```

Der mit “+” gekennzeichnete Eintrag “PROCESS_SUSPEND_RESUME” muss in die Datei “winnt.h” eingefügt werden.

Nachdem dieser Schritt ausgeführt wurde kann nun auch *ecore* kompiliert werden. Dazu wird auch hier wieder zuerst das “autogen.sh” Skript ausgeführt.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce --disable-pthread
```

Nachdem dies erfolgreich ausgeführt wurde können nun auch die gleichen zwei Schritte wie bei den vorhergegangenen Programmen ausgeführt werden.

2.2.7 *edje*

Auch hier gilt wieder, Dateien herunterladen.

```
svn co http://svn.enlightenment.org/svn/e/trunk/embryo
```

Nachdem die Dateien heruntergeladen wurden, muss auch hier wieder das Skript aufgerufen werden.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce
```

Die letzten beiden Schritte sind auch hier wieder kompilieren und installieren.

```
make ; make install
```

2.2.8 elementary

Zuerst müssen auch hier die benötigten Daten heruntergeladen werden.

```
svn co http://svn.enlightenment.org/svn/e/trunk/TMP/st/elementary
```

Nun muss auch wieder das “autogen.sh” Skript heruntergeladen werden.

```
./autogen.sh --prefix=$WINCE_PATH --host=arm-mingw32ce --with-edje-cc=$WINCE_PATH/bi
```

Um zwei Fehlern vorzubeugen, welche beim erstellen der Test-Files von *elementary* auftreten muss man im in der Datei “Makefile.am” im Ordner “src/bin/” alle Vorkommnisse von “test_fileselector.c” entfernen und folgende Zeilen auskommentieren.

```
bin_PROGRAMS = elementary_test
if BUILD_QUICKLAUNCH
bin_PROGRAMS += elementary_quicklaunch elementary_run elementary_testql
endif
```

Nun kann das Programm auf gewohnte Art und Weise erstellt und installiert werden.

```
make ; make install
```

2.2.9 Weitere Schritte

Im Anschluss an das Erstellen dieser Programme muss nun noch ein Skript in WINCE_PATH angelegt und dessen Zugriffsrechte abgeändert werden.

```
touch efl_zip.sh
chmod 774 efl_zip.sh
```

In dieses Skript wird nun der Code eingefügt, welcher unter Anhang 4 zu finden ist. Bei Ausführung dieses Skripts werden die vorhandenen *DLL*'s nocheinmal komprimiert und alles in einen Ordner mit dem Namen “efl” kopiert. Im Anschluss wird der ganze Ordner noch in einem *Zip-Archiv* komprimiert. Möchte man nun noch eigene Anwendungen hinzufügen, so muss man diese nur in diesen “efl” Ordner hinzufügen und erneut komprimieren. Nun kann dieses Archiv auf das Mobile Gerät kopiert und entpackt werden.

3 Friend Finder

Die Software die als praktischer Teil dieser Abschlussarbeit entwickelt wurde hat den Namen *Friend Finder*. Im folgenden werden zuerst die verschiedenen Features des Programms vorgestellt. Im Anschluss werde die verwendeten Verfahren und Bibliotheken, welche zur Realisierung des Projektes verwendet wurden, erläutert. Im Abschnitt Implementierung wird kurz auf den Aufbau des Programmes eingegangen. Zuletzt wird unter Analyse der gemessene Datenverkehr des *Friend Finders* besprochen.

3.1 Features

Friend Finder hat bietet verschiedene Features, welche vom Nutzer angewendet werden können und komplett in C implementiert wurden. Bei diesen Features handelt es sich um:

- Versenden von Nachrichten
- Versenden der eigenen Position
- Anzeigen der Position von anderen Teilnehmern
- Erstellen eines 2D-Barcodes

3.1.1 Nachrichten versenden

Der Nutzer kann hiermit Nachrichten verfassen und an einen, vorher festgelegten, Freund senden. Dieser kann wiederum auf diese Nachrichten antworten. Die versendeten Textnachrichten werden vom Programm, mit einem vorher festgelegten *private key* verschlüsselt. Damit nun die andere Person die erhaltene Nachricht entschlüsseln kann, muss er den selben *private key* besitzen.

3.1.2 Eigene Position senden

Mit *Friend Finder* kann man die eigene Position im *Latitude/Longitude* Format versenden. Auch hier werden die gesendeten Daten mit symmetrischen Verschlüsselungsverfahren für dritte unleserlich gemacht.

3.1.3 Position anderer Teilnehmer anzeigen

Zusätzlich zum senden der eigenen Position, kann man sich auch die Positionen der anderen Teilnehmer anzeigen lassen. Hierbei kann man wählen, ob man alle Nutzer innerhalb eines Radius von 100, 250 oder 1000 Metern sehen möchte. Dieser Radius geht von der momentanen Position des Nutzers aus.

3.1.4 2D-Barcode

Da das Programm ein symmetrisches Verfahren anwendet, stellt sich die Frage wie man den *private key* an andere Personen weitergeben kann, ohne dass dritte diesen auch erhalten können. Hier wäre ein Ansatz, dass man aus einer Zeichenketten einen *2D-Barcode* erstellt und ihn von anderen Nutzern abfotographieren lässt. Diese können aus dem erhaltenen Barcode nun wieder die ursprüngliche Zeichenkette erstellen und haben somit den *private key* erhalten.

Dieser Programmteil kann anhand einer solchen beliebigen Zeichenkette einen Barcode erstellen und ausgeben. Dieser könnte dann, wie schon erwähnt, mit einer entsprechenden anderen Software fotografiert und wieder umgewandelt werden.

3.2 Verwendete Verfahren und Bibliotheken

Um die Software, welche im Rahmen dieser Bachelor-Arbeit realisiert wurde, zu implementieren war die Nutzung verschiedener Bibliotheken nötig. Im folgenden werden die verwendeten Bibliotheken kurz erläutert.

3.2.1 Versenden der Nachrichten

Um Nachrichten zu versenden wurde für dieses Projekt die *IRC-Client Bibliothek*[libircclient] verwendet. Diese *library* bietet verschiedene Funktionen um eine Verbindung mit einem *IRC-Server* zu erstellen und Nachrichten an diesen zu senden, sowie eingehende Nachrichten zu empfangen. Des weiteren ist es mit Hilfe dieser Bibliothek auch möglich, Nachrichten zu versenden.

3.2.2 Verschlüsselung der Daten

Zur Verschlüsselung der gesendeten Nachrichten wird die Bibliothek des *OpenSSL-Projekts*[?] verwendet. Hierzu werden die Daten mit dem *Blowfish-Algorithmus* verschlüsselt. Bei diesem Algorithmus handelt es sich um ein symmetrisches Verfahren, bei welchem alle Teilnehmer den gleichen privaten Schlüssel zum ver- sowie entschlüsseln nutzen. Die genutzte Implementierung entstammt der *libcrypto* Bibliothek.

3.2.3 Erzeugen eines 2D-Barcodes

3.2.4 Grafisches Benutzeroberfläche

Die Graphische Nutzeroberfläche wurde mit Hilfe von Bibliotheken aus dem *Elementary*-Paket realisiert. Insbesondere sind wurden *Elementary* und *Evas* genutzt. *Elementary* ist eine Sammlung von *Widgets* für Benutzeroberflächen, welche einfach und schnell zu implementieren sind. Um das zeichnen der Elemente auf verschiedenen Systemen kümmert sich *Evas*.

3.3 Implementierung

Das Programm *Friend Finder* ist so konzipiert, dass die Graphische Darstellung ohne großen Aufwand vom den restlichen Teilen der Software abgekoppelt werden kann und durch eine andere Bibliothek ersetzt werden kann. Dies hat zum Vorteil, dass man *Enlightenment* durch eine andere Art der Darstellung austauschen kann, ohne dabei die Funktionalität der zugrunde liegenden Komponenten zu zerstören. Somit ist, was die Art der Darstellung angeht, schon von vornherein einfacher möglich die Software auf unterschiedlichen Plattformen zu konvertieren.

Friend Finder ist an sich in fünf Submodule aufgeteilt. Diese wären

1. Graphische Benutzeroberfläche
2. Versenden von Textnachrichten
3. Versenden der eigenen Position
4. Empfangen der eigenen Position
5. Erstellen eines Barcodes

Graphische Benutzeroberfläche

Zum erstellen der Oberfläche wurde, wie schon erwähnt, *Enlightenment* verwendet. Durch diese Bibliothek werden alle benötigten Funktionen bereitgestellt. Der gesamte Programmcode der Nutzeroberfläche wurde in eine Datei geschrieben (*gui.c*). Diese Tatsache macht die erhaltung der Modularität einfacher, da man einfach nur diese Datei durch eine andere ersetzen muss um eine anderen Typ von Oberfläche zu nutzen.

Inner der Datei *gui.c* sind alle Funktionen enthalten um die Oberflächenelement zu erzeugen und zu platzieren. Um die gewünschte Funktionalität der einzelnen Elemente zu realisieren wurden auch die Aufrufe der benötigten Funktionen aus anderen Modulen in dieser Datei implementiert.

Versenden von Textnachrichten

In der Datei *msg_sender.c* sind alle Funktionen und Aufrufe implementiert, welche nötig sind um die Verbindung zum *IRC-Server* zu erstellen und die Nachrichten zu verschicken. Um eine Verbindung zu einem gegebenen *IRC-Server* zu erstellen muss eine *IRC-Session* initialisiert werden. Diese *Session* beinhaltet Informationen wie zum Beispiel den *Nickname* des Benutzers oder die *IP-Adresse* des Servers. Nachdem diese *Session* gestartet wurde, kann man nun durch das Aufrufen der Funktion `”set_txt_msg(char* msg)”` die Nachricht versenden. Wird eine Nachricht empfangen so wird diese an die Funktion `”show_message(char* msg)”`, welcher zur Benutzeroberfläche gehört, übergeben. Bei der Implementierung des Nachrichtenversandes ist eine Besonderheit zu erwähnen. Das genutzte Verschlüsselungsverfahren *Blowfish* wurde seitens der *OpenSSL* Bibliothek als *Blockcipher* implementiert. Das bedeutet, dass immer nur maximal 64 Bit Nachrichten verschlüsselt werden können. Da in der Programmiersprache *C* dies genau acht Zeichen entspricht, werden alle zu sendenden Nachrichten in Blöcke der Größe acht aufgeteilt, versendet und beim Empfänger wieder zusammengesetzt.

Ein weiterer wichtiger Unterschied zu den Modulen Senden und Empfangen von *GPS*-Positionen ist die Tatsache, dass bei diesem Programmteil Sender und Empfänger in der gleichen Datei implementiert wurden. Der Grund hierfür ist, dass man hier nicht zwischen mehreren Sendern oder Empfängern unterscheiden muss, und diese zwei Teile hier somit nicht komplett getrennt voneinander arbeiten müssen.

Versenden der eigenen Position

Der benötigte Programmcode zum versenden der eigenen Position ist in der Datei *sender.c* zu finden. Auch hier muss zuerst eine *IRC-Session* initialisiert werden um danach die Position zu versenden. Der Ablauf beim senden der Positionen erfolgt in einer vorgegebenen Reihenfolge. Zuerst wird der verschlüsselte Längengrad, danach der verschlüsselte Breitengrad gesendet. Daraufhin werden solange keine Daten mehr gesendet, bis der Empfänger eine Bestätigung an den *IRC-Kanal* sendet. Diese Bestätigung ist unverschlüsselt und stellt einfach nur den *Nickname* des Senders dar. Kommt dieses *Acknowledgement* beim Sender an, so versendet dieser wieder ein *Latitude/Longitude Paar*.

Empfangen der eigenen Position

noch nicht final....empfänger empfängt nachricht, ordnet sie, leitet sie zum zeichnen weiter....grob gesehen

Erstellen eines Barcodes

Die Datei *barcode.c* beinhaltet die Funktionen zum erstellen eines 2D-Barcodes. Hierzu wird die Funktion `generate_barcode(char* key)`, mit einer Zeichenkette als Übergabeparameter, aufgerufen. Aus dieser Zeichenkette wird dann ein Barcode erstellt, welcher im darauf folgenden Schritt als *.png* Datei auf das Speichermedium geschrieben wird.

3.4 Analyse

4 Ausblick

Die Software *Friend Finder* zeigt nur einen recht einfachen Ansatz auf, um Daten sicher zu versenden und die Position von anderen Benutzern aufzuzeigen. Im nun folgenden Teil dieser Arbeit werden Ausblicke auf weitere Möglichkeiten und Ansätze, für Nachrichtenverschlüsselung und *location awareness*, aufgezeigt.

4.1 Plattformunabhängigkeit

Für mobile Systeme existieren zum momentanen Zeitpunkt mehrere verschiedene Betriebssysteme, welche zum Teil andere Ansätze verfolgen. Es werden nun die drei bekanntesten Vertreter der Betriebssysteme, für Smart Phones, vorgestellt und es wird erläutert werden, inwiefern sie sich zur portierung von Software eignen.

4.1.1 Windows Mobile

Der wohl bekannteste Vertreter ist *Windows Mobile*. Die aktuelle Version 6.5 wurde von Microsoft auch *Windows Phone* betitelt. Das gesamte Betriebssystem basiert auf der *Windows Win32 API* und lässt Ähnlichkeiten zu den Desktop-Varianten der *Windows*-Familie erkennen.

Will man für dieses Betriebssystem Anwendungen entwickeln so bietet Microsoft eine eigenes *Software Development Kit (SDK)* an, welches auch jeder frei nutzen kann. Bei der Programmierung kann hierbei sowohl auf *C\C++*, sowie auch auf Java zurückgegriffen werden. Allerdings ist die *Win32 API* nicht kompatibel mit der Desktopversion, weshalb Anwendungen getrennt entwickelt oder portiert werden müssen.

Wie im Kapitel Tutorial schon erwähnt, lässt sich mit Hilfe eines *Cross-Compilers* das *Enlightenment*-Paket für dieses System, mit etwas Aufwand, portieren. Durch diese Tatsache und der Unterstützung von *C\C++* eignet sich Windows Mobile sehr gut als Plattform für Anwendungen, welche nicht unbedingt von vorne herein für diese entwickelt wurden.

4.1.2 Android

Bei *Android* [Android] handelt es sich um ein neueres Betriebssystem für Smart Phones. Das von Google entwickelte System setzt auf einen Linux-Kernel der Version 2.6 auf. Dieser Kernel

kümmert sich um die Prozess- und Speicherverwaltung, Kommunikation sowie um die Hardwareabstraktion.

Zum implementieren von Anwendungen stellt Google eigens ein eigenes *SDK* bereit. Dieses greift allerdings nur auf *Java*-Bibliotheken zurück, womit sich die nutzbaren Sprachen im moment eben auf diese beschränken. Somit können Programme die in *C* oder *C++* geschrieben wurden nicht portiert werden. Auch wenn man einen passenden *Cross-Compiler* nutzt funktionieren ist die Portierung nicht immer möglich, da Google die *libc*-Bibliothek (unter *Android* nun *Bionic* genannt) an mobile Geräte angepasst und verändert hat.

Durch diese starken Einschränkungen ist es somit auch nicht wirklich möglich das *Elementary*-Paket für *Android* zu compilieren, da dieses komplett in *C* geschrieben wurde. Somit muss Software, um auf einem *Android*-Handy ohne Probleme lauffähig zu sein, in *Java* geschrieben sein. Als graphische Darstellung kann der Entwickler die von Google bereitgestellten *.GUI*-Elemente nutzen.

4.1.3 WebOS

WebOS [WebOS] gehört nicht zu den weit verarbeiteten Betriebssystemen, allerdings wird es hier aufgeführt, da *Enlightenment* portiert werden kann. Das System wurde von *Palm* als Nachfolger von *PalmOS* entwickelt. Momentan ist das System nur auf zwei Geräten zu finden: Auf dem *Palm Pre* und dem *Palm Pixi*.

Für dieses Betriebssystem existiert sowohl ein *SDK* für *HTML5*, *CSS* und *Java* sowie ein weiteres, welches im März 2010 veröffentlicht wird, für *C* und *C++*. Hiermit werden mehrere Programmiersprachen unterstützt und die Portierungsmöglichkeiten dieser Plattform gewinnen erheblich an Attraktivität.

Wie bereits erwähnt, wurde *Enlightenment*, genauer gesagt *Evas*, bereits erfolgreich für *WebOS* compiliert und ausgeführt. Mit dem *C SDK* und dieser Portierung wäre es so ohne weitere Probleme möglich den *Friend Finder* auf einem *WebOS*-Gerät auszuführen.

4.2 Kryptographische Verfahren auf Mobilien Plattformen

In diesem Absatz werden mögliche kryptographische Verfahren für Smart Phones behandelt. So wird prinzipiell zwischen zwei verschiedenen Arten der Verschlüsselung unterschieden. Zum einen existieren die sogenannten symmetrischen und zum anderen die asymmetrischen Verfahren.

Bei ersterem existiert nur ein *private-key* welcher von allen Teilnehmern zum ver- sowie entschlüsseln genutzt wird. Bei den asymmetrischen Verfahren sind mehrere Schlüssel nötig. Hierbei werden Daten mit einem öffentlichen Schlüssel verschlüsselt und können nur mit dem dazu passenden privaten Schlüssel wieder entschlüsselt werden.

Im folgenden werden verschiedene Algorithmen und deren Tauglichkeit vorgestellt. Des weiteren wird auf Möglichkeiten eingegangen werden, um Schlüssel zu verteilen.

4.2.1 Symmetrische Verschlüsselungsverfahren

4.2.2 Asymmetrische Verschlüsselungsverfahren

4.3 Alternative location awareness Verfahren

4.4 Zusammenfassung

Anhang

Anhang 1

Anhang 2

Archive für *eet*:

- `zlib-1.2.3-bin.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/zlib-1.2.3/zlib-1.2.3-bin.tar.bz2/download>
- `zlib-1.2.3-dev.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/zlib-1.2.3/zlib-1.2.3-dev.tar.bz2/download>
- `libjpeg-6b-bin.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/libjpeg-6b/libjpeg-6b-bin.tar.bz2/download>
- `libjpeg-6b-dev.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/libjpeg-6b/libjpeg-6b-dev.tar.bz2/download>

Anhang 3

Archive für *evas*:

- `freetype-2.3.7-bin.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/freetype-2.3.7/freetype-2.3.7-bin.tar.bz2/download>
- `freetype-2.3.7-dev.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/freetype-2.3.7/freetype-2.3.7-dev.tar.bz2/download>
- `libpng-1.2.33-bin.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/libpng-1.2.33/libpng-1.2.33-bin.tar.bz2/download>
- `libpng-1.2.33-dev.tar.bz2`: <http://sourceforge.net/projects/cegcc/files/ported%20packages/libpng-1.2.33/libpng-1.2.33-dev.tar.bz2/download>

Anhang 4

`efl_zip.sh`:

```
#!/bin/sh

rm -rf efl/
rm -f efl.zip

mkdir -p efl/eina/mp
mkdir -p efl/evas/modules/engines/buffer/mingw32ce-arm/
mkdir -p efl/evas/modules/engines/software_16/mingw32ce-arm/
mkdir -p efl/evas/modules/engines/software_16_wince/mingw32ce-arm/
mkdir -p efl/evas/modules/engines/software_generic/mingw32ce-arm/
mkdir -p efl/evas/modules/loaders/eet/mingw32ce-arm/
mkdir -p efl/evas/modules/loaders/jpeg/mingw32ce-arm/
mkdir -p efl/evas/modules/loaders/pmaps/mingw32ce-arm/
mkdir -p efl/evas/modules/loaders/png/mingw32ce-arm/
mkdir -p efl/evas/modules/loaders/xpm/mingw32ce-arm/
mkdir -p efl/evas/modules/savers/eet/mingw32ce-arm/
mkdir -p efl/evas/modules/savers/png/mingw32ce-arm/

cp bin/eet.exe efl/
cp bin/libdl-0.dll efl/
cp bin/libevil-0.dll efl/
cp bin/libeina-0.dll efl/
cp bin/libeet-1.dll efl/
cp bin/libevas-0.dll efl/
cp bin/libecore-0.dll efl/
cp bin/libecore_evas-0.dll efl/
cp bin/libecore_job-0.dll efl/
cp bin/libecore_wince-0.dll efl/
cp bin/libembryo-0.dll efl/
cp bin/libedje-0.dll efl/

arm-mingw32ce-strip efl/libdl-0.dll
arm-mingw32ce-strip efl/libevil-0.dll
arm-mingw32ce-strip efl/libeina-0.dll
arm-mingw32ce-strip efl/libeet-1.dll
arm-mingw32ce-strip efl/libevas-0.dll
arm-mingw32ce-strip efl/libecore-0.dll
arm-mingw32ce-strip efl/libecore_evas-0.dll
arm-mingw32ce-strip efl/libecore_job-0.dll
arm-mingw32ce-strip efl/libecore_wince-0.dll
arm-mingw32ce-strip efl/libembryo-0.dll
arm-mingw32ce-strip efl/libedje-0.dll
```

```
cp lib/eina/mp/eina_chained_mempool.dll efl/eina/mp
cp lib/eina/mp/eina_fixed_bitmap.dll efl/eina/mp
cp lib/eina/mp/eina_pass_through.dll efl/eina/mp
```

```
arm-mingw32ce-strip efl/eina/mp/eina_chained_mempool.dll
arm-mingw32ce-strip efl/eina/mp/eina_fixed_bitmap.dll
arm-mingw32ce-strip efl/eina/mp/eina_pass_through.dll
```

```
cp lib/evas/modules/engines/buffer/mingw32ce-arm/module.dll efl/evas/modules/engines/
cp lib/evas/modules/engines/software_16/mingw32ce-arm/module.dll efl/evas/modules/eng
cp lib/evas/modules/engines/software_16_wince/mingw32ce-arm/module.dll efl/evas/modul
cp lib/evas/modules/engines/software_generic/mingw32ce-arm/module.dll efl/evas/module
```

```
cp lib/evas/modules/loaders/eet/mingw32ce-arm/module.dll efl/evas/modules/loaders/eet
cp lib/evas/modules/loaders/jpeg/mingw32ce-arm/module.dll efl/evas/modules/loaders/jp
cp lib/evas/modules/loaders/pmaps/mingw32ce-arm/module.dll efl/evas/modules/loaders/p
cp lib/evas/modules/loaders/png/mingw32ce-arm/module.dll efl/evas/modules/loaders/png
cp lib/evas/modules/loaders/xpm/mingw32ce-arm/module.dll efl/evas/modules/loaders/xpm
```

```
cp lib/evas/modules/savers/eet/mingw32ce-arm/module.dll efl/evas/modules/savers/eet/m
cp lib/evas/modules/savers/png/mingw32ce-arm/module.dll efl/evas/modules/savers/png/m
```

```
arm-mingw32ce-strip efl/evas/modules/engines/buffer/mingw32ce-arm/engine_buffer.dll
arm-mingw32ce-strip efl/evas/modules/engines/software_16/mingw32ce-arm/engine_softwar
arm-mingw32ce-strip efl/evas/modules/engines/software_16_wince/mingw32ce-arm/engine_s
arm-mingw32ce-strip efl/evas/modules/engines/software_generic/mingw32ce-arm/engine_so
```

```
arm-mingw32ce-strip efl/evas/modules/loaders/eet/mingw32ce-arm/loader_eet.dll
arm-mingw32ce-strip efl/evas/modules/loaders/jpeg/mingw32ce-arm/loader_jpeg.dll
arm-mingw32ce-strip efl/evas/modules/loaders/pmaps/mingw32ce-arm/loader_pmaps.dll
arm-mingw32ce-strip efl/evas/modules/loaders/png/mingw32ce-arm/loader_png.dll
arm-mingw32ce-strip efl/evas/modules/loaders/xpm/mingw32ce-arm/loader_xpm.dll
```

```
arm-mingw32ce-strip efl/evas/modules/savers/eet/mingw32ce-arm/saver_eet.dll
arm-mingw32ce-strip efl/evas/modules/savers/png/mingw32ce-arm/saver_png.dll
```

```
cp freetype-2.3.7-bin/bin/libfreetype-6.dll efl/
cp libjpeg-6b-bin/bin/jpeg62.dll efl/
cp libpng-1.2.33-bin/bin/libpng12-0.dll efl/
cp libpng-1.2.33-bin/bin/libpng-3.dll efl/
cp zlib-1.2.3-bin/bin/zlib1.dll efl/
```

```
zip -r -9 efl.zip efl/
```


Literatur

- [Android] *Android*. – URL <http://www.android.com/>. – [Online; letzter Aufruf 25.01.2010]
- [cegcc] *CeGCC*. – URL <http://cegcc.sourceforge.net/>. – [Online; letzter Aufruf 20.11.2009]
- [efl] *Enlightenment*. – URL <http://www.enlightenment.org/>. – [Online; letzter Aufruf 20.11.2009]
- [openssl] *libCrypto*. – URL <http://www.openssl.org/>. – [Online; letzter Aufruf 25.01.2010]
- [libircclient] *libircclient*. – URL <http://libircclient.sourceforge.net/>. – [Online; letzter Aufruf 25.01.2010]
- [PalmOS] *PalmOS*. – URL <http://www.palm.com/>. – [Online; letzter Aufruf 25.01.2010]
- [WebOS] *WebOS*. – URL <http://palmwebos.org/>. – [Online; letzter Aufruf 25.01.2010]
- [Windows] *Windows Mobile*. – URL <http://www.microsoft.com/windowsmobile/de-de/default.aspx>. – [Online; letzter Aufruf 25.01.2010]