HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

ita Institute for Internet Technologies and Applications

strongSwan VPN Plasmoid unter KDE

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2010

Autor:	Maurus Rohrer
Betreuer:	Prof. Dr. Andreas Steffen
Projektpartner:	revosec AG, Wangs, SG
Experte:	Dr. Ralf Hauser
Gegenleser:	Prof. Dr. Lothar Müller

ERKLÄRUNG

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum: Name:

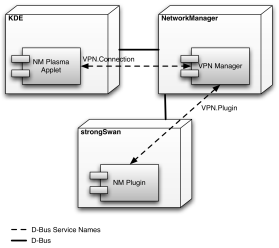
. . . . . .

Unterschrift

Maurus
Rohrer

Diplomand	Maurus Rohrer
Examinator	Prof. Dr. Andreas Steffen
Experte	Dr. Ralf Hauser, PrivaSphere AG, Zürich, ZH
Themengebiet	Internet-Technologien und -Anwendungen
Projektpartner	revosec AG, Wangs, SG

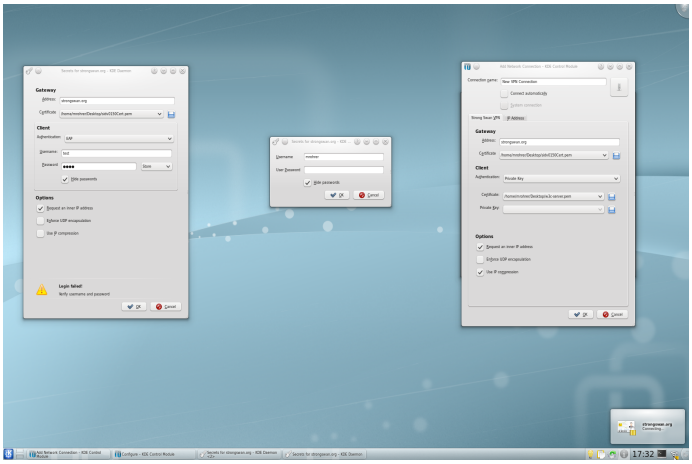
strongSwan VPN Plasmoid unter KDE



Ausgangslage: Die Open Source strongSwan Software ist eine weit verbreitete IPsec- basierte VPN Lösung. Ein zentraler Bestandteil des strongSwan Projekts ist das NetworkManager Plugin für den Gnome Linux Desktop. Dieses Plugin ermöglicht es den Benutzern "Remote Access" VPN Verbindungen einfach einzurichten und zu verwalten. Da für die Benutzeroberfläche KDE noch kein Plugin für strongSwan existierte, war es Ziel dieser Arbeit ein solches für die Netzwerkumgebung von KDE zu realisieren.

Vorgehen/Technologien: Damit eine strongSwan VPN Verbindung mit der KDE Plasmoid Benutzeroberfläche aufgebaut werden kann, müssen die Komponenten NM Plasma Applet (User Interface), NetworkManager (Netzwerklogik) und strongSwan NM Plugin (IKEv2 Daemon) über den D - Bus miteinander verknüpft werden. Fundierte Kenntnisse dieser Komponenten waren eine äusserst wichtige Voraussetzung für die Umsetzung der Ziele dieser Arbeit. Ein Grossteil der Implementierung (C + + / QT) wurde im NM Plasma Applet realisiert, das neu in KDE 4 eingeführt wurde und zurzeit noch im Review- Status ist. Ein neues Fehlermeldungskonzept verlangte zudem Erweiterungen und Anpassungen in allen Komponenten.

Ergebnis: Das KDE 4 NM Plasma Applet wurde erfolgreich für den Gebrauch des strongSwan Daemons erweitert. Es ist nun möglich, über die gewohnte Netzwerkumgebung in KDE eine strongSwan Verbindung zu konfigurieren. Neu im NM Plasma Applet wurde die dynamische Abfrage des Benutzer- Passwortes eingeführt. Weiter werden nun Statusänderungen einer VPN Verbindung visualisiert. Zusätzlich wurden diverse Bugs im NM Plasma Applet behoben. Ebenfalls lassen sich erstmals Fehlermeldungen beim Verbindungsaufbau detailliert und klar verständlich darstellen. Diese basieren auf dem für dieses Projekt speziell entwickelten Fehlermeldungskonzept.



EIN INSTITUT DER
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Institute for
Internet Technologies
and Applications

strongSwan VPN Plasmoid unter KDE

Student: Maurus Rohrer

Betreuer: Prof. Dr. Andreas Steffen

Industrieparter: Martin Willi, revosec AG

Ausgabe: Dienstag, 23. Februar 2010

Abgabe: Freitag, 18. Juni 2010

Einführung

Das mit Ubuntu 9.10 Karmic Koala unter dem GNOME Desktop verfügbare strongSwan NetworkManager Applet hat innerhalb kürzerster Zeit eine grosse Popularität erlangt und wird zum Beispiel an der Albert-Ludwigs-Universität in Freiburg im Breisgau für den Remote Access eingesetzt.

Im Rahmen dieser Arbeit soll für den KDE 4.0 Plasma Desktop, der bevorzugt in den OpenSUSE und Kubuntu Projekten eingesetzt wird, ein benutzerfreundliches VPN Plasmoid entwickelt werden, dass über eine DBUS Schnittstelle den strongSwan IKEv2 Dämon konfigurieren, sowie starten und stoppen kann. Dabei soll wahlweise eine Zertifikats- oder Passwortbasierte Benutzerauthentisierung möglich sein.

Aufgabenstellung

- Einarbeitung in die KDE/Plasma Umgebung, sowie die D-Bus und strongSwan Plugin Schnittstellen.
- Erfassen des Enwicklungstands des KDE NetworkManagers. Kontaktaufnahme mit den Entwicklern.
- Plasmoid-basierte Konfiguration der Benutzerauthentisierung mittels EAP Username/Passwort, sowie eines Serverzertifikat für die Gegenstelle.
- Anzeige des Verbindungszustands (VPN Tunnel established) im KDE Network Manager und wenn möglich auch im Icon.
- Erfassung von relevanten Fehlern im charon Daemon (z.B. via bestehende oder neu zu definierende Alerts), Signalisierung des Fehlergrunds via D-Bus und benutzergerechte Visualisierung des Fehlers durch das Plasmoid.

Optional

- Plasmoid-basierte Konfiguration der Benutzerauthentisierung mittels RSA Private Key und persönlichem X.509 Zertifikat.

Links

[1] Plasma Developer Wiki
http://techbase.kde.org/Projects/Plasma

[2] Plasma (KDE) on Wikipedia
http://en.wikipedia.org/wiki/Plasma_(KDE)

[3] D-Bus Project
http://www.freedesktop.org/wiki/Software/dbus

[4] D-Bus on Wikipedia
http://en.wikipedia.org/wiki/DBUS

[5] strongSwan Project
http://www.strongswan.org

[6] strongSwan Wiki
http://wiki.strongswan.org

[7] **strongSwan NetworkManager Applet**

Rapperswil, .23. Februar 2010

Prof. Dr. Andreas Steffen

Inhaltsverzeichnis

Teil I

Technischer Bericht

1 Einleitung

1.1 Ausgangslage

Die vollständige IPsec-Implementierung strongSwan ist eine weit verbreitete Open Source VPN-Lösung. Als einer der beiden Nachfolger des FreeS/WAN-Projekts steht strongSwan unter der GNU General Public License. Das Projekt wird von Andreas Steffen, Professor für Sicherheit und Kommunikation an der HSR, betreut. Software Architekt und Hauptentwickler des IKEv2 Keying Daemons ist Martin Willi. NAT Traversal für IKEv2 wurde von Tobias Brunner und Daniel Röthlisberger beigesteuert [BR06].

Ein wichtiger Bestandteil des strongSwan Projekts ist das NetworkManager Plugin für Linux. Dieses Plugin ermöglicht den Benutzern VPN-Verbindung einfach einzurichten und zu verwalten. Viele VPN-Implementierungen werden von Leihen nicht verwendet, da sie zu kompliziert zu konfigurieren sind. Martin Willi hat ein benutzerfreundliches Plugin für die Gnome Desktop Oberfläche bereits realisiert. Dieses Plugin erlaubt den Linux Anwendern die Konfiguration ihrer strongSwan VPN-Verbindung über die gewohnte Netzwerkumgebung des Betriebssystems einzurichten und zu verwalten.

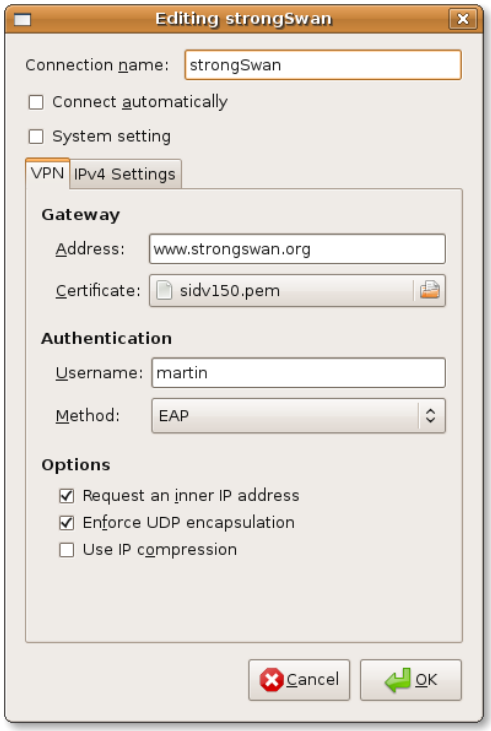


Abbildung 1.1: Gnome NetworkManager Plugin

Für die Benutzeroberfläche KDE, die weit verbreitet ist, existiert noch kein Plugin für strongSwan.

Ziel dieser Arbeit ist es ein strongSwan Plugin für die Netzwerkumgebung von KDE zu realisieren. Dieses sollte genau so intuitiv zu bedienen sein wie das Gnome Plugin. Der NetworkManager wird sowohl für Linux Systeme mit Gnome Oberflächen als auch für KDE Oberflächen benutzt. Er verwaltet die Netzwerkverbindungen des Linux Betriebssystems. Da der NetworkManager von Gnome Entwicklern realisiert wurde, war die Gnome Benutzeroberfläche (Gnome Applet) im Vergleich zu jener von KDE stabiler und ausgereifter. Durch die Einführung von KDE 4 wurde die KDE Benutzeroberfläche für den NetworkManager (NM Plasma Applet) neu implementiert. Diese neue Version wird noch nicht produktiv genutzt, da sie noch nicht ausreichend getestet wurde und einige Funktionalitäten noch fehlen. Sie verspricht jedoch eine sehr benutzerfreundliche und stabile Netzwerkumgebung zu werden. Ziel ist, dass bei Einführung des NM Plasma Applets ein vollumfängliches strongSwan Plugin zur Verfügung steht.

1.2 Aufgabenstellung

Die offizielle Aufgabenstellung ist am Anfang dieses Dokuments zu finden. Die konkrete Aufgabenstellung wurde erst im Laufe des Projektes festgelegt, da diese ursprünglich für ein 2er Team entworfen wurde. Weil ich die Arbeit alleine ausführte, wurden in den ersten zwei Sitzungen folgende Hauptziele festgelegt:

- strongSwan Integration in das NM Plasma Applet
- Konfiguration des strongSwan Daemons
- Starten des strongSwan Daemons
- Stoppen des strongSwan Daemons
- Benutzerauthentifizierung mittels Passwort (EAP)

Als mögliche Zusatzfunktionalität wurde ein Fehlermeldekonzept in Erwägung gezogen, das später auch für das Gnome Applet verwendbar währe. Vor diesem Projekt war es noch so, dass wenn keine Verbindung aufgebaut werden konnte, der Benutzer nicht über die Fehlerursache informiert wurde. Er musste die Logfiles analysieren um zu erkennen woran die Verbindung scheiterte. Die Fehlermeldungen, die in den Logfiles gespeichert werden, sollten dem Benutzer jedoch im idealen Fall auf verständliche Art kommuniziert werden.

In der Aufgabenstellung ist ein benutzerfreundliches Plasmoid gefordert. Das NM Plasma Applet bietet jedoch gerade bei VPN Verbindungen noch nicht alle Funktionalitäten. Es wurde entschieden, dass diverse Änderungen am NM Plasma Applet miteinbezogen werden sollten.

Optional ist Client Authentifizierung mittels Privat Key und Zertifikat.

1.3 Vorgehensweise und Aufbau der Dokumentation

Da ich keine Erfahrungen in der Linuxumgebungsentwicklung hatte, war eine relativ lange Einarbeitungszeit vorgesehen. Die bestehenden Architekturen sind komplex und leider sehr schlecht dokumentiert. Ein grosses Risiko dieser Arbeit ist, sich nicht genügend gut in die Frameworks einzuarbeiten, um mit ihnen produktiv arbeiten zu können.

Es wurde nach dem agilen Rational Unified Process (RUP) Verfahren vorgegangen. Die verschiedenen Arbeitsschritte: Anforderungsanalyse, Analyse und Design, Implementierung und Test wurden für alle erarbeiteten Software-Versionen durchgeführt. Iterativ wurden diese Arbeitsschritte durchgearbeitet und kontinuierlich überarbeitet. Die detaillierte Dokumentation zu den Arbeitsschritten ist in Teil II zu finden.

Die Tabelle 1.1 gibt einen Überblick in den Aufbau der Dokumentation:

Kapitel	**Beschreibung**
Einleitung	Am Anfang dieses Dokuments ist das Abstract zu finden, das die Arbeit kurz und kompakt zusammenfasst. Zudem befindet sich dort die offizielle Aufgabenstellung und der Eid zur Beglaubigung des eigenständigen Arbeitens.
Teil I : Technischer Bericht	Der erste Teil umfasst neben dieser Einführung ein Grundlagen Kapitel, in dem die nötigen Hintergründe der verwendeten Technologien und ihrer Architektur erläutert werden. Weiter werden die Überlegungen, Entscheide und nötigen Entwicklungen für das Realisieren und Entwickeln der Anforderungen dokumentiert. Abschliessend werden die Resultate präsentiert und eine persönliche Schlussfolgerung dargelegt.
Teil II : SW-Projektdokumentation	Im zweiten Teil der Dokumentation ist der Sotfware Prozess dokumentiert. Darin finden sich die Anforderungsspezifikationen sowie die Details der Entwicklung und UML Diagramme für die Visualisierung der getätigten Implementationen. Das Projektmanagement und die Tests sind auch im zweiten Teil dieser Arbeit zu finden.
Anhang	Im Anhang wird der Inhalt der mitgelieferten DVD abgebildet. Weiter sind Tabellen-, Abbildungs- und Literaturverzeichnis aufgeführt. Die Notfiy Messages des IKEv2 Standars und die NetworkManager D-Bus Schnittstellen sind auch im Anhang zu finden, sowie der persönliche Bericht.

Tabelle 1.1: Kapitel Übersicht

2 Grundlagen

Dieses Kapitel erläutert die Grundlagen, welche für die Realisierung der Arbeit notwendig sind. Das zu entwickelnde strongSwan VPN Plugin ist stark von den bestehenden Strukturen und Frameworks des Linux Betriebssystem abhängig, deshalb ist es wichtig, diese Strukturen und Frameworks zu verstehen. Als Erstes wird die Benutzeroberfläche von KDE beschrieben.

Danach wird das Networkmanagement Plasma Applet (NM Plasma Applet) analysiert. Dieses ist die grafische Oberfläche von KDE für die Verwaltung der Netzwerkfunktionalitäten. Das NM Plasma Applet wurde neu in der KDE Version 4 eingeführt. Es ist der Nachfolger des KNetwork-Managers, der grundlegend überarbeitet und verbessert wurde.

Es folgt ein Abschnitt über den NetworkManager, der die Netzwerkfunktionalität des Betriebssystems steuert. Der NetworkManager ist die Schnittstelle zwischen der Benutzeroberfläche NM Plasma Applet und den Treibern der Netzwerkkarten.

Danach wird die Funktionalität des D-Busses beschrieben. Der D-Bus ist eine wichtige Komponente dieser Arbeit, da er die verschiedenen involvierten Technologien verbindet. Die Kommunikation des strongSwan Daemons, dem NetworkManager und dem NM Plasma Applet läuft über den D-Bus. Es müssen die D-Bus Schnittstellendefinitionen eingehalten werden, damit die Komponenten miteinander kommunizieren können.

Abschliessend folgt ein Abschnitt über den strongSwan Daemon, respektive sein Plugin zum NetworkManager. Dieses definiert die Schnittstelle des strongSwan Daemons zum NetworkManager.

2.1 KDE

KDE ist ein internationales Open Source Projekt. Der Kern des KDE Projekts ist der KDE Workspace, eine Arbeitsoberfläche für Unix Betriebssysteme. Mittlerweile ist es möglich die KDE Arbeitsoberfläche auch auf Mac OS X oder Windows System zu benutzten. Neben dem KDE Workspace wird eine grosse Anzahl von Applikationen angeboten (von Office ähnlichen Anwendungen über Verwaltungstools bis hin zu Spielen).

Das KDE Projekt wurde 1996 von Matthias Ettrich gegründet und umfasst bis heute hunderte von aktiven Entwicklern. In der KDE Arbeitsoberfläche wird ausschliesslich in C++, mit den QT Bibliotheken von Trolltec entwickelt.

Durch die Einführung der KDE Version 4 wurde das Projekt mit dem Plasma Framework erweitert. Da die zu entwickelnde Software in Form eines Plasmoids gestaltet werden soll, ist es unumgänglich das Plasma Konstrukt kurz zu erklären.

2.1.1 Plasma

Plasma ist Bestandteil der “KDE Software Compilation 4”. Diese beinhaltet das “Desktop Enviroment” und diverse Userinterface Elemente. Das Plasma Framework ist für die Arbeitsfläche verantwortlich. Diese beinhaltet den Desktop, das Panel (auch Taskleiste genannt) und diverse kleinere Userinterface Elemente (Plasmoids). Ziel von Plasma ist eine flexible und benutzerorientierte Gestaltung der Arbeitsoberfläche. Dies bedeutet, dass Plasmoids (so zum Beispiel die Zeitanzeige) nach Belieben dargestellt und positioniert werden können. Es ist möglich die Zeitanzeige in der Taskleiste zu positionieren, was der gewohnten Anzeige in vielen Betriebssystemen entspricht. Es besteht jedoch auch die Möglichkeit die Zeitanzeige auf den Desktop zu ziehen und somit die Anzeige zu vergrössern und am gewünschten Ort zu positionieren. Dies ist möglich durch die neu eingeführte Architektur. Jedes Plasma Programm hat seine *DataEngine*, welche die Logikfunktionalität des Plasmoids realisiert. Verschiedene Plasmoids können nun diese *DataEngine* ansprechen und ihre eigene Implementation der Benutzeroberfläche realisieren. So ist es möglich, zum Beispiel die Zeitanzeige mit wenig Code als Digitaluhr oder Analoguhr auf dem Desktop darzustellen. [eK10e]

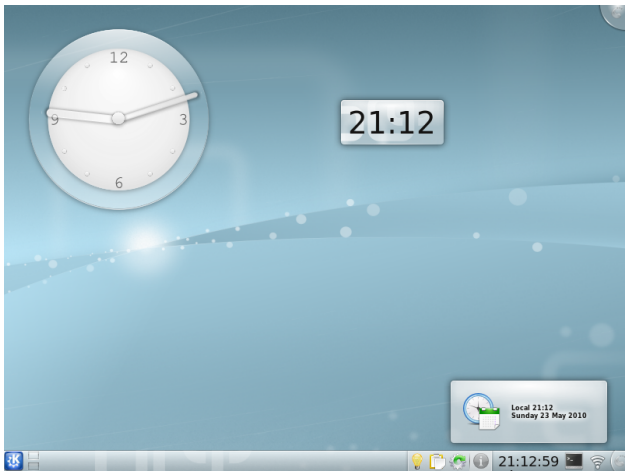


Abbildung 2.1: Plasma Desktop

2.1.2 QT

QT ist eine C++ Klassenbibliothek zur Programmierung grafischer Benutzeroberflächen. QT ist weitverbreitet und sie wird in Projekte wie Google Earth, Skype unter Linux, Mathematica von Wolfram oder dem Media Player VLC benutzt. Die QT Bibliotheken sind gut dokumentiert. QT liefert neben den Klassenbibliotheken auch einige Tools für die Entwicklung und das Gestalten von Benutzeroberflächen.

Speziell an QT ist das “Signal-Slot” Konzept. Dieses realisiert einen ereignisgesteuerten Programmfluss und bietet somit eine Alternative zu Rückruffunktionen (*Callbacks*). Die dynamische Prüfung der Aufrufparameter bezüglich ihres Typs ist einfacher und flexibler als *Callbacks*.

Signale sind Events, die beim Eintreten eines Ereignisses abgeschickt werden. Ein Slot ist eine gängige Funktion, die mit einem Signal verknüpft werden kann und auch durch dieses Signal ausgelöst wird. Slots und Signale sind zunächst lose. Erst durch die Verknüpfung der Funktion “connect” werden Signale den Slots zugeteilt. Jedes Mal, wenn das Signal abgeschickt wird,

wird der verbundene Slot aufgerufen. Ein Signal kann auch mit mehreren Slots verbunden sein, sodass beim Eintreten eines Ereignisses mehrere Funktionen aufgerufen werden. Ebenso kann ein Slot mit mehreren Signalen verbunden werden, wodurch dieselbe Funktion beim Auftreten unterschiedlicher Signale aufgerufen wird. [JM08]

Abbildung 2.2 zeigt eine mögliche Verknüpfung von Objekten mit dem Signal und Slot Konzept.

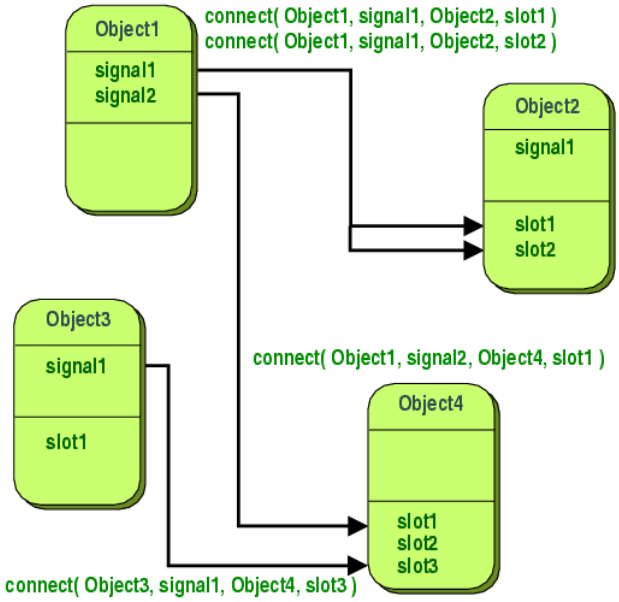


Abbildung 2.2: QT Signal Slot Konzept [JM08]

Die QT Klassenbibliothek ist in verschiedene Module aufgeteilt. Diese übernehmen verschiedene Funktionalitäten (von Datenbankzugriffen über Netzwerkfunktionen bis hin zu Multimediaobjekten). Für die Entwicklung des strongSwan Plugins werden folgende Module benötigt:

- QtCore - Kern-Klassen, die von allen anderen Modulen genutzt werden und Standarttypen definieren.
- QtGui - Komponenten, die der Gestaltung von grafischen Benutzeroberflächen dienen.
- QtTest - Werkzeuge zum Testen der eigenen Anwendungen.

2.2 NM Plasma Applet

Das NM Plasma Applet erlaubt den Benutzern die Netzwerkfunktionen über das KDE übliche Userinterface zu bedienen. Abbildung 2.3 zeigt das Icon des NM Plasma Applets, welches in der Taskleiste positioniert ist. Durch Klicken des Icons wird Abbildung 2.4 gestartet und die vorhandenen Verbindungen lassen sich nun steuern. Will man Verbindungen konfigurieren muss in Abbildung 2.4 “Manage Connections” angeklickt werden. Danach erscheint das Verwaltungsfenster des NM Plasma Applets (Abbildung 2.5).

Das abgebildete NM Plasma Applet ist noch in Entwicklung. Es wird jedoch schon bald in die Benutzeroberfläche von KDE integriert. Es gilt zu beachten, dass das NM Plasma Applet im Gegensatz zum Gnome Applet noch nicht über den vollen Funktionsumfang verfügt. Mehrere

Abbildung 2.3: NM Plasma Applet Taskleiste Icon

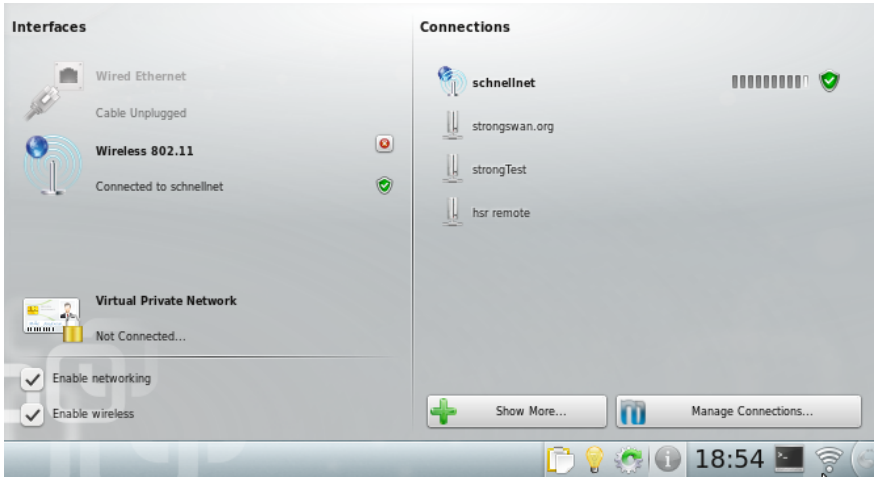


Abbildung 2.4: NM Plasma Applet Übersicht

aktive KDE Entwickler arbeiten jedoch intensiv an den Erweiterungen und Verbesserungen des NM Plasma Applets.

2.2.1 Funktionsumfang

Wichtig für die Analyse des NM Plasma Applets war der Funktionsumfang, sprich die Möglichkeit VPN Verbindungen zu verwalten und durch das Applet zu starten und zu stoppen. Folgende Funktionen sind in der jetzigen Version bereits vorhanden:

- Ethernet (IEEE 802.3) Verbindungen verwalten
- Wireless (IEEE 802.11) Verbindungen verwalten
- VPN Verbindungen verwalten
- DSL (PPPoE) Verbindungen verwalten
- Dial-Up (PPP) Verbindungen verwalten
- Mobile Broadband (GSM, CDMA, UMTS, usw.) Verbindungen verwalten
- Verbindungen starten und stoppen
- Anpassung des Icons anhand des Status der Verbindungen
- Automatisches Verbinden einer verfügbaren Verbindung
- Erkennung der Verschlüsselungsart von Wireless Netzwerken

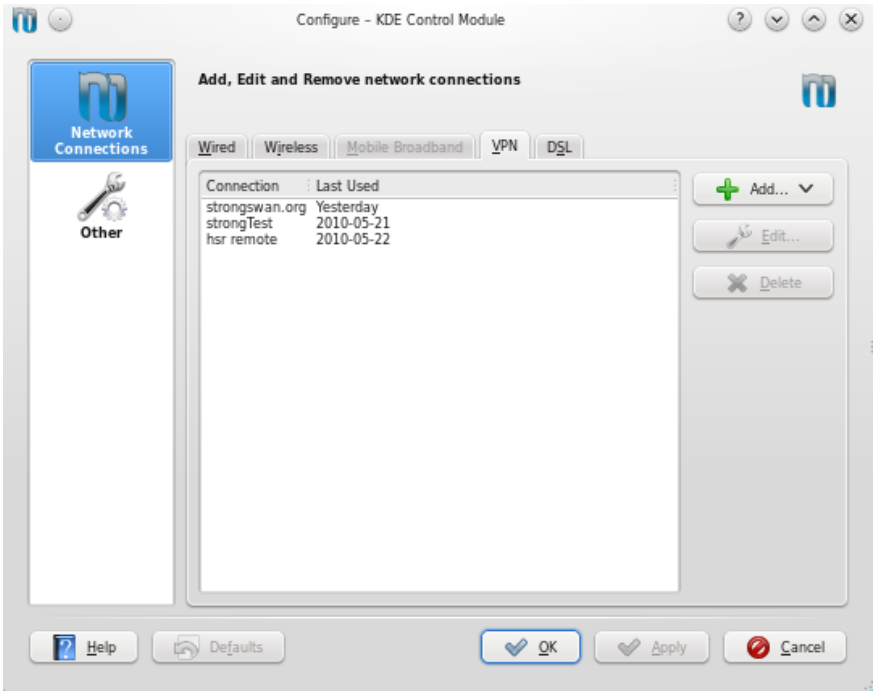


Abbildung 2.5: NM Plasma Applet Manager

Wichtig für diese Arbeit sind die Punkte; VPN Verbindungen verwalten, Verbindungen starten und stoppen. Da diese Punkte implementiert sind, ist es möglich VPN Verbindungen über das NM Plasma Applet zu verwalten und zu steuern. Es gibt jedoch auch Funktionen, die noch nicht implementiert sind und für ein intuitives und benutzerfreundliches Userinterface notwendig wären. Wenn das zu entwickelnde strongSwan Plugin in die NM Plasma Applet Umgebung eingebunden wird, müsste es um folgende Funktionen erweitert werden, damit die Anforderungen der Aufgabenstellung erfüllt werden können.

Fehlende Funktionen:

- Anpassung des Icons bei VPN Verbindungen
- Anzeige der Verbindungsdetails (IP Adresse, Gateway, usw.)
- Passwort-Abfrage bei Verbindungsaufbau
- Informationen und Rückmeldung bei missglücktem Verbindungsaufbau

2.2.2 Architektur

Die NM Plasma Applet Architektur scheint auf den ersten Blick komplex und überdimensioniert, insbesondere da doch “nur” eine Netzwerkverbindung gestartet oder gestoppt werden soll. Jedoch macht die Architektur durchaus Sinn und ist ganz den Neuerungen der KDE Version 4 angepasst. Damit die Netzwerkkarte und physikalischen Devices den Userinterface Elementen bekannt gemacht werden können, braucht es einen Vermittler. Dieser Vermittler ist der KDE Daemon (kded), welcher die Status Änderungen der physikalischen Devices dem Userinterface mitteilt. Der kded wird durch den NetworkManager, der die Devices abstrahiert, über Änderungen auf den Devices informiert. Der D-Bus ist zuständig für die Kommunikation zwischen dem kded

und dem NetworkManager. [Küg10]

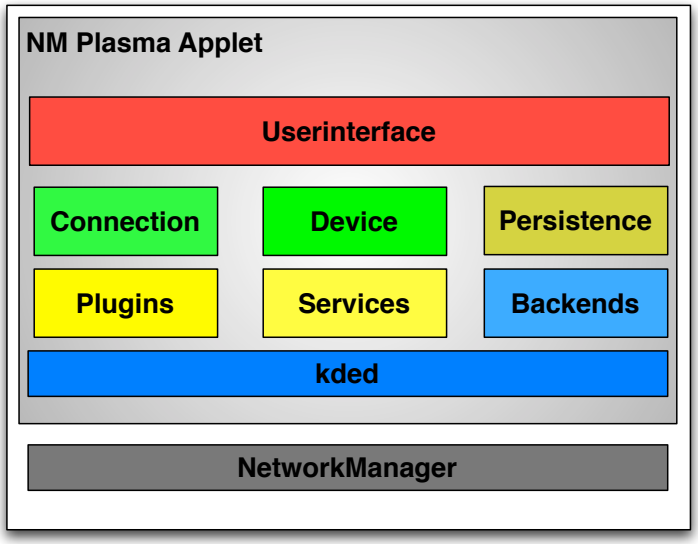


Abbildung 2.6: NM Plasma Architektur

Zudem bietet das NM Plasma Applet eine Möglichkeit Verbindungen auf der Festplatte zu speichern (**Persistence**) und diese dynamisch zu laden. Es wandelt die gespeicherten Informationen in Verbindungselemente. Diese sind immer an eine Userinterface-Element (**Userinterface**) gebunden. Dadurch lässt sich durch Anklicken eines Icons über das Verbindungselement **kded** weiter zum **NetworkManager**, bis hin zu den Treibern eine Verbindung aufbauen.

Zudem wird unterschieden zwischen Verbindungen (**Connection**) und Devices (**Device**). Devices repräsentieren Netzwerkkarten, während Verbindungen Softwareelemente mit Konfigurationsfiles repräsentieren. Verbindungen können sich in der Art ihrer Konfiguration unterscheiden, da man z.B. verschiedene kabellose Verbindungen über dieselbe Netzwerkkarte steuert. Somit kann ein Device mehrere Verbindungen zugeteilt haben. VPN Verbindungen werden im NM Plasma Applet als Verbindungen repräsentiert. Eine VPN Verbindung kann also erst gestartet werden, wenn auch ein Device verfügbar ist und dieses mit einem Netzwerk verbunden ist. Abbildung 2.7 visualisiert die verschiedenen Klassen und ihre Verknüpfungen.

Das NM Plasma Applet bietet eine Plugin Infrastruktur, in der verschiedene externe Komponenten einfach hinzugeladen und dem Applet zur Verfügung gestellt werden können. Das zu schreibende strongSwan Plugin wird ebenfalls durch diese Plugin Infrastruktur (**Plugins**) dem NM Plasma Applet hinzugefügt. **Services** definieren Abläufe, welche die verschiedenen Komponenten benutzten und abrufen. Im Packet **Backends** sind die D-Bus Schnittstellen des NetworkManagers untergebracht. Über die D-Bus Schnittstellen wird das NM Plasma Applet z.B. über den Statuswechsel einer Verbindung informiert oder ermöglicht dem NM Plasma Applet eine Verbindung aufzubauen.

Die im Plugin involvierten Klassen und ihre Beziehungen sind in Abschnitt 6 dokumentiert.

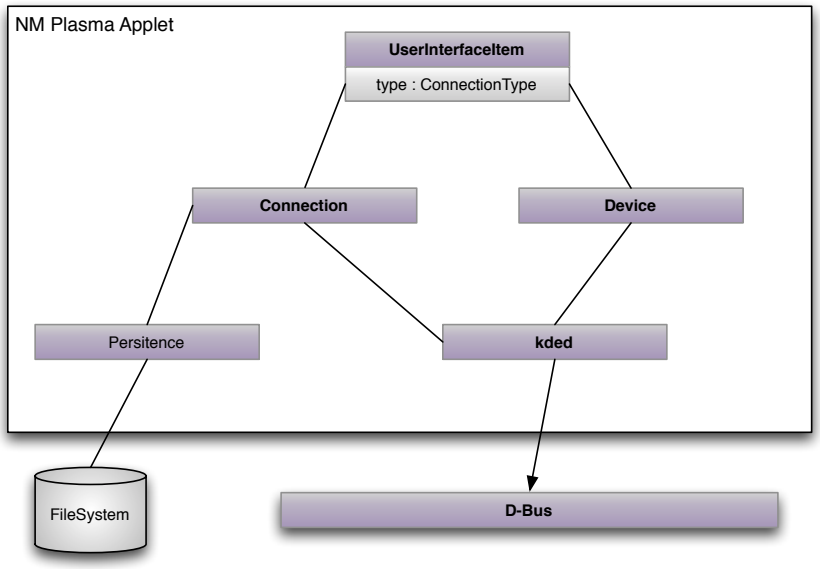


Abbildung 2.7: NM Plasma Applet: Connection und Device

2.3 NetworkManager

Der NetworkManager erlaubt es dem Benutzer eines Linuxsystems seine Netzwerkadministration über eine grafische Benutzeroberfläche zu verwalten. Der NetworkManager implementiert selbst keine Benutzeroberflächen, er stellte lediglich die Funktionen für eine grafische Darstellung zur Verfügung. Er dient als Bindeglied zwischen Hardware und Benutzeroberfläche. Der NetworkManager informiert das Userinterface über Änderungen auf den Devices oder startet eine neue Verbindung und ändert nach erfolgreichem Aufbau der Verbindung die Netzwerkkonfiguration auf dem Betriebssystem. Dadurch muss sich der Benutzer nicht mit Konfigurationsfiles und Shells um die Administration seines Netzwerks kümmern. Der NetworkManager wird von den meisten Linux Distributionen zu Verfügung gestellt.

2.3.1 Architektur

In Abbildung 2.8 ist die Architektur dargestellt, in welcher der NetworkManager untergebracht ist. Die Architektur besteht aus vier Komponenten. Die Kommunikation der verschiedenen Komponenten läuft über die D-Bus (Abschnitt 2.4) Schnittstelle. Durch die Verwendung der D-Bus Schnittstelle wird eine Loskoppelung realisiert und neue Komponenten können so einfach an die Architektur angeschlossen oder ersetzt werden.

Hardware Die Hardware Komponente repräsentiert die physikalischen Netzwerkschnittstellen sowie die dazugehörigen Treiber, die vom Linuxkernel geladen und verwaltet werden.

NetworkManager Der NetworkManager ist ein Daemon, der direkt an die physikalische Schnittstelle angeschlossen ist und auf Events der Hardware reagiert. So erkennt der NetworkManager wenn ein Ethernetkabel eingesteckt wird und kann dessen Verbindung dann automatisch starten. Er reagiert auf Statusänderungen eines Devices sowie auf Fehler, die in der Hardwarekomponente auftreten. Der NetworkManager kann diese Informationen über die

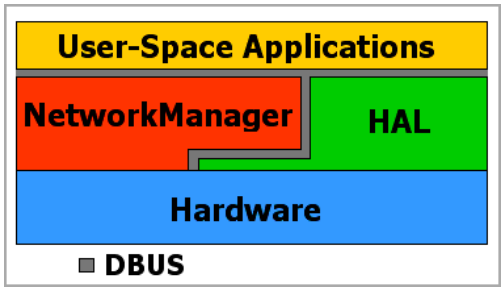


Abbildung 2.8: NetworkManager Architektur [Red09]

physikalischen Netzwerkschnittstellen dem Userinterface weiterleiten.

HAL HAL (*Hardware Abstraction Layer*) ist eine Zwischenschicht zwischen Hardware und Software, welche die Zugriffe der Software auf die Hardware abstrahiert. HAL wird neben dem NetworkManager gebraucht, da einige der Netzwerkkarten nicht von NetworkManager unterstützt sind. Diese Devices werden dann von HAL abstrahiert. Im Laufe der letzten Jahre wurden die Treiber der Hardware jedoch verbessert und standardisiert. Somit wird die HAL Komponente irgendwann überflüssig sein.

User-Space Applications Die User-Space Applications definieren die verschiedenen grafischen Oberflächen und Programme, die Netzwerkfunktionen benötigen. Dazu gehören das NM Plasma Applet sowie ein Webbrowser oder E-Mail Client.

2.4 D-Bus

D-Bus ist ein Open Source IPC[1] Framework, das sich insbesondere an den Bedürfnissen von Desktopumgebungen einer graphischen Benutzeroberfläche orientiert. Es ist Bestandteil des freedesktop.org Projekts und wird nahezu bei jeder modernen Linux-Distribution eingesetzt, die über eine graphische Oberfläche verfügt. In seiner Gesamtheit stellt der D-Bus mit all seinen Komponenten einen *Object Request Broker* (ORB) dar.

2.4.1 Architektur

Die D-Bus Spezifikation ist in drei Komponenten aufgeteilt: D-Bus-Daemon, D-Bus-Bibliothek (libdbus) und D-Bus-Protokoll.

D-Bus-Daemon Der D-Bus-Daemon implementiert einen sogenannten Nachrichtenbus, der in seiner Funktion einem *Object Request Broker* gleich kommt. Eine beliebige Anzahl von Clients verbindet sich mit dem Daemon basierend auf dem D-Bus-Protokoll und in der Regel unter Zuhilfenahme von libdbus. Einzelne RPC[2]-Aufrufe werden vom Daemon entgegen genommen und an den zuständigen Client vermittelt. Der Daemon ist somit ein Broker. Die RPC-Aufrufe reagieren auf abstrakte Objekte und stellen in Anlehnung an das OOP[3]-

[1] Inter-Process Communication
[2] Remote Procedure Call
[3] Object-Oriented Programming

Objektmodel folglich Methodenaufrufe dar. Schliesslich stellt der Daemon zeitgleich basierend auf den Einzelverbindungen ein *Broadcast* zur Verfügung.

D-Bus-Bibliothek Die D-Bus-Bibliothek stellt eine Referenzimplementierung des D-Bus-Protokolls bzw. der D-Bus-Spezifikation als Ganzes dar. Sie bietet eine API[4] in der Programmiersprache C, mit der zwei Prozesse eine Verbindung eingehen können, um danach via RPC zu kommunizieren. Sie stellt weiterhin das nötige Marshalling zur Verfügung, also das Ausrichten und korrekte Anordnen der Bytes für den Transport. Letztlich bietet die API alle Dienste für eine einfach abstrahierte Punkt-zu-Punkt Interprozesskommunikation.

Aufbauend auf libdbus wurden Implementierungen für alle verbreiteten Programmiersprachen und Frameworks entwickelt, bei letzterem insbesondere auf Qt und Glib (Gtk+) zugeschnittene Implementierungen.

D-Bus-Protokoll Das D-Bus-Protokoll ist ein binäres IPC-Protokoll. Spezifiziert sind sowohl Primitivtypen als auch zulässige Nachrichtentypen sowie die Kodierung für den Transport.

2.4.2 D-Bus Kommunikation

Wollen zwei Applikation über den D-Bus kommunizieren können sie einen sogenannten **Service Name** beziehen. Über diesen Service Name macht sich eine Applikation auf dem D-Bus bekannt. Der Service Name ist in der Regel in Form eines Domain-Names wie z.B. org.strongswan.charon.

Neben dem Service Name existiert immer ein **Objekt**. Dieses Objekt repräsentiert eine Verbindung und hat immer einen Pfad im Stile von "/org/strongswan/charon/Objekt1". Objekte werden erstellt, wenn sich ein Client am D-Bus anmeldet und bleibt solange bestehen bis sich der letzte Client wieder abmeldet. Nun können sich andere Clients an den D-Bus anschliessen und mit der Gegenstelle über diese Objekte kommunizieren.

Objekte besitzen **Interfaces**, die Methoden und Signale definieren. **Methoden** werden aufgerufen, wenn die Clients untereinander kommunizieren wollen. Methoden können beliebig viele Parameter haben, die über die Interfaces festgelegt werden. Parameter können von beliebiger Form sein, die D-Bus-Bibliothek konvertiert die Parameter in die gewünschten Typen. Ruft ein Client Methoden auf, die nicht existieren oder mit den falschen Parametern, wird eine Exception geworfen. Die Methoden werden normalerweise asynchron ausgeführt, was bedeutet, dass der Client nicht auf die Antwort des D-Busses warten muss und zeitgleich andere Arbeiten durchführen kann.

Signale sind die zweite Variante der Kommunikation, hier findet eine Einwegkommunikation statt. Signale werden von einem Objekt an alle angeschlossenen Clients gesendet, identisch wie bei einem *Broadcast*. Die Clients müssen sich beim Objekt für die gewünschten Signale anmelden und in ihren Interfaceimplementationen Mehtoden anbieten, welche die Signale behandeln. Hat sich ein Client angemeldet, erhält dieser alle Signale des Objektes. Signale können wie Methoden Parameter haben, jedoch haben diese keine Rückgabewerte, da sie *Broadcast* Meldungen sind und keine Antworten dem Objekt liefern.

```
[ Service Name -> Object (Path) -> Interface -> Methods & Signals ]
```

[4] Application Programming Interface

Die Parameter der Methoden und Signale werden in der D-Bus Implementation von QT automatisch in folgende Typen gewandelt:

Qt type	D-BUS equivalent type	Interface Definition
uchar	BYTE	y
bool	BOOLEAN	b
int	INT32	i
uint	UINT32	u
qlonglong	INT64	x
qulonglong	UINT64	t
double	DOUBLE	d
QString	STRING	s
QDBusVariant	VARIANT	v

Tabelle 2.1: D-Bus Parameter Typen

Neben den einfachen Typen können auch Arrays oder Maps den Signalen und Methoden als Parameter übergeben werden. Diese werden in QT in Listen oder Maps gespeichert. Folgende Arrays sind auf dem D-Bus möglich:

Interface Definition	Beschreibung
ay	Array of bytes
au	Array of uint
ai	Array of int
ax	Array of int64
at	Array of uint64
ad	Array of double
ab	Array of boolean
a{ss}	Map of strings, strings

Tabelle 2.2: D-Bus Parameter Arrys

Maps können auch verschachtelt werden, so kann z.B: als Interface Definition folgender Parameter definiert werden: **a**{**sa**{**sv**}} was einer **Map** mit dem *Key* **String** und als *Value* wieder eine **Map** hat. Diese zweite *Map* hat als *Key* einen **String** und als *Value* eine Variable vom Typ **Variant**. [fre10]

2.4.3 NetworkManager Schnittstellendefinition

Wie bereits erwähnt, läuft die ganze Kommunikation der verschiedenen Komponenten über die D-Bus Schnittstelle. Deshalb ist es wichtig, die Schnittstellendefinitionen des NetworkManagers zu analysieren und einzuhalten. In Appendix B ist die komplette VPN D-Bus Schnittstellendefinitionen für den NetworkManager aufgelistet.

Die Methoden und Signale der VPN Schnittstelle des NetworkManagers sind für die hier vorliegende Arbeit von grosser Bedeutung. Daher werden sie kurz erläutert. Es existieren zwei “Service Names” für die VPN Kommunikation. Der eine Service bildet die Schnittstelle zwischen dem

VPN Daemon und dem NetworkManager (VPN.Pugin). Der andere ist für die Kommunikation zwischen NetworkManager und NM Plasma Applet zuständig (VPN.Connection). Der Service **org.freedesktop.NetworkManager.VPN.Connection** repräsentiert eine aktive VPN Verbindung. Dieser Service besitzt keine Methoden, definiert jedoch zwei Signale, über die das Userinterface informiert werden kann, wenn die Verbindung ihren Status ändert.

Signal	**Parameter**
PropertiesChanged	a{sv}
VpnStateChanged	u, u

Tabelle 2.3: VPN.Connection Signale

Der Service **org.freedesktop.NetworkManager.VPN.Plugin** besitzt einige Methoden, die gebraucht werden um eine Verbindung aufzubauen und abzubrechen. Es können so auch Fehler des Plugins gemeldet und Passwortabfragen angefordert werden. Folgende Tabellen listen die Methoden und Signale des “VPN.Plugin” Services auf:

Methoden:

Methoden	**Parameter**	**Rückgabewert**
Connect	a{sa{sv}}	void
NeedSecrets	a{sa{sv}}	s
Disconnect	-	void
SetIp4Config	a{sv}	void
SetFailure	s	void

Tabelle 2.4: VPN.Plugin Objekte Methoden

Signale:

Signal	**Parameter**
StateChanged	u
Ip4Config	a{sv}
LoginBanner	s
Failure	u

Tabelle 2.5: VPN.Plugin Objekte Signale

2.4.4 D-Bus in QT

QT und KDE bieten Mechanismen an um Applikationen über D-Bus Services zu verknüpfen. Die Services und deren Schnittstellendefinitionen, die auf dem D-Bus definiert sind, können in Form von XML Files in den D-Bus Verzeichnissen abgerufen werden. Wenn man selbst einen Service schreibt, kann man mit dem Tool “qdbuscpp2xml” ein solches Schnittstellen-XML-File erzeugen. Dieses File muss im *Make* File beim Kompilieren von KDE angegeben werden.

Beispiel eines XML Schnittstellen Files:

```
<node>
  <interface name="org.foo.bar.test">
    <method name="currentBackground">
      <arg type="s" direction="out"/>
    </method>
    <signal name="halloBack">
    </signal>
  </interface>
</node>
```

Folgende Zeilen müssen beim Kompilieren von KDE mitgegeben werden:

```
set(test_xml ${DBUS_PREFIX}/interfaces/org.foo.bar.xml)
QT4_ADD_DBUS_INTERFACE(myapp_SRCS ${test_xml} test_interface )
```

Wenn das XML File in den Verzeichnissen von D-Bus gefunden und die Applikation erfolgreich kompiliert wurde, wird ein Objekt “Test” erzeugt. Über dieses Objekt kann man nun die D-Bus Methoden aufrufen und Signale abfangen. Nachfolgend ein Beispiel, das ein neues “Test” Objekt kreiert und die Methode “testMethode” mit den Parameter “hallo world” aufruft.

```
1. org::foo::bar::test *interface = new org::foo::bar::test("org.foo.bar",
2. "/network", QDBusConnection::sessionBus(), this);
3. interface->testMethode("hallo world");
```

2.5 strongSwan NM Plugin

StrongSwan ist eine auf IPsec/IKEv2 basierte VPN Lösung. Im Kernel Space des Betriebssystems ist die IPsec Implementation untergebracht. Sie verschlüsselt die IP Pakete durch das IP-Encapsulating-Security-Payload (ESP) Protokoll. Im User Space ist der Charon Daemon untergebracht, dieser implementiert den IKEv2 Standard.

Neben dem Schlüsselaustausch (Hauptfunktionalität des Charon Daemons), bietet Charon diverse Komponenten an. Eine dieser Komponenten ist der Bus. Der Bus erhält Singale der Threads, die für den Schlüsselaustausch kreiert werden und er kann so Fehler während des Schlüsselaustausches erkennen. Weiter werden Statusänderungen über den Bus kommuniziert. Die Meldungen auf dem Bus können in Charon über den *Listener* abgerufen werden. [HM05]

Über eine Plugin Infrastruktur können diverse Komponenten dem Charon Daemon beim Start hinzugeladen werden. Eines dieser Plugins ist das NM Plugin. Dieses ermöglicht die Kommunikation mit dem NetworkManager. Das Plugin wurde von Martin Willi ihm Rahmen der Integration von strongSwan in das Gnome Applet entwickelt. Über dieses Plugin können die Verbindungsparameter geladen werden, die nötig sind für eine *Remote Access* (Read Warrior) Verbindung. Das NM Plugin besitzt ein *Listener* Objekt, dadurch ist es möglich Änderungen und Fehler einer Verbindung im Plugin festzustellen.

Das *Listener* Objekt des NM Plugins kann sich für Methoden und Signale auf dem Bus anmelden und wird so z.B. über Fehler im strongSwan Daemon informiert. Jedoch werden nur die Meldungen des Clients über den Bus propagiert. Damit die *Notify Messages* des Servers analysiert

werden können, muss die Methode “message” im Listener des strongSwan NM Plugins implementiert sein. Danach lassen sich im NM Plugin die *Notify Messages* des Servers abfangen.

2.5.1 Verbindungsparameter

Damit eine *Remote Access* Verbindung aufgebaut werden kann, muss der NetworkManager dem strongSwan Plugin die dazu nötigen Verbindungsparameter übermitteln. In Tabelle 2.6 sind die nötigen Verbindungsparameter aufgelistet. Das NM Plugin prüft schon beim Erhalt dieser Verbindungsparameter, ob sie zulässig sind. Das heisst, es wird geprüft, ob die Gateway Adresse gültig ist und ob die Zertifikate gelesen werden können. Falls die Verbindungsdetails verifiziert werden können, baut das NM Plugin über den Charon Daemon eine Verbindung mit dem Gateway auf. Die Verbindungsparameter die übermittelt werden, sind in einer *Map* untergebracht.

Key	**Beschreibung**	**Value**
address	Gateway Adresse	String
certificate	Pfad zum Gateway Zertifikat	String
method	Methode der Client Authentifizierung	“psk”, “agent” oder “key”
user	Benutzername	String
password	Passwort	String
usercert	Pfad zum Benutzer Zertifikat	String
agent	Pfad zum SSH Zertifikat	String
userkey	Pfad zum User Key	String
virtual	Verbindungsaufbau mit einer virtuellen IP Adresse	“yes” oder “no”
encap	UPD Einkapselung	“yes’ oder “no”
ipcomp	IP Kompression	“yes” oder “no”

Tabelle 2.6: strongSwan NM Plugin Verbindungsparameter

2.6 Zusammenfassung

Die Herausforderung besteht darin, die vorhandenen Komponenten vollumfänglich zu verstehen. Damit eine strongSwan VPN Verbindung unter KDE Plasmoid aufgebaut werden kann, müssen die vorgestellten Komponenten: NM Plasma Apple, NetworkManager und strongSwan NM Plugin miteinander verknüpft werden. Die Abbildung 2.9 visualisiert die involvierten Komponenten und zeigt wie sie verbunden werden.

Der grösste Teil der Entwicklung in dieser Arbeit wird im NM Plasma Applet realisiert. Damit der Verbindungsaufbau funktioniert, muss die Komponente NM Plasma Applet erweitert werden. Damit Verbindungsfehler angezeigt werden können, müssen in den Komponenten des NetworkManagers und des strongSwan NM Plugin Änderungen vorgenommen werden. Wichtig ist dabei die Kommunikation über den D-Bus. Es ist notwendig sich an die Schnittstellendefinition zu halten.

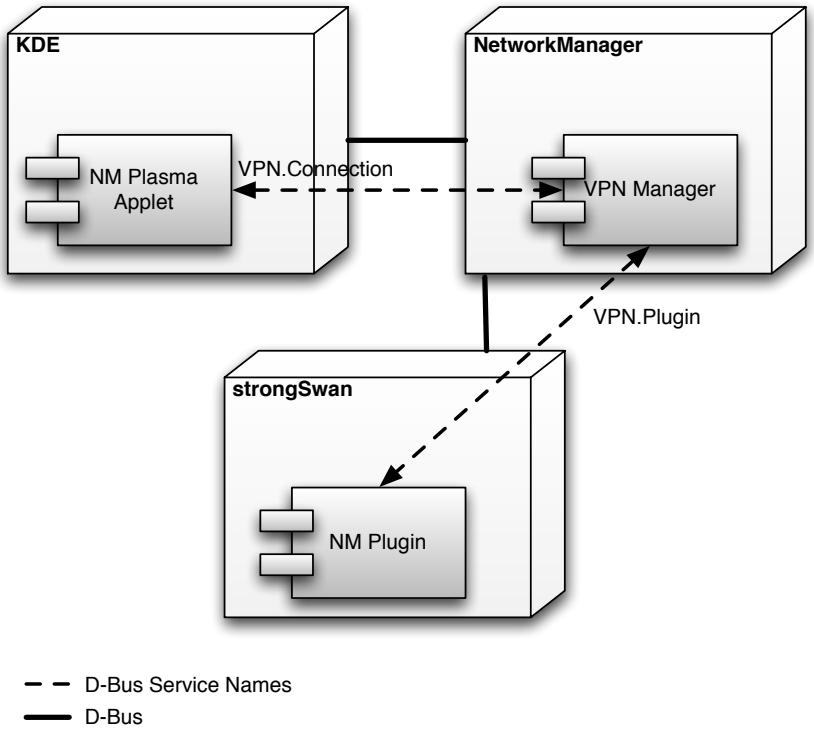


Abbildung 2.9: Involvierte Komponenten

3 Umsetzungskonzept

In diesem Kapitel werden die wichtigsten Entscheide, die basierend auf den erarbeiteten Grundlagen getroffen wurden, dokumentiert. Weiter werden die zu entwickelnden Funktionalitäten beschrieben und die damit verbundenen Änderungen und Erweiterungen in den existierenden Lösungen von KDE, strongSwan und dem NetworkManager. Die detaillierte Software Dokumentation befindet sich in Teil II dieser Arbeit.

3.1 Art des Plasmoid

Zuerst musste ich mich für ein Plasmoid entscheiden. Zwei Optionen schienen plausibel. Es gab entweder die Möglichkeit das strongSwan Plugin in die existierende NM Plasma Applet 2.2 Umgebung einzubinden oder ein eigenständiges Plasmoid für strongSwan zu entwickeln. Als Beispiel solcher grafischen VPN Clients diente der Cisco VPN Anyconnect (Abbildung 3.1) und KVpnc (Abbildung 3.2). Die zwei VPN Clients und das NM Plasma Applet habe ich analysiert und getestet.

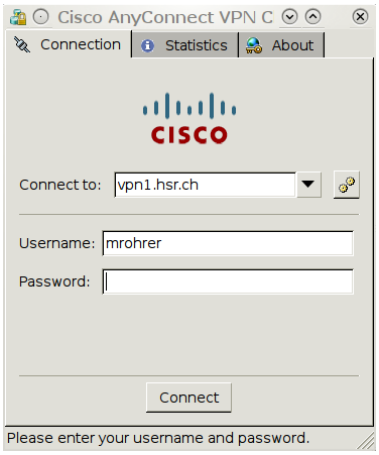


Abbildung 3.1: Cisco VPN Anyconnect

Cisco Anyconnect

Der Cisco Anyconnect hat in der Analyse gut abgeschnitten, da er kompakt und einfach zu konfigurieren ist. Zudem ist der Cisco Anyconnect sehr stabil und ist in den durchgeführten Tests nie abgestürzt. Mit dem Cisco Anyconnect lässt sich aber immer nur eine Verbindung konfigurieren. Ein weiterer Nachteil des VPN Anyconnects sind die schwer verständlichen Fehlermeldungen, die bei missglücktem Verbindungsaufbau oder unerwartetem Abbruch der Verbindung angezeigt werden. Aufgrund der zu komplexen Fehlermeldungen können Fehler nicht effektiv behoben werden. Da der Client von Cisco entwickelt wurde, lässt sich der Anyconnect auch nicht erweitern und verbessern.

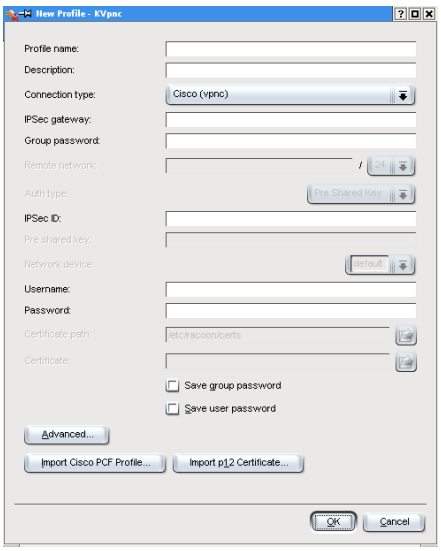


Abbildung 3.2: KVpnc

KVpnc

Der KVpnc ist ein Client für “Profis”, sprich er ist umfangreich bezüglich den Konfigurationsmöglichkeiten. Dieses Tool ist für den “einfachen” Benutzer nicht empfehlenswert, da es detaillierte Kenntnisse über die einzurichtende VPN Verbindung voraussetzt. In der Aufgabenstellung ist jedoch ein benutzerfreundliches strongSwan Plasmoid gefordert, deshalb ist ein Client im Stile des KVpnc nicht empfehlenswert. Zudem hat er durch seine Komplexität und sein unübersichtliches Userinterface einen vertrauensunwürdigen Eindruck hinterlassen.

NM Plasma Applet

Wie bereits erwähnt, ist das NM Plasma Applet noch im Stadium einer Beta Version. Diese bietet jedoch eine stabilere und benutzerfreundlichere Handhabung als ihre Vorgänger. Es ist noch nicht ausführlich getestet worden und dadurch auch nicht gegen Abstürze geschützt. Das bedeutet z.B. dass sich bei unerwartetem Abbruch der Netzwerkverbindung der NM Plasma Applet neu startet. Falls man während des Abbruchs mit einem VPN Gateway verbunden ist, kann keine genaue Fehleranalyse durchgeführt werden. Dafür ist das NM Plasma Applet sehr benutzerfreundlich. Benutzer schätzen insbesondere, dass es von KDE entwickelt wurde. Zudem ist die Konfiguration der Verbindung sehr einfach und es lassen sich im Vergleich zum Cisco Anyconnect beliebig viele Verbindungen konfigurieren.

3.1.1 Bewertung und Entscheid

Die drei analysierten Clients wurden während der Tests mit diversen Bewertungskriterien evaluiert. In der Tabelle 3.1 ist die Auswertung der Evaluation ersichtlich. Die Skala reicht von 1 = sehr gut, bis 6 = schlecht.

Bewertungskriterien	Cisco	KVpnc	NM Plasma
Benutzerfreundlichkeit	2	6	1
Installation	3	3	1

Stabilität	1	3	5
Konfigurationsmöglichkeiten	2	1	1
Komplexität	1	6	1
Fehlerbehandlung	5	5	6
Erweiterbarkeit	6	3	3
Vertrauen in Software	3	5	1
Handhabung	1	6	1
Total	**24**	**38**	**20**

Tabelle 3.1: VPN Client Bewertung

Das NM Plasma Applet hat in der Bewertung am besten abgeschlossen. Der Cisco Anyconnect hat gut abgeschlossen und der KVpnc ist abgeschlagen. Entscheidend beim NM Plasma Applet war die Installation, die genau genommen nicht notwendig ist, da das NM Plasma bereits in der Desktopumgebung eingebunden und somit automatisch installiert ist. Dies widerspiegelt sich auch im Vertrauen der Benutzer. Da das KDE Projekt *opensource* ist, lässt sich das NM Plasma Applet einfach erweitern und bietet die Möglichkeit Verbesserungen einzubringen, jedoch ist mit einer langen Einarbeitungszeit zu rechnen da die Architektur komplex ist.

Der grösste Vorteile eines individuellen strongSwan Plasmoid wäre die Unabhängigkeit gegenüber dem sich noch in der Beta Version befindenden NM Plasma Applet. Der Nachteil ist jedoch, dass die Benutzer extra Software installieren müssen und nicht mit der gewohnten Netzwerkumgebung arbeiten können. Das strongSwan Plugin unter Gnome hat seine Beliebtheit auch der Integration in die bestehende Umgebung zu verdanken. Fazit: auf kurze Sicht ist ein individuelles Plasmoid stabiler. Auf lange Sicht gesehen, ist die Integration in den NM Plasma Applet die bessere Variante, da das NM Plasma Applets jede Woche stabiler wird und schon bald optimal funktionieren wird.

In Absprache mit dem Betreuer dieses Projekts wurde die Entscheidung getroffen, das strongSwan Plugin in die NM Plasma Applet Umgebung einzubauen. Ebenfalls wurde entschieden, dass der Hauptteil der Entwicklung auf der Verbesserung und Erweiterung des NM Plasma Applets fokussiert wird.

3.2 strongSwan im NM Plasma Applet

Da entschieden wurde das strongSwan Plugin in das NM Plasma Applet einzubetten, muss die Architektur des NM Plasma Applet genauer analysiert werden. Im Abschnitt 2.2 sind die Grundfunktionalitäten des NM Plasma Applets beschrieben. Das NM Plasma Applet verfügt bereits über diverse VPN Plugins wie Vpnc, OpenVpn, NovellVpn. Das heisst, es existieren bereits Schnittstellen für VPN Verbindungen. Die VPN Verbindungen werden als Plugins in das NM Plasma Applet geladen. Das Plugin Framework von KDE erlaubt dem Entwickler, dynamisch Funktionalitäten einer Applikation zur Verfügung zu stellen. Damit ein Plugin vom Framework geladen werden kann, müssen einige Anpassungen im Plugin Code gemacht werden. Es ist nötig die Macros “K_PLUGIN_FACTORY” und “K_EXPORT_PLUGIN” im Code anzuwenden. Diese Macros machen das Plugin dem Framework bekannt, daneben muss neben diesen Macros ein “.desktop” File vorhanden sein. Dieses definiert seine Klassen und Services auf dem D-Bus.

Im dem entwickelten strongSwan Plugin wurden diese Macros hinzugefügt:

```
K_PLUGIN_FACTORY( StrongswanUiPluginFactory, registerPlugin<StrongswanUiPlugin>(); )
K_EXPORT_PLUGIN( StrongswanUiPluginFactory( "networkmanagement_strongswanui",
 "libknetworkmanager" ) )
```

Das Plugin “StrongswanUiPlugin” kann nun an beliebiger Stelle in dem NM Plasma Applet geladen werden. Dies geschieht anhand der Klasse “KServiceTypeTrader”. Folgendes Beispiel kann nun angewendet werden:

```
VpnUiPlugin *plugin = KServiceTypeTrader::createInstanceFromQuery<VpnUiPlugin>
 ("NetworkManagement/VpnUiPlugin",
  "[X-KDE-PluginInfo-Name]==networkmanagement_strongswanui" );
```

Folglich kann über das Objekt “plugin” auf die Funktionen und Daten des geschriebenen Plugins zugegriffen werden. Das Objekt “plugin” hat den Typ “VpnUiPlugin” und ist ein Interface des NM Plasma Applets, das erlaubt die VPN üblichen Funktionen zu implementieren.

Damit das Plugin vom “KServiceTrader” geladen werden kann, muss neben dem Sourcecode ein “*.desktop” File vorhanden sein. Dieses Desktop File beschreibt das Plugin und definiert seinen Service auf dem D-Bus. Folgende Definitionen müssen im “.desktop” File vorhanden sein:

```
[Desktop Entry]
 ServiceTypes=NetworkManagement/VpnUiPlugin
 X-KDE-Library=networkmanagement_strongswanui
 X-NetworkManager-Services=strongswan
 X-KDE-PluginInfo-Name=networkmanagement_strongswanui
```

Der Eintrag “ServiceTypes” definiert die Art des Plguins und dessen Interface. Das bedeutet, dass das zu entwickelnde strongSwan Plugin von dieser Klasse abgeleitet werden muss. Der Eintrag “X-KDE-PluginInfo-Name” definiert das strongSwan Plugin und es lässt sich somit von den Vpnc und NovellVpn Plugins unterscheiden. Der Eintrag “X-NetworkManager-Services” ist sehr wichtig und definiert den strongSwan Daemon, der mit diesem Plugin kommunizieren wird. Es spezifiziert ebenfalls den Service auf dem D-Bus des NetworkManagers.

Neben der Definition für die Plugin Infrastruktur muss ein strongSwan Widget (Fenster) erstellt werden. Dieses Fenster wird dann beim Einrichten und Konfigurieren der strongSwan Verbindungen angezeigt. Das strongSwan Fenster wird als *Tab* im Standard Fenster einer Verbindung geladen. Abbildung 3.3 zeigt das Verbindungsfenster mit den *Tabs*: strongSwan VPN und IP Adresse. Das Laden des strongSwan Fensters innerhalb des Verbindungsfensters wird über das Plugin Framework realisiert. Bei der Gestaltung des Userinterface des strongSwan Fensters gibt es kaum Freiheiten, da das Fenster in das existierende GUI passen muss. Mehr zur Entwicklung des Userinterfaces ist in Abschnitt 6.3 zu finden.

Wird eine strongSwan Verbindung angelegt, so wird diese anhand des KConfig Frameworks auf der Festplatte als Verbindung gespeichert. Die erstellte Verbindung ist nun für das Plasma Applet verfügbar und kann im Userinterface des NM Plasma Applets dargestellt werden. Zudem ist es nun möglich dank der Definition im “.desktop” File eine strongSwan Verbindung aufzubauen. Das NM Plasma Applet sendet die gespeicherten Informationen über *kded* dem NetworkManager.

Abbildung 3.3: NM Plasma Applet Verbindungsfenster

Dieser wiederum spricht über den D-Bus den strongSwan Daemon an und baut eine Verbindung auf.

Jede gespeicherte VPN Verbindung besitzt nun ein Verbindungsobjekt im NM Plasma Applet, das auch mit einem Verbindungs–Userinterface-Element verbunden ist, damit es auch über den NM Plasma Applet gestartet werden kann.

3.3 Erweiterungen im NM Plasma Applet

Durch das Erstellen des strongSwan Plugins im NM Plasma Applet, kann nun eine strongSwan Verbindung eingerichtet, gestartet und gestoppt werden. Die Hauptfunktionalität der Aufgabenstellung ist somit erfüllt. Die VPN Funktionalitäten des NM Plasma Applets sind jedoch noch nicht sehr ausgereift, es fehlt z.B: die Benachrichtigung nach einem erfolgreichen Verbindungsaufbau. Deshalb musste einiges am NM Plasma Applet erweitert und verbessert werden. In den folgenden Unterkapiteln sind die von mir realisierten Funktionalitäten im NM Plasma Applet beschrieben. Für die Arbeiten am NM Plasma Applet war die Absprache mit den Entwicklern von KDE sehr wichtig. Der Kontakt fand ausschliesslich per E-Mail statt. Der erstellte Code wurde von den KDE Entwicklern überprüft und dann in die Versionsverwaltung geladen.

3.3.1 Passwort Abfrage

In der analysierten NM Plasma Applet Version konnte man eine Verbindung nur starten, wenn das Passwort des Benutzers gespeichert wurde. Benutzer sollen jedoch die Möglichkeit haben, das Passwort bei jedem Verbindungsaufbau erneut anzugeben. Es gibt vereinzelt Benutzer, die das bewusst nicht wünschen und somit das Plugin nicht benutzten werden, wenn sie wissen, dass ihr Passwort gespeichert wird. In der analysierten Version konnte man den Modus für das Speichern der Passwörter nur global (Loginsession) festlegen, das bedeutet es werden alle vorhandenen Passwörter gespeichert oder abgefragt. Da es aber Anwendungsfälle gibt, bei denen ein Benutzer zwar das Passwort seines drahtlosen Netzwerkes gespeichert haben will, aber nicht jenes seiner VPN Verbindung, müsste die Art das Passwort zu speichern verbindungsspezifisch behandelt werden können. Aus diesem Grund muss das NM Plasma Applet um die Funktion zur Speicherung der Passwörter auf Verbindungsebene erweitert werden.

Einer *Connection* sind im NM Plasma Applet *Settings* zugewiesen, die auch über das KConfig Framework auf der Festplatte gespeichert werden. Die bestehenden *Settings* (wie z.B. “name”, “id”, “type”) müssen mit dem Parameter “secret-storage-mode” erweitert werden, damit auf Verbindungsebene der Passwort-Speicher-Modus ermittelt werden kann. Der Aufzählungstyp “secret-storage-mode” ist vom Type “uint” und bezieht sich auf folgenden Aufzählung:

ID	Value
0	ASK
1	SAVE
2	PLAIN

Tabelle 3.2: secretStorageMode Aufzählungstyp

Damit auf den Aufzählungstyp “secret-storage-mode” zugegriffen werden kann, muss der *Connection* die Zugriffsfunktionen hinzugefügt werden. Während des Verbindungsaufbaus wird ein “KJob” abgearbeitet, der verantwortlich für das Laden des Passworte ist. Dieser “KJob” prüft nun neu den “secret-storage-mode” der *Connection*. Ist der “secret-storage-mode” auf “ASK” gesetzt, muss ein Passwort Dialog angezeigt werden und das vom Benutzer eingegebene Passwort darf nicht in der KWallet gespeichert werden. Der Passwort Dialog muss im VPN Plugin erstellt werden. Beim Modus “SAVE” wird das Passwort in der KWallet gespeichert und im “KJob” auch von dort geladen. Der Modus “PLAIN” besagt, dass das Passwort unverschlüsselt im Konfigurationsfile der *Connection* zu finden ist. Wird der “secret-storage-mode” einer Verbindung nicht über die “Set”-Methode gesetzt, wird die globalen Speichereinstellungen übernommen. Um dies funktionstüchtig zu machen, wurden diverse Klassen im NM Plasma Applet angepasst (Abschnitt 6.1.2).

3.3.2 VPN Icon in der Taskleiste

Wenn der Benutzer eine Verbindung erfolgreich aufgebaut hatte, wurde in der analysierten Version keine Benachrichtigung angezeigt. Das Icon des Devices hat sich nicht geändert. Dies liegt daran, dass die Icons in der Taskleiste nur auf Signale und Statusänderungen des Devices reagieren. Das bedeutet, dass nur Statusänderungen der Netzwerkkarten registriert werden. Da eine VPN Verbindung kein Device sondern eine *Connection* ist, muss das NM Plasma Applet erweitert werden, damit auch auf Änderungen des Status einer *Connection* reagiert wird. Folglich werden alle vorhandenen *Connections*, auch solche die neu erstellt werden, überwacht. Sind diese vom

Typ “VPN”, wird das Plasma Applet über die Änderung des Status informiert und kann dadurch den Bereich des Icons neu zeichnen und entsprechend ein Schloss Icon (Standart in KDE für VPN) angezeigen.

Es wird eine Referenz der aktiven *Connection* gespeichert, damit man Informationen über diese Verbindung anzeigen kann. Dies geschieht in der “ToolTip”[1], die den Namen der aktiven Verbindung, deren Status und Dauer anzeigt (Abbildung 4.5).

3.4 Fehlermeldungskonzept

Während der Geschäftsmodellierung hat sich herausgestellt, dass die meisten Anwender wenig Kenntnisse zur VPN Technologie besitzen. Deshalb haben viele Benutzter Probleme, VPN spezifische Fehlermeldungen zu verstehen. Ziel ist es deshalb ein möglichst einfaches und klar verständliche Fehlermeldungskonzept umzusetzen. Das bedeutet, dass der Benutzter nur über Fehlermeldungen informiert wird, die auch von ihm verursacht werden, wie z.B. die Eingabe eines falschen Passworts.

Ein solches Fehlermeldungskonzept existiert weder im NM Plasma Applet noch im strongSwan NetworkManager Plugin. Jedoch bietet die NetworkManager D-Bus Schnittstellendefinition eine Methode und ein D-Bus Signal für das Melden von Fehlern des VPN Plugins (Appendix B) an.

Wie bereits erwähnt, sollten nur Fehlermeldungen gemeldet werden, die der Benutzter beheben kann. Fehler, welche vom Gateway verursacht werden, sind dem Benutzer nicht oder in allgemeiner Form wie z.B: “Kontaktieren Sie Ihren Administrator” zu melden. Dadurch wird vermieden, dass der Benutzer mit komplexen und für ihn nicht verständlichen Fehlermeldungen konfrontiert wird. Der Benutzer hat denn auch nur eine beschränkte Möglichkeit Fehler auf dem Gateway zu beheben.

Es ist zudem zu erwähnen, dass VPN Verbindungen sicherheitskritische Anwendungen sind und somit auch mit den Fehlermeldungen vorsichtig umgegangen werden muss. Durch eine zu genaue Fehlermeldung kann eine Verbindung systematisch angegriffen werden. Deswegen ist in der Spezifikation von IKEv2 nur eine beschränkte Kategorie von Rückmeldungen und Fehlermeldungen definiert worden. Dies ist jedoch für die Implementation der Fehlermeldungen ein grosses Problem, da der Benutzer so nicht genau feststellen kann, wo das Problem aufgetaucht ist, respektive worauf der Fehler gründet.

3.4.1 Fehlermeldungen

In Tabelle 3.3 sind die Fehlermeldungen aufgelistet, die dem Benutzter gemeldet werden sollten. In Tabelle 3.4 sind mögliche Fehlerursachen beschrieben.

Fehlermeldung	**Beschreibung**
Login Failed	Benutzer Authentifizierung fehlgeschlagen
Gateway Authentication Failed	Gateway Authentifizierung fehlgeschlagen
Authentication Failed	Gateway oder User Authentifizierung fehlgeschlagen

[1] Ein Tooltip ist ein kleines Pop-up-Fenster in einem Anwendungsprogramm, das dem Benutzer einen Beschreibungstext zu einem Element der grafischen Benutzungsoberfläche anzeigt.

Gateway Adresse Failure	Gateway Adresse ungültig
Connection Timeout	Gateway nicht erreichbar
Loading Gateway Certificate Failed	Gateway kann Zertifikat nicht lesen
Virtual IP Required	Gateway kann dem Client keine IP zuweisen
IP Pool Error	Gateway kann dem Client keine IP zuweisen
Init Failed	strongSwan Plugin konnte nicht initialisiert werden
Reason Unkown	unbekannter Fehler

Tabelle 3.3: Fehlermeldung Auflistung

Fehlermeldung	**Ursache**
Login Failed	Username oder Passwort sind ungültig
Gateway Authentication Failed	Benutzer hat falsches Server Zertifikat angegeben “Man in the Middle Attack”, falscher Gateway antwortet
Authentication Failed	Gateway Zertifikat ungültig Username existiert nicht auf dem Gateway
Gateway Adresse Failure	Benutzer hat keinen Gateway konfiguriert Gateway Domainname kann nicht aufgelöst werden Gateway IP ist ungültig
Connection Timeout	Gateway ist nicht erreichbar Gateway ist kein VPN Server Benutzer ist hinter einer Firewall
Loading Gateway Certificate Failed	Gateway Zertifikat existiert nicht Gateway Zertifikate können nicht geladen werden Gateway Zertifikate haben falsches Format
Virtual IP Required	Client IP Adresse Konflikt - Benutzer nutzt z.B. NAT
IP Pool Error	Server kann Client keine virtuelle IP zuweisen
Init Failed	strongSwan NM Plugin nicht installiert
Reason Unkown	andere möglichen Fehlerursachen

Tabelle 3.4: Gründe der Fehlermeldung

Die Fehlermeldung “Authentication Failed” scheint auf den ersten Blick überflüssig, da die Fehlermeldungen “Gateway Authentication Failed” und “Login Failed” diese beiden Fehlerzustände behandeln. Es ist jedoch notwendig, neben den zwei spezifischen Fehlermeldungen eine allgemeine Authentifizierung Fehlermeldung zu definieren. Diese Fehlermeldung trifft immer dann zu, wenn auf Seite des Gateways die Authentifizierung fehlschlägt. Die Spezifikation von IKEv2 [Kau05] definiert eine *Notify Payload*. Diese wird dem Client übermittelt, falls auf Seiten des Gateways irgendwas fehlgeschlagen hat. Die *Notify* Error Messages sind in Appendix C nachzulesen. Es ist eine Error Message “AUTHENTICATION_FAILED” definiert. Leider ist aus dieser Meldung nicht ersichtlich, ob die Authentifizierung des Gateways oder die des Users fehlgeschlagen hat. Die Generalisierung der Fehlermeldung wurde bewusst so gewählt, da sonst Angreifer das System systematisch attackieren könnten. Meist wird diese *Notify* Meldung dem Client übermittelt, falls keine Konfiguration für den gewünschten Gateway oder Username gefunden wurde. Existiert jedoch eine gültige Konfiguration, werden die Fehlermeldungen in der spezifischen Authentifizierung Payload übermittelt. So wird zum Beispiel bei einer Authentifizierung mittels EAP die Eingabe eines falsches Passwortes in der EAP Payload übermittelt. Falls diese fehlschlägt, kann

daraus sicher geschlossen werden, dass die User Authentifizierung fehlgeschlagen hat.

3.4.2 Fehlererkennung

In Abbildung 3.4 ist ein IKEv2 Verbindungsaufbau dargestellt. Anhand des Verbindugsaufbaus wird verdeutlicht, wo und wann ein Fehler erkannt wird. Die rot gekennzeichneten Fehler werden immer an das NM Plasma Applet (Userinterface) weitergeleitet. Es sollten keine Fehler im NM Plasma Applets selbst erkannt werde, denn wenn die Fehler auf der NetworkManager und strongSwan NM Plugin Seite erkannt werden, können diese auch an andere Userinterface Implementationen wie z.B. das Gnome Applet, delegiert werden.

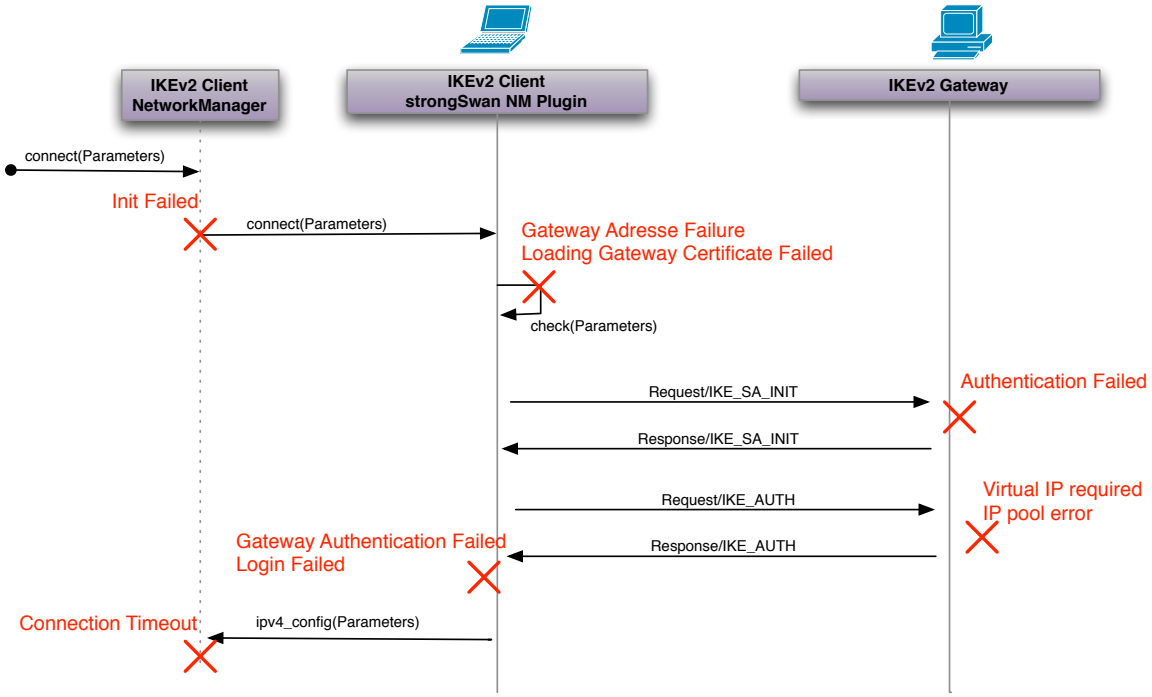


Abbildung 3.4: Fehlererkennung

3.4.3 Umsetzung

Damit die Fehlermeldungen, die vom strongSwan Plugin ausgelöst werden, auf der grafischen Benutzeroberfläche dargestellt werden können, müssen diverse Komponenten angesprochen werden. Die Kommunikation läuft über den D-Bus. Abbildung 3.5 zeigt die involvierten Komponenten und die dazugehörigen Signale und Methoden, die auf dem D-Bus ausgeführt werden. Das strongSwan Plugin muss die Methode “setFailure(s:reason)” auf dem D-Bus Objekt VPN.Plugin ausführen. Das Objekt löst nun ein Signal “failure(u:reason)” aus. Dieses wird vom NetworkManager entgegen genommen. Der NetworkManager löst nun ein “VpnStatChanged(u:state,u:reason)” aus, welches vom NM Plasma Applet abgefangen werden muss.

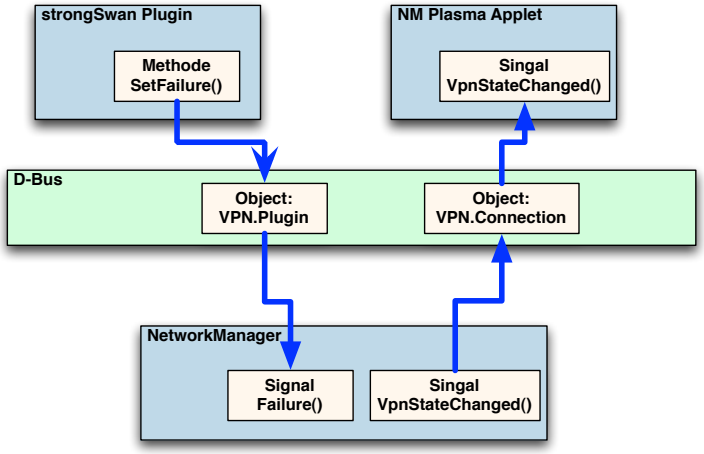


Abbildung 3.5: Fehlermeldungen auf dem D-Bus

Änderungen im NetworkManager

In der D-Bus Schnittstellendefinition des NetworkManagers existiert eine Methode “SetFailure(s:reason)”. Zudem besteht ein Signal “Failure (u:reason)”, das vom NetworkManager abgefangen werden kann. Der Aufzähltyp “u:reason” definiert folgende Zustände:

`NM_VPN_PLUGIN_FAILURE_LOGIN_FAILED`

`NM_VPN_PLUGIN_FAILURE_CONNECT_FAILED`

`NM_VPN_PLUGIN_FAILURE_BAD_IP_CONFIG`

Für diese Anwendung müssten dem NetworkManager fünf neue Fehlerzustände hinzugefügt werden. Es handelt sich dabei um nachfolgende Zustände:

`NM_VPN_PLUGIN_FAILURE_GATEWAY_AUTH_FAILED`

`NM_VPN_PLUGIN_FAILURE_GATEWAY_CER_FAILED`

`NM_VPN_PLUGIN_FAILURE_AUTH_FAILED`

`NM_VPN_PLUGIN_FAILURE_TS_UNECEPTABLE`

`NM_VPN_PLUGIN_FAILURE_INTERNAL_ADDRESS_FAILURE`

Der NetworkManager reagiert auf die Signale des D-Bus Objekts *VPN.Plugin*. Erhält der NetworkManager das Signale “Failure (u:reason)”, löst dieser intern wieder ein Signal auf dem Objekt *VPN.Connection* “VpnStateChanged (u: state, u: reason)” aus. Der NM Plasma Applet sollte nun auf die Signale des Objekts *VPN.Connection* reagieren. Die Aufzählungstypen “u:reason” des Signals “Failure” und “VpnStateChanged” sind nicht identisch, deswegen muss ein Mapping zwischen den zwei Aufzählungstypen durchgeführt werden.

Folgende “Reasons” sind für das Signal “VpnStateChanged” vom NetworkManager bereits konfiguriert:

`NM_VPN_CONNECTION_STATE_REASON_UNKNOWN`

`NM_VPN_CONNECTION_STATE_REASON_NONE`

NM_VPN_CONNECTION_STATE_REASON_USER_DISCONNECTED

NM_VPN_CONNECTION_STATE_REASON_DEVICE_DISCONNECTED

NM_VPN_CONNECTION_STATE_REASON_SERVICE_STOPPED

NM_VPN_CONNECTION_STATE_REASON_IP_CONFIG_INVALID

NM_VPN_CONNECTION_STATE_REASON_CONNECT_TIMEOUT

NM_VPN_CONNECTION_STATE_REASON_SERVICE_START_TIMEOUT

NM_VPN_CONNECTION_STATE_REASON_SERVICE_START_FAILED

NM_VPN_CONNECTION_STATE_REASON_NO_SECRETS

Auch hier müssen einige Zustände hinzugefügt werden:

NM_VPN_CONNECTION_STATE_REASON_LOGIN_FAILED

NM_VPN_CONNECTION_STATE_REASON_GATEWAY_AUTH_FAILED

NM_VPN_CONNECTION_STATE_REASON_CONNECT_FAILED

NM_VPN_CONNECTION_STATE_REASON_GATEWAY_CER_FAILED

NM_VPN_CONNECTION_STATE_REASON_AUTH_FAILED

NM_VPN_CONNECTION_STATE_REASON_TS_UNECEPTABLE

NM_VPN_CONNECTION_STATE_REASON_INTERNAL_ADDRESS_FAILURE

Einige der Fehlermeldungen werden direkt im NetworkManager erkannt. So wird zum Beispiel festgestellt, ob das strongSwan NM Plugin installiert ist oder nicht. Falls dies nicht der Fall sein sollte, wird direkt aus dem NetworkManager die Fehlermeldung “Init Failed” delegiert. Weiter wird vom NetworkManager die Fehlermeldung “Connection Timeout” erkannt.

Änderungen im NM Plasma Applet

Im NM Plasma Applet muss auf das Signal “VpnStateChanged()” des D-Bus Objekts *Vpn.Connection* reagiert werden. Die Implementation dieses Signals muss dem existierenden Interfaceproxy hinzugefügt werden. Falls ein Signal empfangen wird, so wird ein KJob ausgeführt. Dieser stellt dem Benutzer die Fehlermeldung und das Konfigurationsfenster der Verbindung dar. So hat der Benutzer die Möglichkeit, Änderungen an seiner Verbindung vorzunehmen und erneut einen Verbindungsaufbau auszuführen. Das Fehlermeldungsfenster setzt sich aus dem Verbindungswidget und einer Messagebox zusammen. Mehr zur Gestaltung des Fehlermeldungsfensters ist in Kapitel 6.3 nachzulesen.

Änderungen im strongSwan NM Plugin

Da die Fehlerzustände verschiedene Ursachen und Fehlerquellen besitzen, bedingt dies, dass im strongSwan NM Plugin verschiedene Komponenten überwacht werden.

Im strongSwan NM Plugin müssen die *Notify Messages* vom Gateway abgefangen werden. Die Umsetzung verlangt, dass im NM Plugin die “message_hook” Methode des *Listeners* implementiert ist. Diese wiederum analysiert alle *Notify Messages* vom Gateway und leitet diese dem Net-

workManager weiter. Über die *Notify Messages* können folgende Fehlermeldungen abgefangen werden: Authentication Failed, Virtual IP required, IP Pool error.

Der Charon Daemon besitzt eine Alert Methode auf dem strongSwan Bus. Die Alert Methode vom Charon Bus muss dazu mit folgenden drei Zuständen erweitert werden:

`ALERT_GW_AUTH_FAILED`

`ALERT_GW_ADDR_FAILED`

`ALERT_EAP_AUTH_FAILED`

Der Alert `ALERT_GW_AUTH_FAILED` wird auf der Client Seite ausgeführt, wenn das vom Server erhaltene Zertifikate nicht mit dem angegeben Zertifikat übereinstimmt. Der Alert `ALERT_GW_ADDR_FAILED` wird beim Auflösen des Domainnamens des angegebenen Gateways ausgeführt. Der Alert `ALERT_EAP_AUTH_FAILED` wird ausgelöst, falls die Benutzerauthentifizierung über EAP fehlgeschlagen ist.

Über die Alerts werden schliesslich folgende Fehlermeldungen abgefangen: Login Failed, Gateway Authentication Failed, Gateway Adresse Failure

4 Resultate

Das NM Plasma Applet wurde erfolgreich für den Gebrauch des strongSwan Daemons erweitert. Es ist nun möglich über die gewohnte Netzwerkumgebung im KDE eine strongSwan Verbindung zu konfigurieren (EAP und Private Key/Zertifikat). Neben dem Einrichten der Verbindung, dem Starten und Stoppen ist das NM Plasma Applet in Absprache mit den KDE Entwicklern erweitert worden. Neu im NM Plasma Applet ist die Abfrage des Passworts bei einer VPN Verbindung mit Passwort. Dazu wurde ein extra Fenster entwickelt, das dem Benutzter erlaubt, sein Passwort und Username bei jedem Verbindungsaufbau neu anzugeben. Zudem wurde das Plasma Applet erweitert, so dass es auf Statusänderungen der VPN Verbindungen reagiert und somit den Benutzer über den jeweiligen Status der VPN Verbindung informiert. Daneben wurden diverse kleine Bugs behoben. Ganz neu ist das Fehlermeldungskonzept. Das NM Plasma Applet hat bis jetzt noch keine Möglichkeit geboten, Fehlermeldungen die von dem VPN Daemon kommen zu signalisieren. Es wurde ein Konzept geschaffen, das detaillierte und klar verständliche Fehlermeldungen zurückzugibt. Das Gnome Applet profitiert ebenfalls von diesen Erweiterungen und kann mit den aufgearbeiteten Signalen erweitert werden, damit auch in Zukunft eine genau Fehleranzeige des Gnome Applets möglich ist.

4.1 Screenshots

Nachfolgend werden anhand einiger Screenshots die Lösungen präsentiert. In Abbildung 4.1 ist das Fenster zu sehen, das mit EAP konfiguriert werden kann und auch die Möglichkeit bietet, ein Client Authentifizierung mittels Private Key und Certificate einzurichten.

Wenn der Benutzer das Passwort nicht speichern will, kann er die Auswahl (Abbildung 4.2) anklicken und wird dann bei einem Verbindungsaufbau aufgefordert sein Passwort einzugeben (Abbildung 4.3).

Hat der Benutzer eine Verbindung aufgebaut, ändert sich das Icon in der Taskleiste (Abbildung 4.4). Fährt der Benutzer mit der Maus über das Icon, werden ihm die Details zur aktuellen Verbindung angezeigt. Zudem ist ersichtlich wie lange die Verbindung schon aktiv ist (Abbildung 4.5).

Hat ein Verbindungsaufbau fehlgeschlagen, wird dem Benutzer das Konfigurationsfenster in einer Fehlermeldung angezeigt. Der Abschnitt, in dem der Fehler aufgetreten ist, wird mit einem Rahmen visualisiert. Abbildung 4.6 zeigt zwei Beispiele einer möglichen Fehlermeldung.

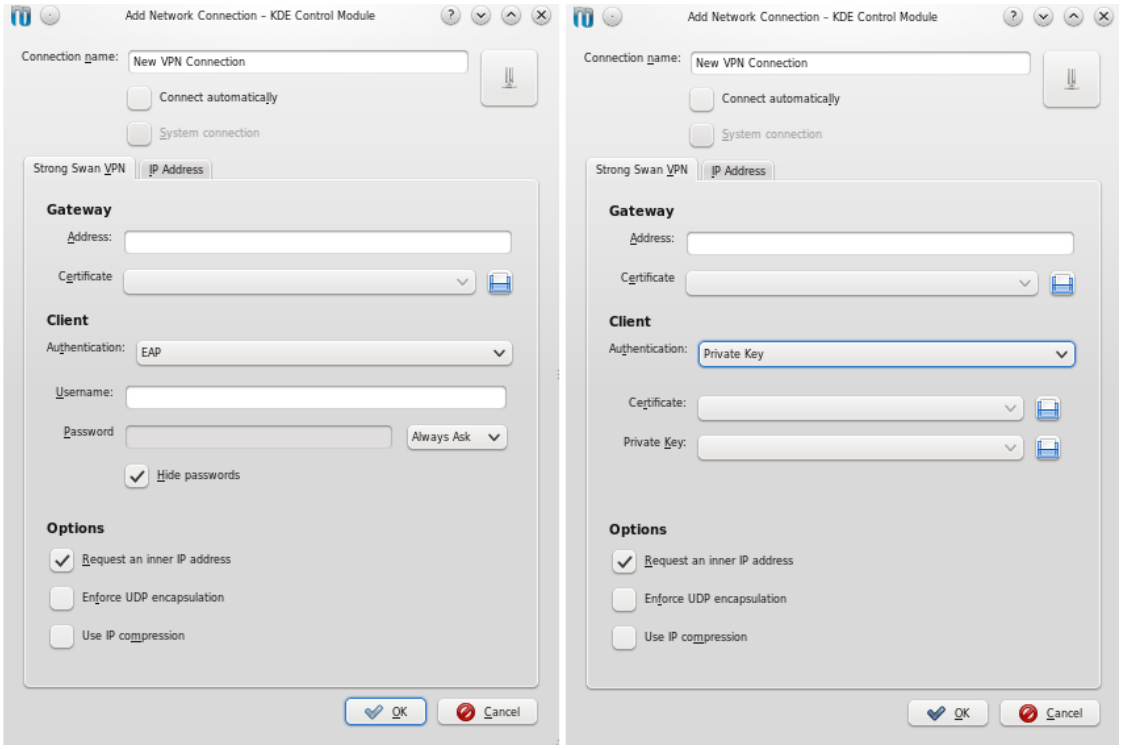


Abbildung 4.1: Resultate: stronSwan im NM Plasma Applet

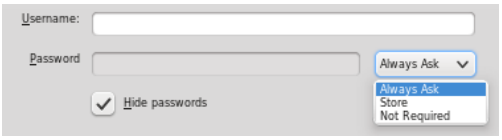


Abbildung 4.2: Resultate: Passwort Speicherart

4.2 Schlussfolgerungen

Nach langer Einarbeitungszeit ist es mir gelungen, mich gut in die Architektur und Entwicklungsumgebung von KDE und im Speziellen die des NM Plasma Applets einzuarbeiten. Die Komplexität der existierenden Frameworks wurden von mir adäquat eingeschätzt und in kurzer Zeit habe ich die gewünschten Punkte umgesetzt. Es ist nun möglich eine strongSwan VPN Verbindung über das neue NM Plasma Applet einzurichten und zu verwalten. Zudem konnte ich aktiv in der Entwicklung des NM Plasma Applets mithelfen. Ich möchte nochmals darauf hinweisen, dass das NM Plasma Applet bei Abgabe der Arbeit noch nicht in der Standard Desktop Umgebung integriert sein wird, da es sich noch in der “Review” Phase befindet. Anfangs Herbst wird das NM Plasma Applet voraussichtlich als Netzwerkuserinterface von KDE ausgeliefert werden. In diesem werden auch meine Weiterentwicklungen integriert sein. Der Entscheid das strongSwan Plugin in die NM Plasma Applet Umgebung einzubauen, ist aus meiner Sicht die richtige Entscheidung.

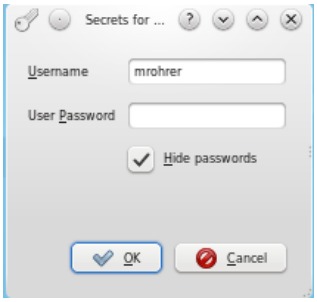


Abbildung 4.3: Resultate: Passwort Fenster

Abbildung 4.4: Resultate: VPN Icon in der Taskleiste

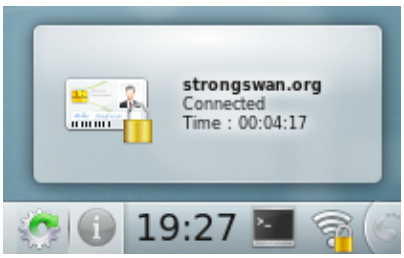


Abbildung 4.5: Resultate: ToolTip in der Taskleiste

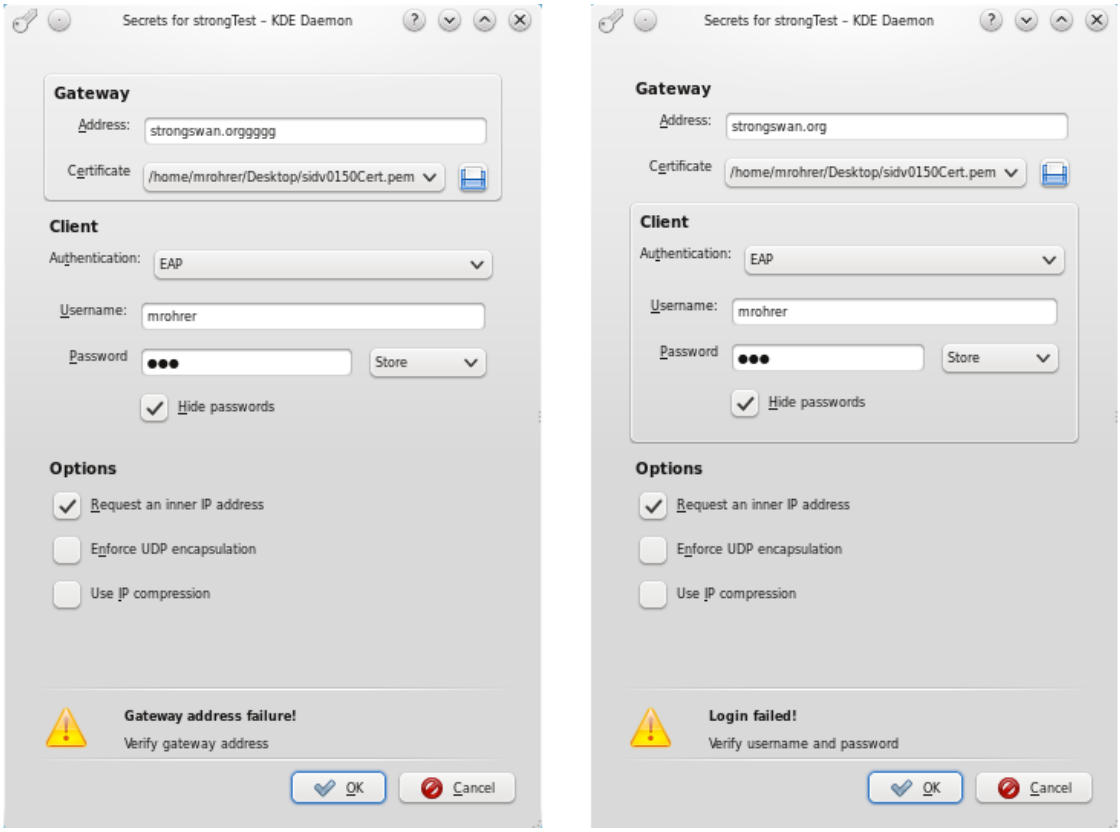


Abbildung 4.6: Resultate: Fehlermeldungsfenster

Teil II

SW-Projektdokumentation

5 Anforderungsanalyse

Um zu definieren, welchen Anforderungen “das System” genügen muss, wurde ein ausführliches Benutzerprofil erstellt und analysiert. Dazu wurde ein Interview mit einem potenziellen Benutzer (Persona) durchgeführt um dessen Bedürfnisse und Wünsche zu evaluieren.

5.1 Hypothetische Personas und Szenarios

Zuerst wurden hypothetische Personas erstellt. Diese Personas sind noch nicht personalisiert, widerspiegeln aber das erwartete Benutzerprofil.

Persona: Linux Anwender

Rolle	erfahrener Linux Anwender, der täglich mit dem System arbeitet
Ziele	Arbeit möglichst schnell erledigen, unabhängig von Ort und Zeit
Kenntnisse	gute Linux Benutzerkenntnisse, jedoch kein technisches Wissen bezüglich VPN und Verschlüsselung
Arbeitsstil	zielorientiert effizient qualitätsorientiert
Eigenschaften	erfolgreich kommunikativ aufnahmefähig ungeduldig
“Pain Points”	technische Fehler benutzerunfreundliche Oberflächen

Tabelle 5.1: Hypothetische Persona: Linux Anwender

Persona: Netzwerk Administrator

Rolle	Netzwerk Administrator, der die Netzwerkinfrastruktur verwaltet
Ziele	durch gute Tools die Effizientz und Funktionsfähigkeit seiner Systeme testen

Kenntnisse	fundiertes Wissen von Netzwerkinfrastrukturen und VPN Systemen
Arbeitsstil	sorgfältig sicherheitsorientiert
Eigenschaften	intelligent kleinlich
“Pain Points”	Sicherheitslücken im System schlechte Tools, die ihn bei seiner Arbeit hindern

Tabelle 5.2: Hypothetische Persona: Netzwerk Administrator

Szenario: Dokumente von Zuhause aus lesen

Die Persona will ihre Dokumente aus dem Firmen/Universität Netzwerk Zuhause lesen. Da der File Server der Firma/Universität nicht übers Internet erreichbar ist, muss sie sich über VPN ins Netzwerk verbinden.

Szenario: E-Mail lesen am Bahnhof

Die Persona hat einen längeren Aufenthalt an einem SBB-Bahnhof mit Hotspot. Sie will während der Wartezeit ihre E-Mail lesen. Damit die Persona ins Internet kommt muss sie sich über VPN mit dem Internet verbinden.

5.2 Interview

Um zu entscheiden welche der hypotetischen Personas zu interviewen ist, wurde eine Interviewmatrix erstellt.

Interviewmatrix

	Netzwerk Administrator	Linux Anwender
VPN Verbindung einrichten	X	X
VPN Verbindung aufbauen	X	X
Verbindungsaufbau Status	X	X
Fehleranalyse	X	
Layout ändern		X

Tabelle 5.3: Interviewmatrix

Da beide Personas die Hauptfunktionalitäten der Anwendung (Aufbau, Einrichten, Status der Verbindung) benutzen werden, wurde die rot markierte Persona für das Interview gewählt. Mehr-

heitlich werden Linux Anwender das strongSwan Plugin benutzen. Sie repräsentiert darum die Hauptanwendergruppe.

User Profil der interviewten Person

Name	anonym
Firma	Universität Zürich - E-Learning OLAT
Arbeitsort	Los Angeles Zuhause - Balkon, Arbeitszimmer
Arbeitsumfang	teilzeit, 50 %
Arbeitsumgebung	Laptop - Kubuntu 9.10, Max OS X Software - OpenOffice, OLAT, Tex, usw.
Tätigkeiten	Koordination - Öffentlichkeitsarbeit

Tabelle 5.4: Interview User Profil

Interview Details

Hauptbefrager	Maurus Rohrer
Protokoll	Maurus Rohrer
Vertraulichkeit garantiert	Ja
Audioaufnahme	Ja
Videoaufnahme	Ja
Artefakte	Anleitung Text File
Dauer des Interviews	25 Minuten
Interviewort	Zürich - Los Angeles
Konkrete Fälle	- VPN Verbindung einrichten - VPN Verbindung aufbauen - Status Analyse bei Fehlern
Letztes Auftreten der Fälle	letzten drei Tage
Fälle vollständig	Ja
Interpretation Session	Maurus Rohrer

Tabelle 5.5: Interview Details

Erkenntnisse aus dem Interview

VPN Verbindung integriert im Betriebssystem Benutzer hat grösseres Vertrauen in integrierten VPN Client, als in proprietäre Software. Schätzt die Stabilität des integrierten Clients.

Status Informationen - Informationsleiste Informationen über die Verbindung in Informationsleiste darstellen.

Status Informationen - Dauer der aktiven Verbindung darstellen Wenn Verbindung erfolgreich aufgebaut wurde, soll die Dauer der Verbindung angezeigt werden. Dies gibt dem Benutzer die Möglichkeit den Verbindungsstatus abzuwägen und die Dauer der Arbeitszeit zu erfassen.

Password speichern Password für die VPN Verbindung soll gespeichert werden können, da das Vertrauen ins Betriebssystem und dessen Schlüsselverwaltung gewährleistet ist.

VPN Einrichten einfach halten Nur nötige Informationen für die Einrichtung der Verbindung abfragen. Benutzer ist zufrieden, wenn Verbindung funktioniert, kein Interesse an detaillierten Konfigurationsmöglichkeiten.

5.3 Persona “Kommunikation”

Nachfolgend soll eine Persona genauer analysiert werden. Sie widerspiegelt die erwartete Hauptanwendergruppe.

Konrad Kommunikativ

- 36 Jahre
- Naturwissenschaftler
- Vater eines 12 jährigen Sohns
- reist viel, beruflich und privat

Konrad Kommunikativ arbeitet für die Universität Zürich im Bereich Informatik. Er ist zuständig für die Kommunikationsarbeit der OLAT E-Learning Software und ist viel unterwegs an Konferenzen und Ausstellungen. Zurzeit lebt Konrad in Los Angeles und arbeitet viel “remote” im Universitätsnetzwerk.

Konrad hat ein ETH Stdudium in Umweltnaturwissenschaften absolviert, arbeitete jedoch nie länger in diesem Bereich. Bereits während seiner Schulzeit sammelte er Erfahrungen mit Computern und dem Internet. Dies hat ihn zu einem erfahrenen Computeranwender gemacht.

Seine zielstrebige, kommunikative und flexible Persönlichkeit hat ihn weit gebracht. Konrad arbeitet viel unterwegs, in Kaffees oder auf seinem Balkon. “Ich arbeite immer, wenn ich Lust habe und eine Internetverbindung vorhanden ist” (lacht).

Nichts bringt ihn aus der Fassung. Funktionieren Computer oder Netzwerk nicht, nimmt er es gelassen und versucht es nochmals. “Wenn’s vorhin geklappt hat, jetzt aber nicht mehr, probiere ich es in 10 Minuten nochmals.” Was er nicht leiden kann ist Unzuverlässigkeit.

5.4 Szenarios

Arbeitsliste prüfen

Da Konrad in Los Angeles lebt, ist in der Schweiz der Arbeitstag schon vorbei wenn er zu arbeiten beginnt. Will sich Konrad über das Tagesgeschehen an der Universität Zürich informieren, kann er in einem Dokument wichtige Informationen, die allen Mitarbeitern des Teams zugänglich sind, nachlesen. Um dieses Dokument zu lesen, muss er sich in das Universitätsnetzwerk einloggen. Dazu startet Konrad seinen eingerichteten VPN Client und drückt auf den “connect” Button. Beginnt die Zeit des Icons zu zählen weiss er, dass er verbunden ist. Danach kann er sich mit dem Server verbinden und das Dokument lesen. Gibt es Aufgaben die ihn betreffen, kann er diese bearbeiten.

Skype Telefonate während der Arbeitszeit

Konrad ist meist in Skype eingeloggt und telefoniert darüber häufig beruflich und privat in die Schweiz. Wenn Konrad für die Universität arbeitet ist er immer über VPN eingeloggt. Will er jedoch “skypen” ist es besser, wenn er nicht über VPN verbunden ist, da dies die Verbindung verlangsamt. Dazu muss Konrad die Verbindung unterbrechen. Er klickt beim VPN Symbol in der Taskleiste auf “disconnect”. Wenn die Zeit stoppt weiss er, dass er nicht mehr über VPN verbunden ist und kann nun einen Skype-Telefonat tätigen.

5.5 Funktionale Anforderungen

Anforderungen mit “V1” (Version 1) wurden zu Beginn des Projekts definiert. Anforderungen höherer Versionen wurden zu einem später Zeitpunkt festgelegt. Die Anforderungen der Version “V1” beinhalten die in der Aufgabenstellung festgelegten Ziele. Die Anforderungen höherer Versionen beinhalten die optionalen Funktionalitäten der Aufgabenstellung.

VPN Verbindung einrichten - V1 Das zu entwickelnde Plugin muss dem Benutzer die Möglichkeit bieten, über ein benutzerfreundliches UI die nötigen Informationen für den Aufbau einer VPN Verbindung einzutragen. Die eingetragenen Verbindungsparameter müssen in Konfigurationsfiles auf dem Filesystem abgelegt werden. Zudem muss der Benutzer die Möglichkeit haben, die Verbindungsparameter zu ändern und zu löschen.

VPN Verbindung starten - V1 Der Benutzer muss die gespeicherte Verbindung starten können. Startet er diese, muss ebenfalls der strongSwan Daemon mit den gespeicherten Verbindungsparametern gestartet werden.

VPN Verbindung stoppen - V1 Der Benutzer muss die aktive VPN Verbindung stoppen können. Die Verbindung muss in den ursprünglichen Zustand zurückgesetzt werden.

Passwort abfrage - V1.2 Dem Benutzer sollte beim Einrichten der Verbindungen die Möglichkeit geboten werden, Passwörter im System zu speichern oder dynamisch bei jedem Verbindungsaufbau nach dem Passwort gefragt zu werden. Die Art des Passwortzugriffs sollte auch in den Konfigurationsfiles der Verbindung gespeichert und bei jedem Verbindungsaufbau geprüft werden. Hat der Benutzer kein Passwort in der KWallet gespeichert, wird

er bei jedem Verbindungsaufbau mit einem Passwort Fenster aufgefordert sein Passwort einzutragen.

Fehlermeldungen - V1.6 Tritt ein Fehler beim Verbindungsaufbau auf, sollte dies dem Benutzer auf der Benutzeroberfläche mitgeteilt werden. Bei einem missglückten Verbindungsaufbau wird dem Benutzer die Fehlermeldung und das Konfigurationsfenster seiner Verbindung angezeigt, damit er Änderungen vornehmen und einen neuen Verbindungsaufbau starteten kann.

Statusänderung Visualisierung - V2 Das System muss dem Benutzer auf der Benutzeroberfläche den Status seiner VPN Verbindung visualisieren. Der Benutzer sieht dadurch ob die Verbindung aufgebaut werden konnte, oder der Vorgang fehlgeschlagen ist. Zudem sollte es möglich sein die Dauer der aktiven Verbindung zu sehen.

5.6 Use Cases

5.6.1 Use Case Diagramm

In Abbildung 5.1 ist das Use Case Diagramm dargestellt. Es beinhaltet die Use Cases, die in der Anforderungsphase festgelegt wurden.

5.6.2 Akteure

In dem zu entwickelnden System gibt es nur einen Akteur, der Benutzer. Er richtet die VPN Verbindung ein und verwaltet diese auch. Zudem startet und stoppt er die Verbindung. Es sollten keine weiteren Akteure direkt mit dem System interagieren. Sollte der Verbindungsaufbau nicht funktionieren und der Benutzer das Problem nicht selbst beheben können, muss der Benutzer den zuständigen Netzwerkadministrator kontaktieren. Der Netzwerkadministrator wird nicht direkt in das System eingreifen, sondern dem Benutzer die nötigen Hinweise geben, oder auf dem VPN Gateway Änderungen vornehmen. Er wird nicht auf dem Client-System aktiv, auf das sich meine Entwicklungen konzertieren.

5.6.3 Fully Dressed Use Cases

Use Case Name	**UC1 - VPN Verbindung einrichten**
Beschreibung	der Benutzer richtet eine strongSwan VPN Verbindung mit seinen Verbindungsdaten erstmalig ein
Beteiligte Akteure	Benutzer
Integrierte Use Cases	—–
Auslöser	Benutzer
Vorbedingungen	- KDE Version 4.4 - NM Plasmoid Applet installiert - strongSwan NM Plugin installiert

Nachbedingungen	- VPN Verbindung gespeichert - alle von Benutzer eingegeben Daten sind auf dem Filesystem gespeichert
Standardablauf	1. Benutzer öffnet die Verbindungskonfiguration 2. Benutzer selektiert, neue VPN Verbindung einrichten 3. Benutzer wählt strongSwan VPN Verbindung 4. Benutzer trägt seine Verbindungsdaten in die Felder ein 4.1. Gateway Adresse eingeben 4.2 Zertifikat auswählen 4.3 User Name eintragen 4.4 Passwort Speichermodus auswählen 4.4.1 Passwort eintragen 5. Benutzer speichert seine Verbindungsdetails
Alternative Ablaufschritte	4.3a User Zertifikat auswählen 4.4a User Private Key auswählen
Änderungsgeschichte	Version 2, Maurus Rohrer, 19.05.2010

Tabelle 5.6: Use Case Details: UC1 - VPN Verbindung einrichten

Use Case Name	**UC2 - VPN Verbindung ändern**
Beschreibung	der Benutzer ändert die Verbindungsdaten seiner unter UC1 eingerichteten strongSwan VPN Verbindung
Beteiligte Akteure	Benutzer
Integrierte Use Cases	——
Auslöser	Benutzer UC6 bei Verbindungsfehler
Vorbedingungen	- strongSwan VPN Verbindung eingerichtet - KDE Version 4.4 - NM Plasmoid Applet installiert - strongSwan NM Plugin installiert
Nachbedingungen	- VPN Verbindungsdaten geändert und gespeichert - alle vom Benutzer geänderten Daten sind auf dem Filesystem gespeichert.
Standardablauf	1. Benutzer öffnet die Verbindungskonfiguration 2. Benutzer selektiert vorhandene strongSwan VPN Verbindung 3. vorhandene Verbindungsdaten werden angezeigt 4. Benutzer ändert seine Verbindungsdaten in den entsprechenden Feldern 5. Benutzer speichert seine Verbindungsdetails
Änderungsgeschichte	Version 2, Maurus Rohrer, 19.05.2010

Tabelle 5.7: Use Case Details: UC2 - VPN Verbindung ändern

Use Case Name	**UC3 - VPN Verbindung löschen**
Beschreibung	der Benutzer löscht seine eingerichtete strongSwan VPN Verbindung

Beteiligte Akteure	Benutzer
Integrierte Use Cases	—–
Auslöser	Benutzer
Vorbedingungen	- strongSwan VPN Verbindung eingerichtet - KDE Version 4.4 - NM Plasmoid Applet installiert - strongSwan NM Plugin installiert
Nachbedingungen	- VPN Verbindung gelöscht - Konfigurationsdateien auf dem Filesystem gelöscht und Eintrag in der KDE-Wallet entfernt
Standardablauf	1. Benutzer öffnet die Verbindungskonfiguration 2. Benutzer selektiert vorhandene strongSwan VPN Verbindung 3. Benutzer löscht ausgewählte VPN Verbindung
Alternative Ablaufschritte	—–
Änderungsgeschichte	Version 1, Maurus Rohrer, 19.03.2010

Tabelle 5.8: Use Case Details: UC3 - VPN Verbindung löschen

Use Case Name	**UC4 - VPN Verbindung starten**
Beschreibung	der Benutzer startet seine strongSwan VPN Verbindung
Beteiligte Akteure	Benutzer
Integrierte Use Cases	SUB UC1
Auslöser	Benutzer UC6 bei Verbindungsfehler
Vorbedingungen	- strongSwan VPN Verbindung eingerichtet - KDE Version 4.4 - NM Plasmoid Applet installiert - strongSwan NM Plugin installiert
Nachbedingungen	- VPN Verbindung wird aufgebaut - Netzwerkeinstellungen des Systems werden durch den strongSwan Daemon geändert.
Standardablauf	1. Benutzer selektiert seine eingerichtete strongSwan VPN Verbindung 2. strongSwan VPN Verbindung startet den Verbindungsaufbau 3. strongSwan VPN Verbindung ist eingerichtet und verbunden
Alternative Ablaufschritte	3a strongSwan VPN Verbindung ist fehlgeschlagen - UC6
Änderungsgeschichte	Version 1.6, Maurus Rohrer, 19.04.2010

Tabelle 5.9: Use Case Details: UC4 - VPN Verbindung starten

Use Case Name	**UC5 - VPN Verbindung stoppen**
Beschreibung	der Benutzer stoppt eine aktive VPN Verbindung
Beteiligte Akteure	Benutzer
Integrierte Use Cases	—–

Auslöser	Benutzer
Vorbedingungen	- Aktive strongSwan VPN Verbindung
Nachbedingungen	- VPN Verbindung beendet - Verbindung wird in Startzustand zurückgesetzt
Standardablauf	1. Benutzer stoppt seine aktive strongSwan VPN Verbindung 2. strongSwan VPN Verbindung wird gestoppt
Alternative Ablaufschritte	—–
Änderungsgeschichte	Version 1, Maurus Rohrer, 21.03.2010

Tabelle 5.10: Use Case Details: UC5 - VPN Verbindung stoppen

Use Case Name	**UC6 - Verbindungsfehler bearbeiten**
Beschreibung	Benutzer ändert Verbindungsparameter nach erfolglosem Verbindungsaufbau
Beteiligte Akteure	Benutzer
Integrierte Use Cases	UC4, UC2
Auslöser	Benutzer
Vorbedingungen	- strongSwan Verbindung wurde gestartet - Verbindungsaufbau ist fehlgeschlagen
Nachbedingungen	- VPN Verbindung wird erneut aufgebaut und geändert - Verbindungsparameter sind im Filesystem gespeichert
Standardablauf	1. Benutzer wird mit Fehlermeldungs- und Verbindungsdetailfenster konfrontiert 2. Benutzer ändert die Verbindungsparameter 3. Benutzer speichert die geänderten Verbindungsparameter 4. Verbindung wird neu gestartet
Alternative Ablaufschritte	3a. Benutzer bricht Fehlerbehebung ab 4a. es werden keine Änderung gespeichert
Änderungsgeschichte	Version 1.6, Maurus Rohrer, 18.04.2010

Tabelle 5.11: Use Case Details: UC6 - Verbindungsfehler bearbeiten

Use Case Name	**SUB UC1 - Passwort Abfrage**
Beschreibung	der Benutzer muss während dem Verbindungsaufbau sein Passwort eingeben
Beteiligte Akteure	Benutzer
Integrierte Use Cases	UC4
Auslöser	UC4 Verbindungsaufbau
Vorbedingungen	- strongSwan Verbindung gestartet - Passwort Speichermodus auf “nicht speichern” gesetzt
Nachbedingungen	- VPN Verbindung wird aufgebaut - Netzwerkeinstellungen des Systems werden durch den strongSwan Daemon geändert

Standardablauf	1. während des Verbindungsaufbaus wird Passwort Eingabe Fenster geöffnet 2. Benutzer trägt sein Passwort ein 3. Benutzer drückt “Connect” Passwort wird dem strongSwan Daemon zur Authentifizierung weitergeleitet
Änderungsgeschichte	Version 1.2, Maurus Rohrer, 25.03.2010

Tabelle 5.12: Use Case Details: SUB UC1 - Passwort Abfrage

5.7 Nicht-funktionale Anforderungen

Daten Integrität Es muss darauf geachtet werden, dass die Benutzer-Passwörter vertraulich behandelt werden. Die Passwörter dürfen nur verschlüsselt auf der Festplatte gespeichert werden. Ist dies nicht der Fall, sollte der Benutzer darauf hingewiesen werden, oder er sollte explizit die Funktion wählen können. Es darf auf keinen Fall als Standard-Speicher-Variante ausgewählt sein.

Benutzbarkeit Das zu entwickelnde Userinterface muss sich an die KDE Userinterface Guidelines halten. Zudem sollte das Userinterface klar strukturiert und verständlich sein. Weiter muss es von der Internationalisierung, wie in KDE üblich, unterstützt werden.

Korrektheit der Fehlermeldungen Bei der Anzeige der Fehlermeldung, muss darauf geachtet werde, dass die angezeigten Fehlermeldungen der tatsächlichen Fehlerursache entsprechen. Sie sollten möglichst einfach und klar verständlich angezeigt werden.

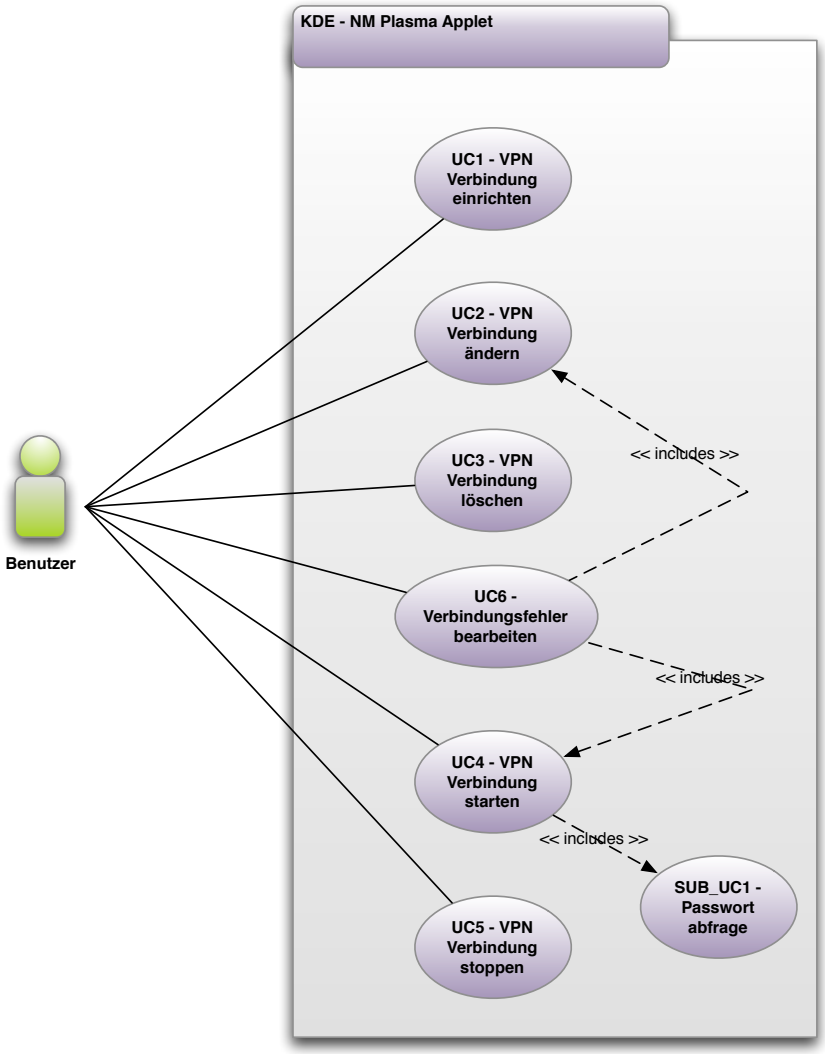


Abbildung 5.1: Use Case Diagramm

6 Design und Analyse

In diesem Kapitel wird das Design für die Entwicklung vorgestellt. Es soll verdeutlicht werden, wie die entwickelten Komponenten mit der bestehenden Architektur kommunizieren. Zudem wird jeweils hervorgehoben, welche Features vom Verfasser dieser Arbeit entwickelt wurden. Im Unterkapitel Architektur sind die Domainmodelle und Klassendiagramme dargestellt. Danach werden Sequenzdiagramme zu den verschiedenen Use Cases abgebildet. Abschliessend wird das Design und die Entstehung des Userinterfaces für die Fehlermeldungen dokumentiert.

Wie in den Grundlagen ersichtlich ist, müssen für das Plugin verschiedene bestehende Softwarepakete erweitert und geändert werden. Die involvierten Komponenten und ihre Zusammenhänge sollen kurz in Erinnerung geruft werden.

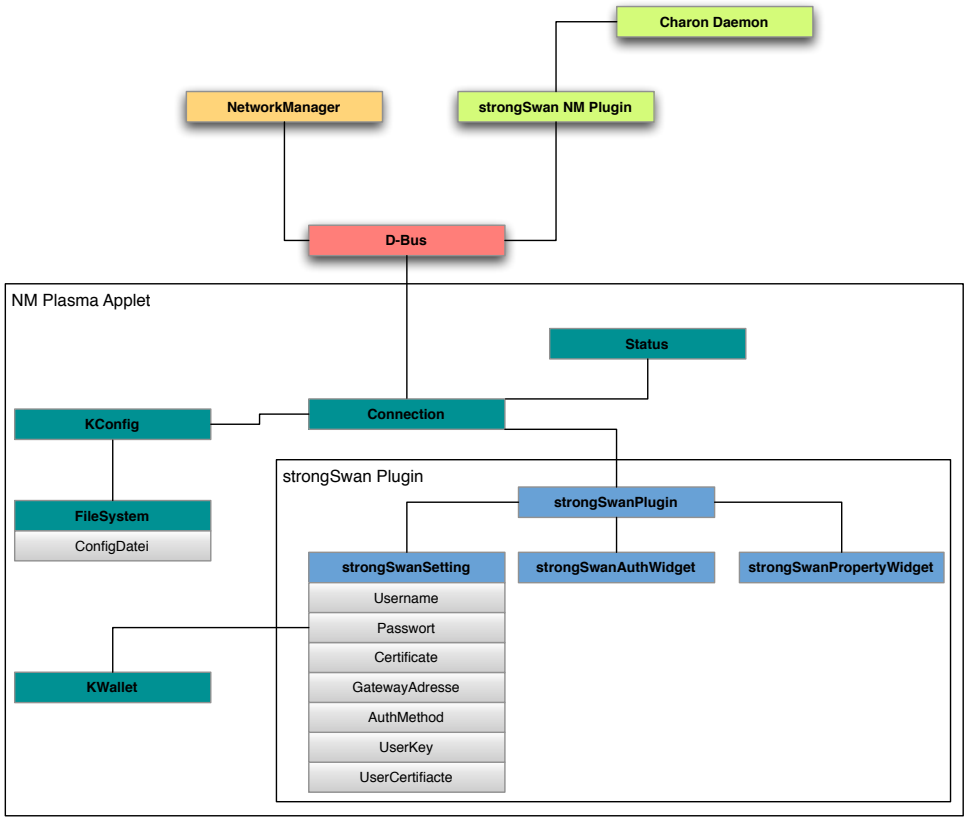


Abbildung 6.1: Architektur Übersicht

6.1 Architektur

6.1.1 strongSwan Plugin

In Abbildung 6.2 ist die Architektur des entwickelten strongSwan Plugins und des bestehenden NM Plasma Applets abgebildet. Es ist in Form eines Domainmodels gehalten und verdeutlicht die verschiedenen involvierten Objekte. Die hellblauen Elemente sind Userinterface-Elemente. Der gestrichelte Rahmen definiert die zu entwickelnden Objekte. Die anderen Objekte sind von NM Plasma Applet vorgegeben.

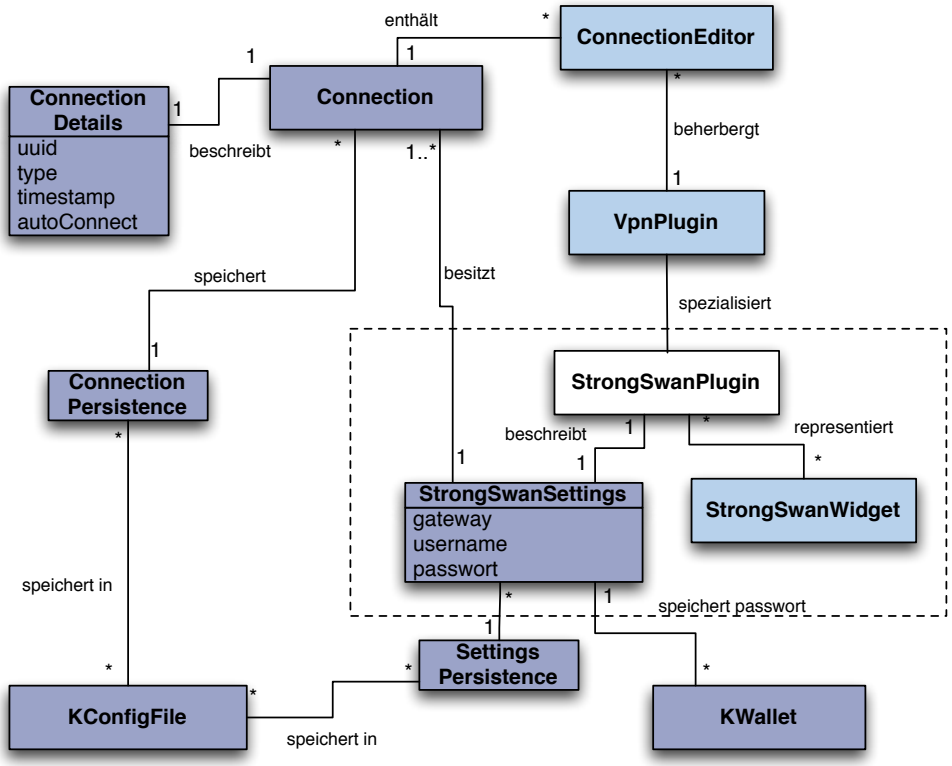


Abbildung 6.2: Domain strongSwan Plugin

6.1.2 Passwort Abfrage

Die Passwortabfrage wird von dem “ConnectionSecretJob” des NM Plasma Applets durchgeführt. Dieses liest das Passwort aus der KWallet oder fragt den Benutzter nach diesem. In der ursprünglichen Implementation wird nicht geprüft, ob eine VPN Verbindung das Passwort gespeichert hat oder nach diesem gefragt werden sollte. Es wird immer angenommen, dass es in der KWallet gespeichert wurde. Damit der Speichermodus einer Verbindung zugewiesen werden kann, wurden die Klassen in Abbildung 6.3 erweitert. Die blauen Klassen wurden neu hinzugefügt und die anderen mit den abgebildeten Methoden und Parametern erweitert.

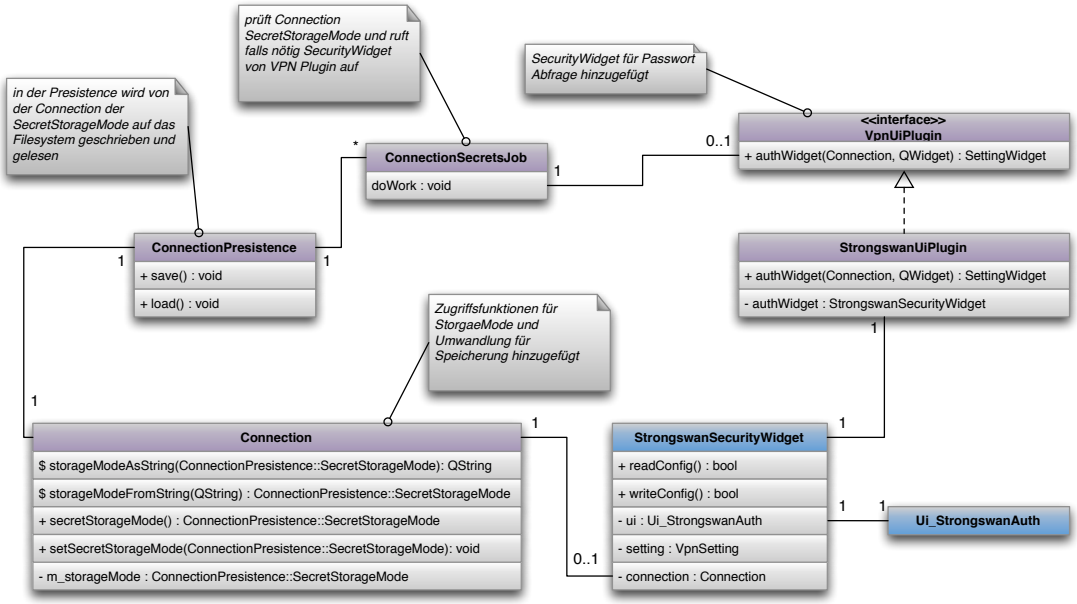


Abbildung 6.3: Klassendiagramm Passwort Abfrage

6.1.3 VPN Icon

Damit das NM Plasma Applet auf die Status Änderungen der VPN Connection reagiert, muss nur eine Klasse angepasst werden (NetworkManagerApplet). Die “NetworkManagerApplet” Klasse muss auf Signale anderer Klassen reagieren. In Abbildung 6.4 sind die involvierten Klassen abgebildet. Die aufgelisteten Methoden und Parameter der Klasse “NetworkManagerApplet” wurden neu hinzugefügt. Ist eine “RemoteInterfaceConnection” aktiv, wird im “NetworkManagerApplet” eine Referenz gespeichert, damit die ToolTip über die nötigen Informationen der aktiven Verbindung verfügt.

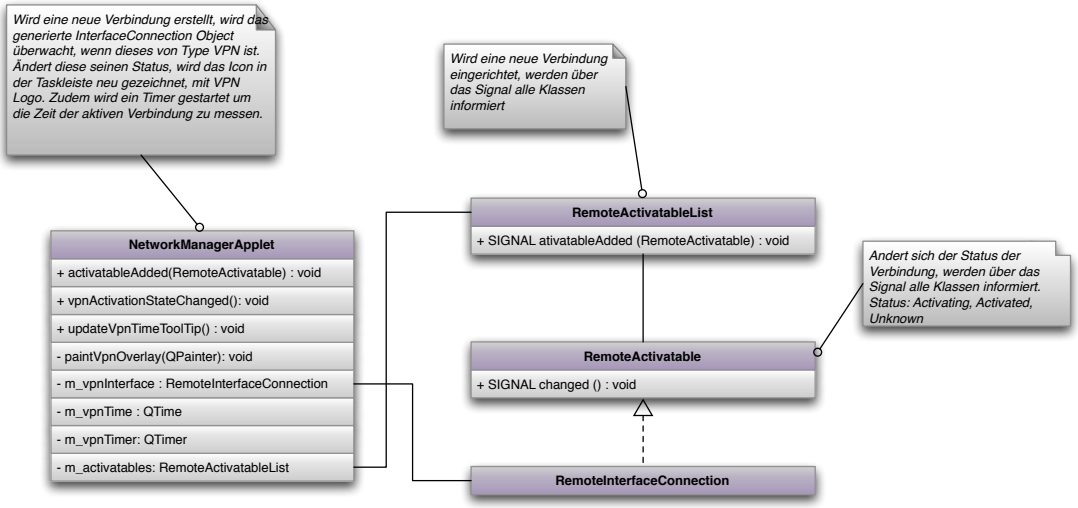


Abbildung 6.4: Klassendiagramm VPN Icon

6.1.4 Fehlermeldungen

In Abbildung 6.5 sind die involvierten Komponenten und ihre Klassen abgebildet. Da eine Fehlermeldung vom strongSwan Daemon über den NetworkManager bis hin zum NM Plasma Applet delegiert werden muss, werden einige Klassen in diesen Ablauf eingebunden. Die blau markierten Klassen und Funktionen mussten neu hinzugefügt werden, die anderen Klassen und Methoden wurden angepasst. Was genau geändert werden musste, ist in Abschnitt 3.4.3 nachzulesen. Die Kommunikation zwischen den Komponenten läuft ausschliesslich über den D-Bus. Im strongSwan Daemon läuft die Kommunikation über den strongSwan Bus. Im NM Plasma Applet wird mit Referenzen gearbeitet.

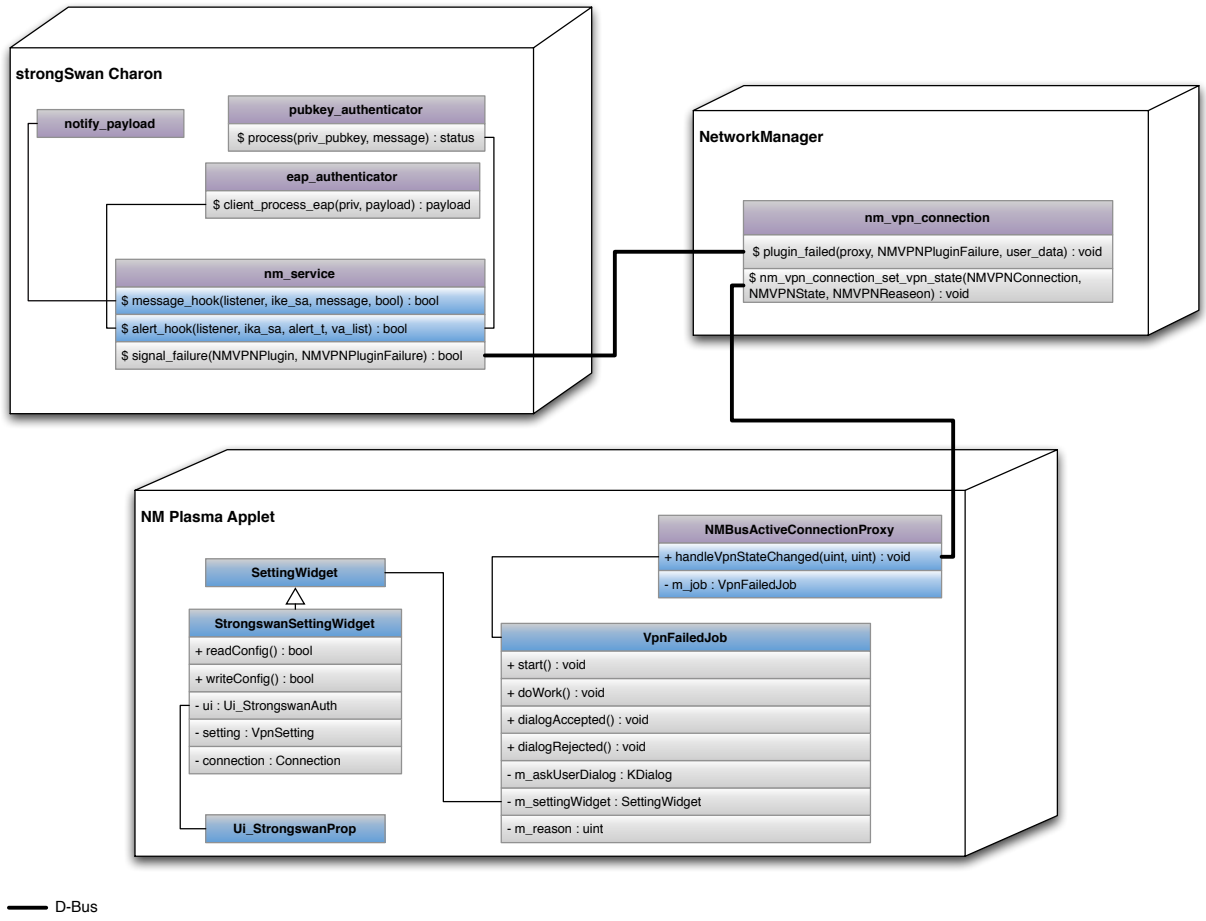


Abbildung 6.5: Klassendiagramm Fehlermeldungen

6.2 Sequenzdiagramme

6.2.1 strongSwan Plugin

Das Sequenzdiagramm (Abbildung 6.6) beschreibt den Ablauf beim Anlegen oder Ändern einer Verbindung. Der Ablauf des Anlegens einer Verbindung ist identisch zum Vorgang des Änderns. Wird eine neue Verbindung angelegt, wird das Laden der Settings fehlschlagen, da diese noch nicht vorhanden sind und das strongSwan Verbindungsfenster (strongSwanPropertyWidget) wird ohne Parameter initialisiert. Wird eine Verbindung geändert, können die Settings aus den Konfigurationsfiles geladen werden und das Verbindungsfenster mit den geladenen Parametern initialisiert.

6.2.2 Passwort Abfrage

Im Sequenzdiagramm (Abbildung 6.7) ist der Ablauf dargestellt, der beim Laden des Passwortes durchgeführt wird. Auf dem D-Bus Service *VPN.Plugin* gibt es eine Methode “NeedSecrets(a{sa{sv}})”. Diese wird ausgeführt, wenn der strongSwan Daemon ein Passwort benötigt. Im dargestellten Sequenzdiagramm ist der Ablauf, der beim Aufruf der Methode auf NM Plasma Applet Seite ausgeführt wird, abgebildet. Neu von mir implementiert ist die Alternative “storageMode = Ask”.

6.2.3 Fehlermeldungen

Im Sequenzdiagramm 6.8 wird eine Fehlermeldung verfolgt, die vom Gateway über eine *Notify Message* gesendet wurde (z.B: AUTHENTICATION_FAILED). Der *Listener* übermittelt die Nachricht dem strongSwan Plugin, das die Fehlermeldung in den NetworkManager Aufzählungstyp “reason” “NM_VPN_PLUGIN_FAILURE_AUTH_FAILED” wandelt. Danach wird die “SetFailure(u:reason)” Methode auf dem D-Bus aufgerufen. Anschliessend beendet sich der Charon Daemon und wird erst beim nächsten Verbindungsaufbau wieder gestartet. Der ”nm_vpn_manager“ des NetworkManagers wandelt den erhaltenen *Failure Reason* in einen *Status Reason* (NM_VPN_CONNECTION_STATE_REASON_LOGIN_FAILED). Danach wird die Methode ”VpnStateChanged (u:state, u:reason)“ auf dem D-Bus aufgerufen. Der Status wird dabei auf *FAILED* gesetzt. Auf Seiten des NM Plasma Applets wird nun der ”ActiveConnectionProxy“ angesprochen. Dieser startet einen ”VpnFailedJob“, der den Benutzer über die Fehlermeldung informiert und das Fehlermeldungsfenster anzeigt.

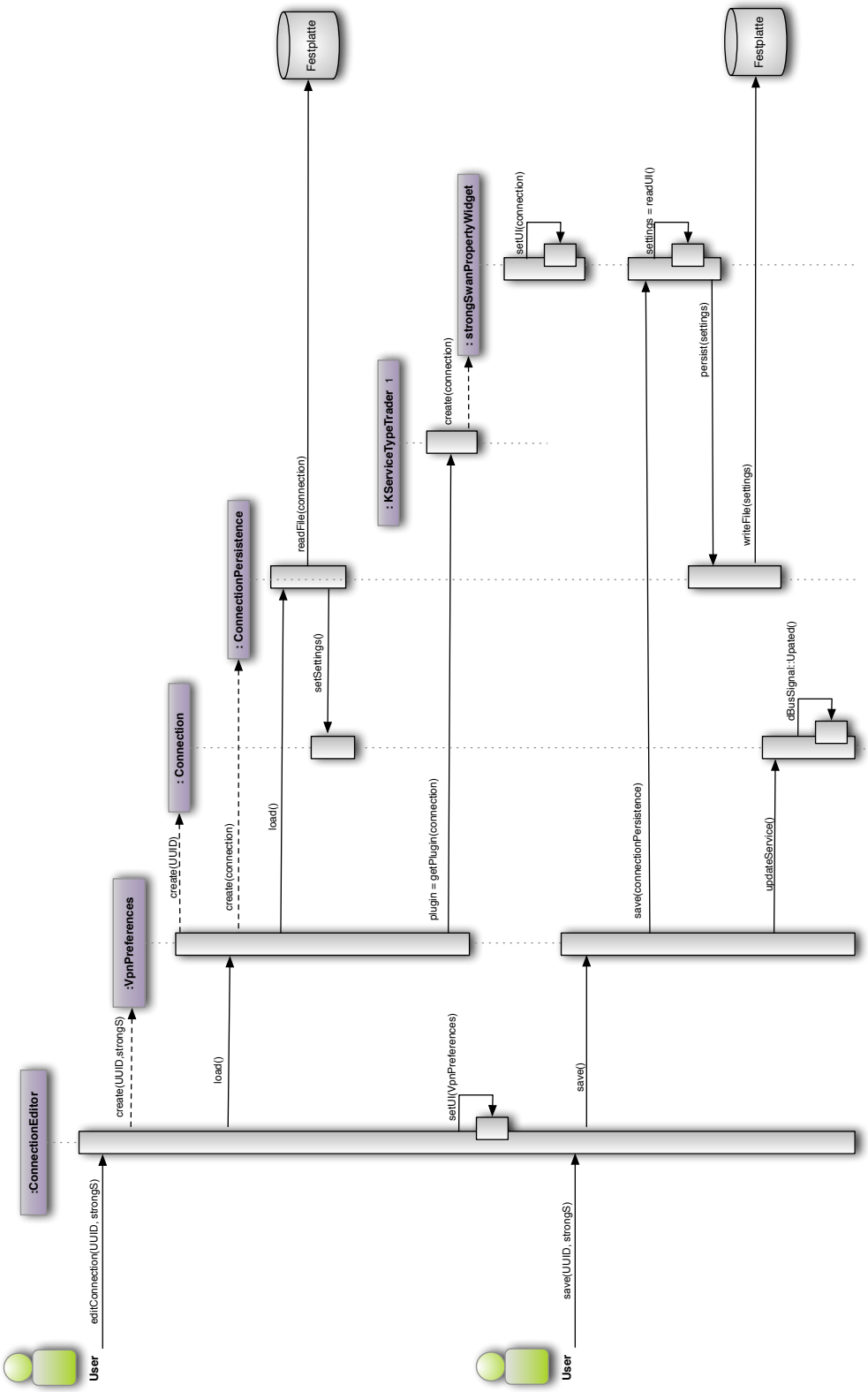


Abbildung 6.6: Sequenzdiagramm - ändern und speichern einer Verbindung

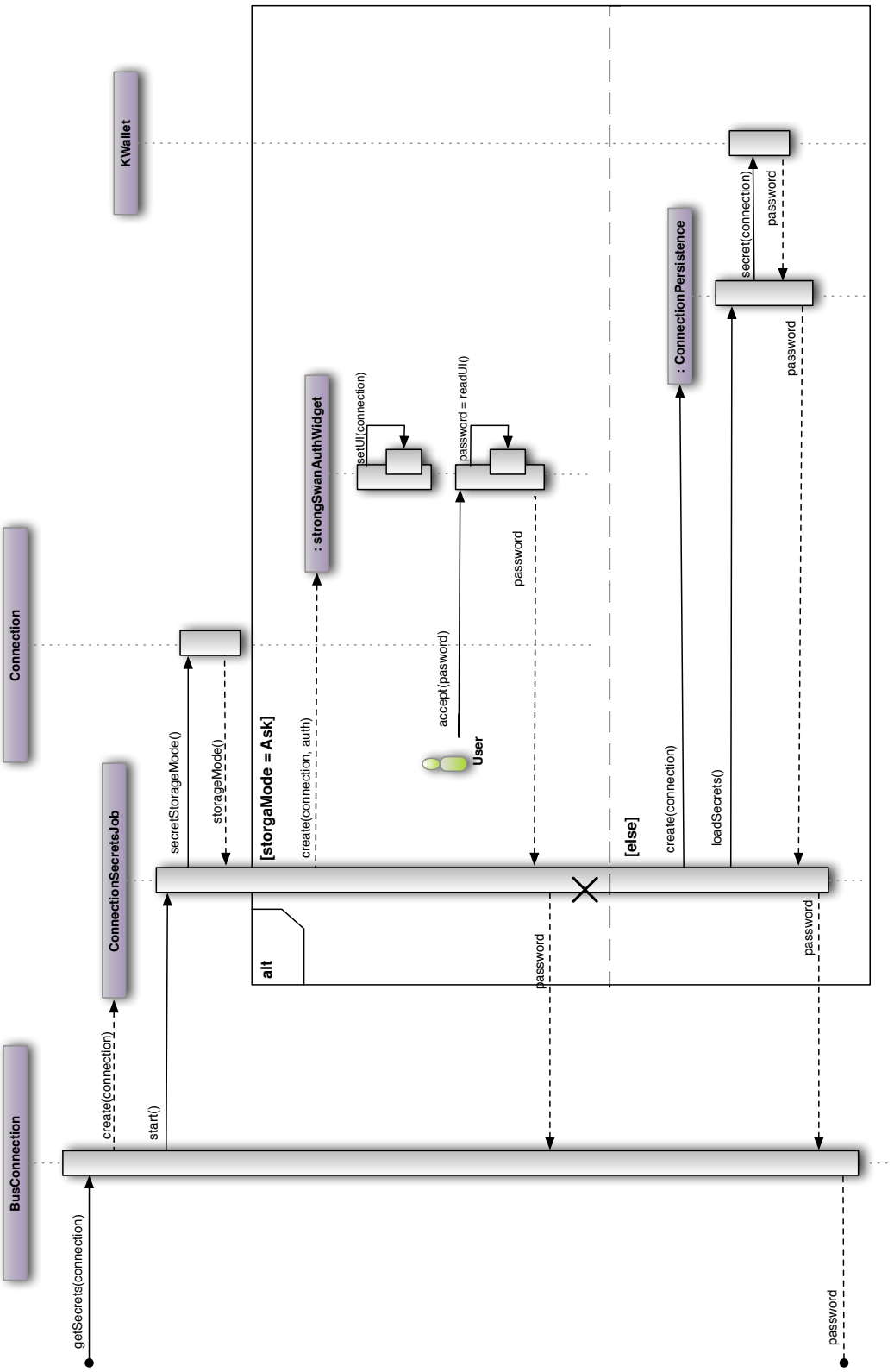


Abbildung 6.7: Sequenzdiagramm - Passwort Abfrage

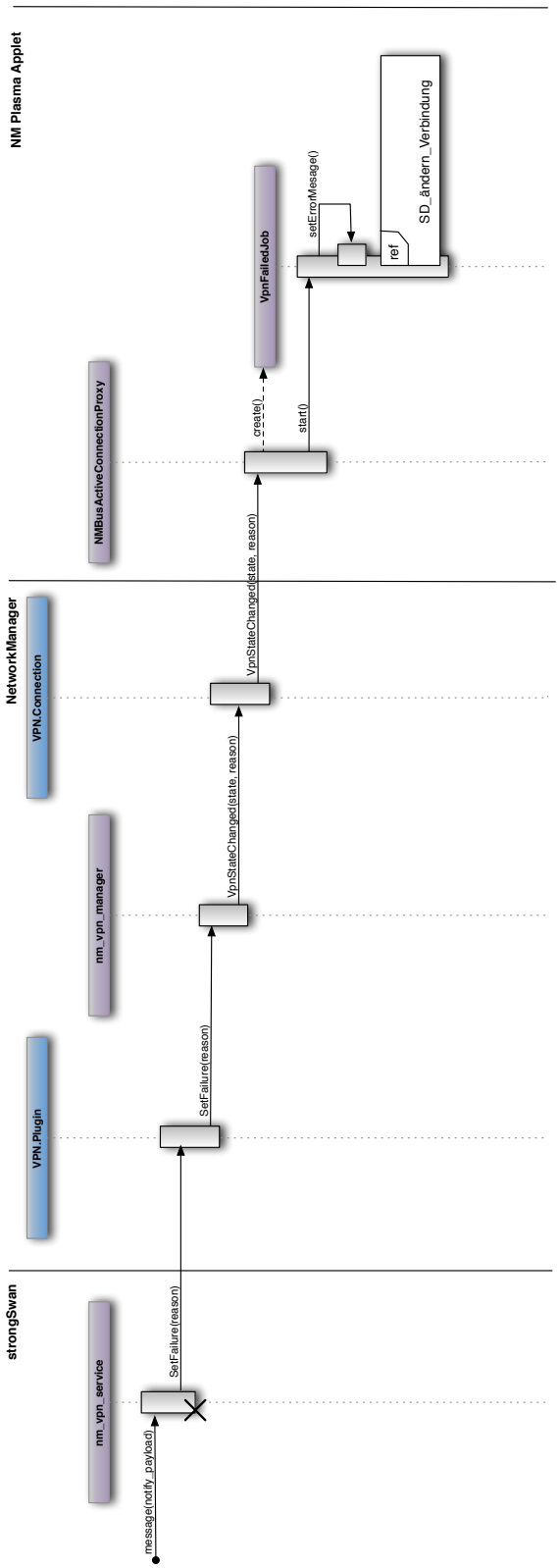


Abbildung 6.8: Sequenzdiagramm - Fehlermeldungen

6.3 UI Design

In diesem Abschnitt sind die Entwürfe und die Entwicklung des Userinterfaces dargestellt. Da das zu entwickelnde GUI in das bestehende GUI der KDE Oberfläche integriert werden sollte, müssen die GUI Richtlinien von KDE eingehalten werden. Wie aus den Use Cases ersichtlich ist, verlangen diese, dass zwei GUI's entwickelt werden. Das erste GUI ist das Eingabefenster für die Verbindungsdetails. Dieses bietet keinen Spielraum in der Gestaltung, da es in das existierende Fenster einer Verbindung integriert werden muss. Das GUI, das bei einem fehlgeschlagenen Verbindungsaufbau dargestellt wird, kann jedoch entworfen werden. Ein solches Fehlermeldungsfenster kombiniert mit den Verbindungsdetails existiert im Standard von KDE noch nicht.

6.3.1 GUI Version 1

Der erste GUI Entwurf hält sich in seiner Form an die in KDE typische Handhabung von Fehlern. Beim Fehlschlagen des Aufbaus wird ein Mitteilungsfenster (KDE MessageBox) angezeigt. In dieser MessageBox ist die Fehlermeldung eingebettet. Der Benutzer hat nun die Möglichkeit über "Continue" auf eine zweite Ansicht zu gelangen, in der die Verbindungsdetails angezeigt werden. Er kann dann die Verbindungsdetails ändern und die Verbindung erneut startet. Es wird neben dem "Continue" Button auch ein "Cancel" Button angeboten, über den der Benutzer den Verbindungsaufbau abbrechen kann. Der Abschnitt, der den Verbindungsaufbau verhindert hat, wird rot selektiert. In Abbildung 6.9 ist das GUI der ersten Version ersichtlich.

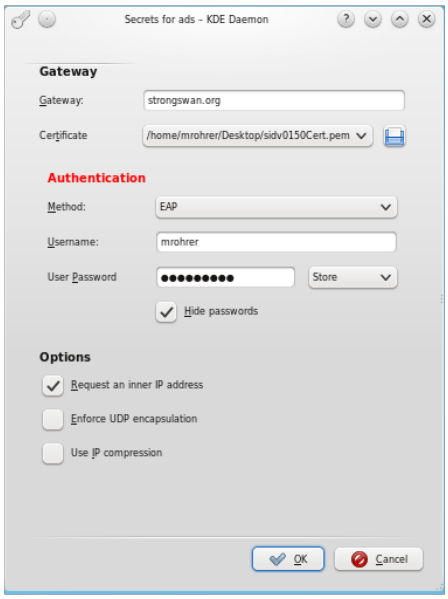


Abbildung 6.9: GUI Version 1

6.3.2 Redesing Entscheide Version 1

Diverse Tests (Abschnitt 8.2) haben gezeigt, dass die GUI Version sehr benutzerunfreundlich ist. Die Testperson wusste nach dem Schliessen der MessageBox nicht mehr, was die eigentliche

Fehlermeldung war. Nach dem Schliessen des Fensters, gibt es keine Möglichkeit mehr die Fehlermeldung einzusehen. Zudem wurde das Bestätigen der MessageBox von der Testperson als überflüssig empfunden. Will der Benutzer den Fehler beheben, muss er sich durch die Fenster klicken.

Folgende Redesing Entscheide wurden getroffen:

- Keine Mitteilungsfenster
- Nur ein Fenster, das Fehlermeldung und Verbindungsdetails beinhaltet

6.3.3 GUI Version 2

Aus den Erkenntnissen der Tests wurde eine zweite Version erarbeitet. Die MessageBox wurde weggelassen und dem Benutzer wird nach fehlgeschlagenem Verbindungsaufbau das Verbindungsdetail-Fenster kombiniert mit einer MessageBox angezeigt. Die Fehlermeldung ist nun oben im neuen Fenster positioniert. Unter der Fehlermeldung sind die Verbindungsdetails abgebildet. So hat der Benutzer die Möglichkeit die Fehlermeldung zu lesen und dann gleich ohne weiter zu klicken die Änderungen vorzunehmen. Zudem kann er jederzeit die Fehlermeldung lesen. Die GUI Version 2 ist in Abbildung 6.10 dargestellt.

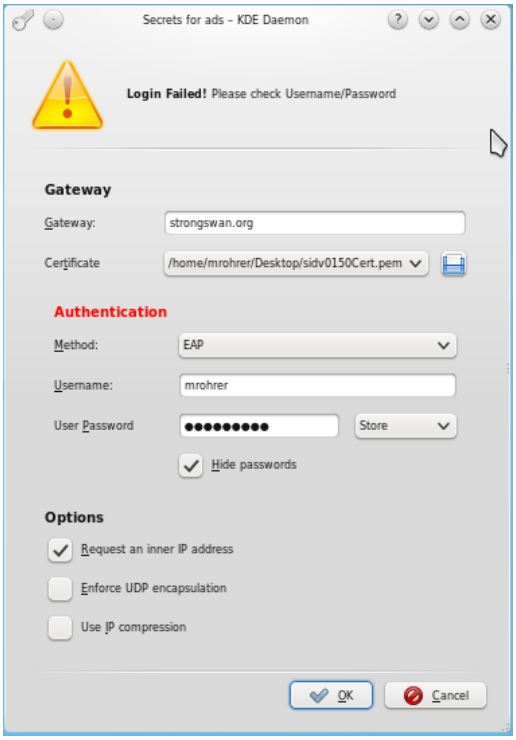


Abbildung 6.10: GUI Version 2

6.3.4 Redesing Entscheide Version 2

Die GUI Version 2 ist bei der Testperson besser angekommen. Sie präsentiert alle Informationen in einem Fenster. Jedoch wurde festgestellt, dass sich die Testperson im Fenster nicht zurecht-

findet. Der Abschnitt mit der Fehlermeldung ist zu gross und am falschen Ort. Der Benutzer hatte sich instinktiv an den Buttons orientiert. Deshalb wurde der Entscheid getroffen die Fehlermeldung über die Buttons zu positionieren. Die rote Schrift des Problembereichs wurde optisch als störend empfunden, jedoch sah der Benutzer dadurch auf einen Blick, wo der Fehler zu finden war.

Folgende Redesing Entscheide wurden getroffen:

- Fehlermeldung kleiner
- Fehlermeldung oberhalb der Buttons
- Rote Schrift weggelassen
- Problembereich durch einen Rahmen verdeutlichen

Zudem wurden die neuen KDE Guidelines berücksichtigt [eK10d]. Neu in KDE 4 müssen in Tabellen die Beschriftung und das Eingabefenster (Abbildung 6.11) angeordnet werden.

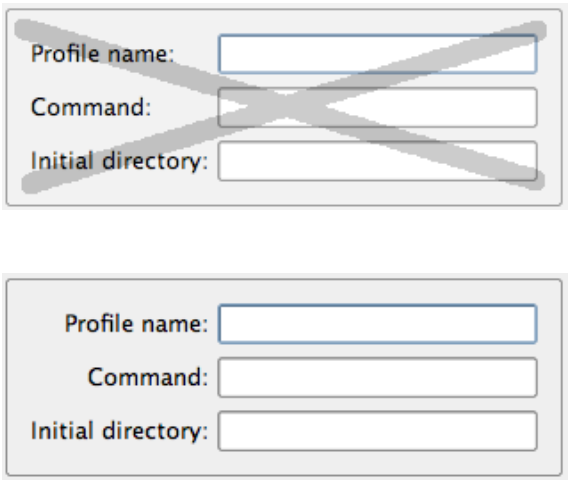


Abbildung 6.11: KDE GUI Guideline [eK10d]

6.3.5 Finales GUI

In Abbildung 6.12 ist die endgültige Version des GUI zu sehen.

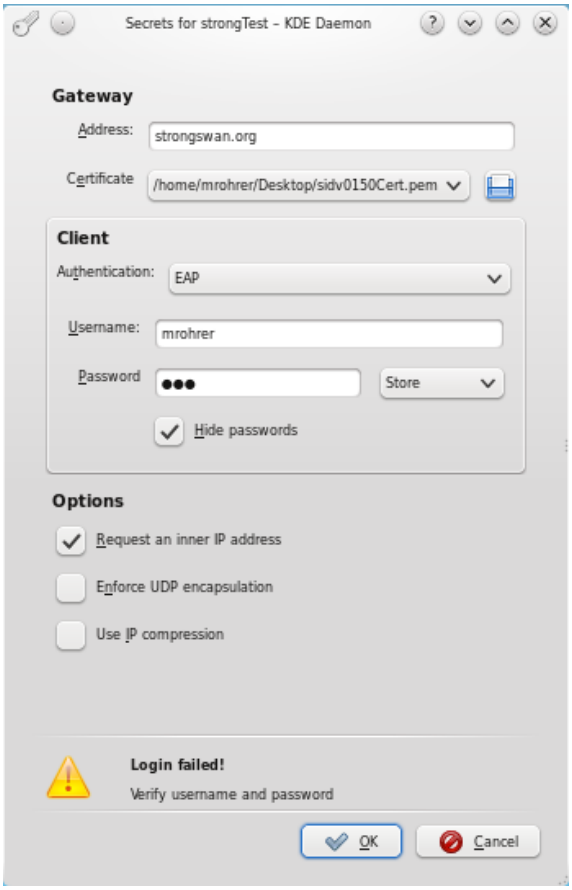


Abbildung 6.12: Final GUI

7 Implementation

In diesem Kapitel wird beschrieben wie die Entwicklungsumgebung von KDE 4 aufgesetzt wird und wie man mit ihr arbeiten kann. Weiter werden die verschiedenen Tools vorgestellt, die während der Arbeit gebraucht wurden. Der entwickelte Quellcode wird hier nicht detailliert aufgelistet. Der Code ist jedoch auf der beigelegten DVD (02_Code) abgelegt und kann da eingesehen werden.

7.1 Entwicklungsumgebung

Damit mit der aktuellen KDE Umgebung gearbeitet werden kann, muss die Entwicklungsumgebung aufgesetzt werden. Da im NM Plasma Applet Erweiterungen vorgenommen werden, muss die komplette Entwicklungsumgebung (Desktop, Runtime, Libs) von KDE installiert werden. Würden nur einzelne Plasmoids entwickelt werden, würden die KDE Librarys genügen.

Vorbereitungen

Als Erstes muss ein Benutzer angelegt werden. Das Anlegen des Benutzers sollte mit *root* Rechten durchgeführt werden:

```
useradd -m kde-devel -s /bin/bash
passwd kde-devel
```

Damit der Benutzer über die nötigen Entwicklungsumgebungsvariablen und Verzeichnisse verfügt, muss das Login-Script ".bashrc" im Home Verzeichnis geändert werden. Das Login-Script ".bashrc" File kann von der KDE Homepage[1] heruntergeladen werden. Um das Login-Script zu testen, muss "*source ./bashrc*" ausgeführt werden. Danach sind die Umgebungsvariablen und Verzeichnisse für die Entwicklung aufgesetzt.

Damit die KDE Umgebung kompiliert werden kann, müssen diverse Softwarepakete installiert werden. Je nach Distribution werden unterschiedliche Pakete verwendet. Unter Kubuntu/Ubuntu und Debian müssen folgende Pakete installiert sein:

```
sudo aptitude install build-essential kde-devel xorg-dev cdbs debhelper cmake \
kdesdk-scripts subversion ssh xserver-xephyr doxygen dbus-x11 libxine-dev\
libxml2-dev libxslt1-dev shared-mime-info libical-dev libgif-dev libssl-dev \
libboost-dev libboost-program-options-dev libboost-graph-dev libgpgme11-dev \
libqimageblitz-dev libbz2-dev libdbus-1-dev libpam0g-dev libpcre3-dev \
```

[1] Login-Script: http://techbase.kde.org/Getting_Started/Increased_Productivity_in_KDE4_with_Scripts/.bashrc

```
libkrb5-dev libsm-dev libclucene0ldbl libclucene-dev libjpeg62-dev \
libxtst-dev xsltproc libxrender-dev libfontconfig1-dev automoc librdf0-dev
```

Zudem muss die neuste Version (min. 4.6) von QT installiert sein. Auf der QT Homepage[2] ist der Quellcode (Sourcen) erhältlich. Die heruntergeladen Sourcen müssen wie folgt kompiliert und installiert werden.

```
tar -xvf qt-everywhere-opensource-src-4.6.X.tar.gz
cd qt-everywhere-opensource-src-4.6.X
./configure
make
make install
```

Das Kompilieren der QT Librarys wird je nach verwendeter Hardware bis zu mehreren Stunden dauern. Die neuen QT Librarys sind nun im dem Verzeichnis "/usr/local/Trolltech/QT-4.6.X" installiert. Damit die KDE Entwicklungsumgebung von den neu kompilierten Librarys Gebrauch machen kann, muss im ".bashrc" Login-Script die Variable "QTDIR" angepasst werden:

```
export QTDIR=/usr/local/Trolltech/Qt-4.6.2
```

KDE Sourcen

Nun können die aktuellen Sourcen von KDE heruntergeladen und installiert werden. Die Sourcen sind in dem KDE svn[3] öffentlich zugänglich. Es sollte jedoch darauf geachtet werden, dass nur die nötigen Sourcen heruntergeladen werden, da sonst das svn von KDE überlastet wird. Es kann auch über das Web auf das svn zugegriffen werden [eK10a].

Mit den folgenden Befehlen wird das svn Lokal angelegt. Es wird jedoch noch nichts heruntergeladen. Der "cs" Befehl befördert den Entwickler in das "src" Verzeichnis der Entwicklungsumgebung.

```
cs
svn checkout --depth empty svn://anonsvn.kde.org/home/kde/trunk
```

Danach können die einzelnen KDE Sourcen heruntergeladen werden. Als erstes werden die "kdesupport" Librarys geladen und kompiliert.

```
cs trunk
svn up kdesupport && cd kdesupport
cmakekde
```

Mit dem Befehl "svn up .." werden die Sourcen heruntergeladen und mit "cmakekde" werden die Librarys kompiliert und installiert. Mehr zu "cmake" ist in Abschnitt 7.2 zu finden. Sind die Support Librarys installiert, müssen die KDE Librarys kompiliert werden. Damit diese kompiliert werden

[2] QT-Sourcen: http://qt.nokia.com/downloads
[3] Versionsverwaltung Tool von Apache

können, muss die aktuellste Phonon (Multimedia Framework von QT) Version vorhanden sein. Phonon wird beim Kompilieren der QT-Librarys nicht installiert. Dazu muss in das Verzeichnis “cs trunk/kdesupport/phonon” gewechselt werden. Darin befindet sich ein “README” File, das beschreibt wie die aktuellste Version von Phonon geladen werden kann. Sind die Phonon Sourcen geladen, können sie mit “cmakekde” installiert werden. Nun können die KDE Librarys heruntergeladen und kompiliert werden:

```
cs trunk
svn up --depth empty KDE
svn up KDE/kdelibs && cd KDE/kdelibs
cmakekde
```

Der KDE Desktop benötigt die “kdepimlibs”. Damit diese installiert werden können, muss zuerst “libical” installiert werden:

```
tar xvzf libical-0.41.tar.gz
cd libical-0.41
./configure && make && make install
```

Nun können die “kdepimlibs” gebildet werden:

```
cs trunk
svn up KDE/kdepimlibs && cd KDE/kdepimlibs
cmakekde
```

Und schliesslich wird die “kdebase”, die den Desktop und diverse Applikationen beinhaltet, gebildet:

```
cs trunk
svn up KDE/kdebase && cd KDE/kdebase
cmakekde
```

Nun ist die Entwicklungsumgebung aufgesetzt. Es fehlen nur noch die NM Plasma Applet Sourcen. Diese sind unter folgender WebSvn URL[4] erhältlich und befinden sich noch im “kdereview” Verzeichnis. Sobald sie vollumfänglich getestet sind, werden sie in das “KDE” Verzeichnis verschoben. Sie können wie folgt installiert werden:

```
cs trunk
svn up --depth empty kdereview
svn up kdereview/networkmanagement && cd kdereview/networkmanagement
cmakekde
```

Entwicklungsumgebung starten

Sobald die KDE Sourcen installiert sind, kann die Entwicklungsumgebung gestartet werden. Damit Änderungen im NM Plasma Applet getestet werden können, muss eine neue KDE Session

[4] NM Plasma Applet: http://websvn.kde.org/trunk/kdereview/networkmanagement/

gestartet werden. Zum Starten einer Session müssen folgende Anpassungen im System gemacht werden:

Als Erstes braucht der kde-devel Benutzer die Berechtigung auf dem Bildschirm Applikationen darzustellen. Dies wird über folgenden Befehl ermöglicht:

```
xhost +local:kde-devel
```

Am besten wird diese Zeile in das Login-Skript des eingeloggten Benutzers (nicht kde-devel) kopiert.

Danach muss ein Start-Script kreiert werden, das die neue Entwicklungsumgebung startet.

```
#! /bin/bash
NESTED_KDE_DISPLAY_BACKUP=$DISPLAY
export DISPLAY=:0
Xephyr :1 -screen 1024x768 &
sleep 2
export DISPLAY=:1
$HOME/kde/bin/startkde &
export DISPLAY=${NESTED_KDE_DISPLAY_BACKUP}
```

Das Script muss dann über “chmod +x start-script.sh” ausführbar gemacht werden. Nun kann der kde-devel Benutzer das Start-Script ausführen und schon wird die Entwicklungsumgebung gestartet. In Abbildung 7.1 ist die Entwicklungssession abgebildet. [eK10c]

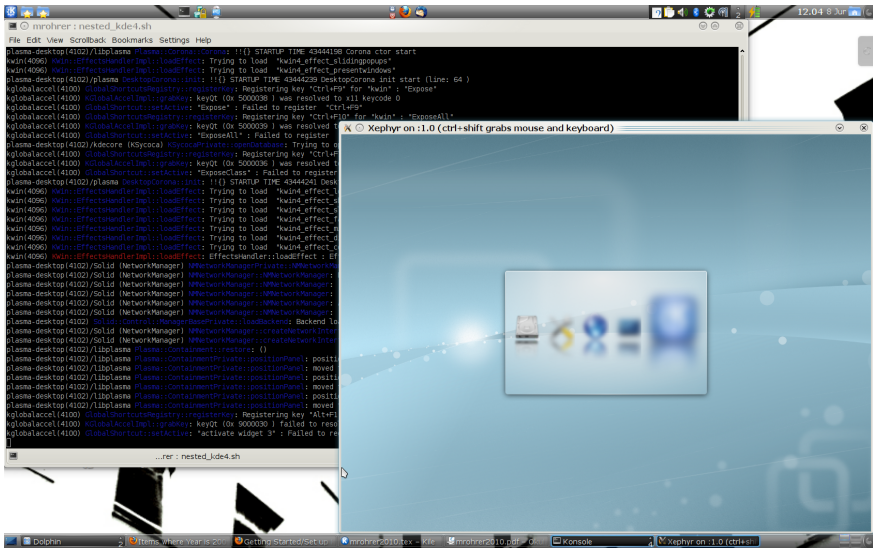


Abbildung 7.1: Entwicklungsumgebung Session

Entwicklungen

Jetzt kann der Sourcecode im NP Plasma Applet ($HOME/kde/src/kdereview/networkmanagement) geändert werden. Will man den varierten Sourcecode kompilieren, muss der Befehl “cmakekde”

ausgeführt werden. Kann “cmakekde” erfolgreich durchgeführt werden, sind die Änderungen installiert und die Entwicklungsumgebung kann neu gestartet werden. Danach sind die Änderungen sichtbar. Fügt man neue Sourcecode Files hinzu, müssen diese “cmake” bekannt gemacht werden. In jedem Verzeichnis befindet sich eine “CMakeLists.txt” Datei. Das neu erstellte Sourcecode File muss dem “CMakeLists.txt” File hinzugefügt werde, damit es kompiliert wird:

```
set(strongswan\_SRCS
  strongswan.cpp
  strongswanwidget.cpp
  strongswanauth.cpp //neue Datei
  )
```

7.2 Entwicklungstools

In diesem Abschnitt werden die Tools vorgestellt, die während der Implementierung genutzt wurden. Die Sourcecodes wurden mit dem Texteditor Kate[5] geändert und erstellt. Durch den Gebrauch von “cmake” ist ein spezielles Entwicklungswerkzeug im Stile von Eclipse, Netbeans überflüssig. Für die Gestaltung der Userinterfaces wurde das Werkzeug “QT Designer” benutzt.

QT Designer

QT Designer ist ein Tool zum Erstellen und Designen von Userinterfaces (GUIs). Es können Widgets oder Dialoge erstellt werden und über Drag-and-Drop GUI Elemente wie z.B: Labels, Buttons, usw. in die Widgets positioniert werden. QT Designer bietet verschiedene Layouts an damit die Userinterface-Elemente nach belieben positioniert werden können. Das “Signal-Slot” Prinzip von QT wird dabei vollumfänglich unterstützt. Signale und Slots können im QT Designer verknüpft werde. In Abbildung 7.2 ist ein Screenshot des QT Designers abgebildet.

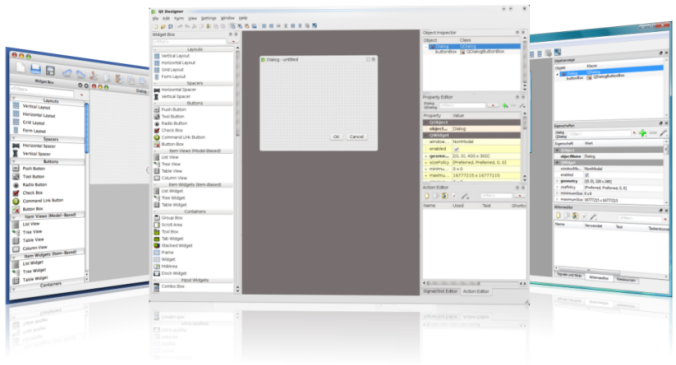


Abbildung 7.2: QT Designer [Tro10]

Die in QT Designer erstellten Widgets werden mit der Endung “.ui” gespeichert. Damit diese in KDE mit kompiliert werden, muss das “.ui” File in “CMakeLists.txt” hinzugefügt werden:

[5] Kate Editor: http://kate-editor.org/

```
kde4_add_ui_files(strongswan\_SRCS strongswanprop.ui)
```

Beim Kompilieren von KDE wird nun eine “ui_strongswanprop.h” Headerdatei erstellt. Diese Headerdatei kann nun im Quellcode (*strongswanwidget.cpp*) inkludiert werden. Damit im Quellcode auf das Userinterface zugegriffen werden kann, muss eine UI Variable definiert werden:

```
Ui_StrongswanProp ui;
```

Im Konstruktor der Quellcodedatei wird das Userinterface initialisiert und aufgesetzt:

```
  ui.setupUi(this);
```

Nun kann an beliebiger Stelle im Quellcode über das “ui” Objekt auf die Userinterface-Elemente zugegriffen werden. So kann z.B. der Text eines Eingabefensters gesetzt oder gelesen werden:

```
  ui.gateway->setText(gateway);
  QString gateway = ui.gateway->text();
```

cmake

CMake (cross-platform make) ist ein plattformunabhängiges Programmierwerkzeug zur Entwicklung und Erstellung von Software. Damit “cmakekde” ausgeführt werden kann, muss im Verzeichnis eine “CMakeLists.txt” Datei vorhanden sein. In der “CMakeLists.txt” sind die zu kompilierenden Quellcodedateien aufgelistet. Im Loginscript des Entwicklungsbenutzers ist “cmakekde” definiert. Durch die Definition von “cmakekde” kann der Entwickler in einem Schritt seinen Sourcecode kompilieren und installieren.

Beispiel des CMakeLists.txt des strongSwan Plugins:

```
include_directories(${CMAKE_CURRENT_SOURCE_DIR}/../../libs/ui)
include_directories(${CMAKE_CURRENT_SOURCE_DIR}/../../libs/internals)
set(strongswan_SRCS
   strongswan.cpp
   strongswanwidget.cpp
   strongswanauth.cpp
   )
kde4_add_ui_files(strongswan_SRCS strongswanprop.ui strongswanauth.ui)
kde4_add_plugin(networkmanagement_strongswanui ${strongswan_SRCS})
target_link_libraries(networkmanagement_strongswanui ${KDE4_KIO_LIBS} knminternals
 knmui)
install(TARGETS networkmanagement_strongswanui  DESTINATION ${PLUGIN_INSTALL_DIR})
install(FILES networkmanagement_strongswanui.desktop DESTINATION ${SERVICES_
 INSTALL_DIR})
```

8 Tests

In diesem Kapitel werden die durchgeführten Tests dokumentiert. QT bietet ein Framework für Unittests (QtTestLib). Neben den funktionalen QT Unittests, wurde das Userinterface auf die Bedienbarkeit getestet. Eine Testperson hat unter Aufsicht des Entwicklers das Userinterface bedient. Die daraus folgenden Erkenntnisse wurden in das Design des Userinterface integriert.

Die Kommunikation zwischen NM Plasma Applet und strongSwan Daemon lassen sich nicht mit dem *QTTestLib* Framework testen. Die Verbindungstests wurden repetitiv durchgeführt. Es wurden diverse Verbindungsaufbau-Versuche mit fehlenden und falschen Parameter durchgeführt. Die daraus folgenden Erkenntnisse wurden direkt in den Code integriert. Zudem wurde das Plugin am LinuxTag 2010 in Berlin zu Demonstrationszwecken eingesetzt.

8.1 QT Unittests

Das QT Test Framework (QTTestLib) prüft die Funktionalität, das Verhalten und die “Korrektheit” einzelner Software Komponenten (C++ Klassen). Jeder Testfall ist eine eigene Applikation und wird individuell aufgerufen. Neben den Code Tests bietet *QTTestLib* Möglichkeiten GUI’s und das “Signal and Slot” Prinzip von QT zu testen. Im NM Plasma Applet sind einzelne Testklassen vorhanden, welche zum Beispiel das Speichern der Verbindungsdetails über das KConfig Framework testen. Die von mir entwickelten Erweiterungen wurden in diesen Testklassen geprüft (TestConnectionListPresistence).

8.2 GUI Tests

Die verschiedenen GUI Versionen (Abschnitt 6.3) wurden von einer Testperson geprüft, welche aus Datenschutzgründen anonym bleibt. Die Testperson erhielt eine Aufgabe, die sie selbständig durchführen musste. Ihr Verhalten und ihre Reaktion wurden während des Tests analysiert. Darauf basierend wurde entschiden, wie das Redesing aussehen muss.

8.2.1 Test Persona Student Strebig

Student Sterbig ist 25 Jahre alt und absolviert in Zürich sein Master Studium in Geschichte. Er lebt in Zürich in einer Wohngemeinschaft mit drei Kollegen, ebenfalls Studenten. Sein Interesse an Geschichte hatte er schon früh entdeckt und somit war sein Studienfach klar. Er nimmt sein Studium ernst und investiert viel Zeit und Energie in dieses.

Finanziell wird er von seiner Mutter unterstützt, damit er neben dem Studium nicht arbeiten muss.

Privat reist er gerne an Ort von historischer Bedeutung: “wenn ich das nötige Kleingeld als Geburtstagsgeschenk bekomme”. Auch im Urlaub prüft er regelmässig seine E-Mails. Er recherchiert viel in der Universitätsbibliothek oder im Internet. Auf Internetrecherche geht er meistens von Zuhause aus.

Er benutzt einen älteren Windows Laptop: “Alte Kiste, tut’s aber noch für den Moment”. Von seinen Kollegen wird er als gebildet und streberisch beschrieben. Sich selber beschreibt er als ruhig und interessiert.

8.2.2 Test Aufgaben

Der Testperson wurden die folgenden zwei Aufgaben gestellt. Die Testperson sollte diese Aufgabe ausführen und ohne zusätzlichen Hilfsmitteln lösen. In der Aufgabe “Verbindung aufbauen” wurde der Testperson absichtlich ein falsches Passwort gegeben, sodass der Verbindungsaufbau nicht funktioniert und die Testperson überprüfen muss, woran das Fehlschlagen liegen könnte.

Aufgabe 1: Verbindung einrichten

Richten sie eine neue strongSwan VPN Verbindung ein. Über das Netzwerk-Icon in der Taskleiste kommen Sie zu den Verbindungsdetails. Die Daten für Ihre neue VPN Verbindung sind:

Option	Eingabe
Gateway	strongswan.org
Gateway Certificate	“cert.pem” zu finden in /home/TestPerson/Documents
Methode	EAP
Benutzername	Testperson
Passwort	1234

Bitte überprüfen Sie, ob die Verbindung angelegt wurde.

Aufgabe 2: Verbindung aufbauen

Bitte bauen sie eine VPN Verbindung auf. Starten Sie die Verbindung, die Sie in Aufgabe 1 eingerichtet haben. Bitte prüfen Sie, ob Sie verbunden sind. Falls der Verbindungsaufbau nicht funktionieren sollte, könnte es sein das Ihr Passwort nicht mehr gültig ist und daher vom Administrator zurückgesetzt wurde. Das Standard Passwort lautet: “bitteAendern”.

8.2.3 Testerkenntnisse

Aufgabe 1

Aufgabe 1 konnte bei allen GUI Versionen ohne Probleme durchgeführt werden. Die Testperson konnte die Verbindung ohne zu zögern einrichten. Das GUI wurde als positiv und übersichtlich eingeschätzt.

Aufgabe 2

Die Aufgabe 2 wurde in der GUI Version 1 als mühsam und unübersichtlich wahrgenommen. Die Testperson wusste nicht, wie sie vorgehen muss. Sie konnte sich jedoch nach Ändern des Passworts mit dem VPN Server verbinden.

Das GUI 2 kam wesentlich besser an, obwohl die Testperson sich mehrmals in dem Fenster umschauen musste, bevor sie verstand was genau das Problem und was zu tun war.

Mit dem finalen GUI war die Testperson sehr zufrieden. Sie konnte sich schnell über die Fehlerursache informieren und hatte danach das Passwort geändert, sowie die Verbindung erfolgreich aufgebaut.

9 Weiterentwicklung

In diesem Kapitel wird vorgestellt, wie sich Entwickler bei den Erweiterungen des NM Plasma Applets und dem strongSwan Plugin beteiligen können. Dazu werden die noch fehlenden Funktionalitäten des strongSwan Plugins und des NM Plasma Applets aufgelistet. Der aktuelle Stand des NM Plasma Applets kann über die Mailingliste vom NM Plasma Applets abgefragt werden. Zudem existiert ein System, in dem bekannte Probleme aufgelistet sind. Bevor jedoch Änderungen und Erweiterungen vorgenommen werden können, muss die Entwicklungsumgebung aufgesetzt werden. Die aktuelle Version des NM Plasma Applets ist über den in Abschnitt 7.1 beschriebenen Weg zugänglich. Einige Änderung, die ich vorgenommen habe, sind zum jetzigen Zeitpunkt noch nicht im svn, da sie noch von den KDE Entwicklern geprüft werden. Diese können vom Verzeichnis “02_Code/networkmanagement” der beigelegten DVD kopiert werden. Darin sind alle Entwicklungen meiner Arbeit abgelegt.

9.1 NM Plasma Applet

In den Grundlagen (Abschnitt 2.2.1) dieses Dokumentes sind die fehlenden Funktionalitäten des NM Plasma Applets aufgelistet. Während der Arbeit konnten alle fehlenden Funktionalitäten mit einer Ausnahme realisiert werden. Bei der fehlenden Funktionalität handelt es sich um die Anzeige der VPN Verbindungsdetails.

9.1.1 Verbindungsdetails

Die Verbindungsdetails wie IP, Remote IP, Gateway, DNS, usw. einer VPN Verbindung werden im NM Plasma Applet noch nicht angezeigt. Die Ursache dieser fehlende Anzeige ist auf die Unterscheidung zwischen Verbindung und Device zurückzuführen (Abschnitt 2.2.2). Abbildung 9.1 zeigt auf der linken Seite eine aktive Wireless Verbindung und deren Details. Auf der rechten Seite ist eine aktive VPN Verbindung mit fehlenden Details abgebildet.

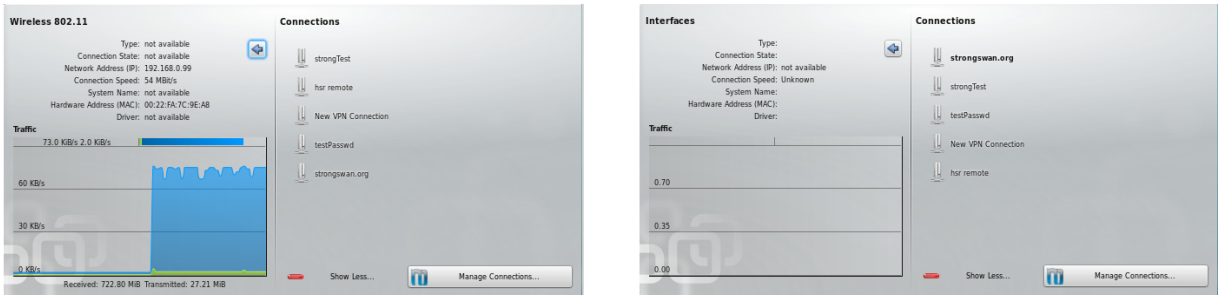


Abbildung 9.1: Fehlende VPN-Details

Ansonsten ist das NM Plasma Applet mit VPN Verbindungen einfach zu bedienen und verfügt über die nötigen Funktionalitäten. Es gibt jedoch noch Verbesserungsmöglichkeiten: Das Password Fenster wird nicht automatisch vor den Plasmoid Fenstern dargestellt. Somit muss das Passwort Widget von “Hand” in den Vordergrund geholt werden.

Das strongSwan Plugin im NM Plasma Applet verfügt über alle Funktionalitäten. Es fehlen jedoch noch die Übersetzungen in die verschiedenen Sprachen. Zurzeit sind lediglich die englische und deutsche Version umgesetzt. Die Übersetzung wird von KDE Entwicklern aus den jeweiligen Sprachgebieten vorgenommen.

9.1.2 Verbindungsabbruch

Das realisierte Fehlermeldungskonzept wurde ausführlich für den Verbindungsaufbau getestet und optimiert. Es müssen noch kleine Änderungen vorgenommen werden, damit die korrekten Fehler auch bei möglichen Abbrüchen einer aktiven Verbindung gemeldet werden. Dazu müssen im strongSwan Charon und dem NetworkManager neue Fehlermeldungen definiert werden. Sind diese neuen Fehlermeldungen festgelegt, würde während eines Abbruchs das Fehlermeldungsfenster angezeigt. Es müsste analysiert und getestet werden, ob dies von den Benutzern als sinnvoll empfunden wird. Es ist gut möglich, dass die Benutzer lieber durch eine ToolTip, Popup Nachricht über den Verbindungsabbruch informiert werden.

9.1.3 VPN Icon

Die Visualisierung des Verbindungsstatus muss noch verbessert werden. Konkret bedeutet dies, dass zum jetzigen Zeitpunkt schwer zu unterscheiden ist, ob eine VPN Verbindung einen Aufbau durchführt oder ob diese bereits eingerichtet ist. Die zwei unterschiedlichen Icons, geöffnetes Schloss und geschlossenes Schloss (Abbildung 4.4), sind schwer voneinander zu unterscheiden. Das geöffnete Schloss, welches beim Verbindungsaufbau angezeigt wird, könnte durch eine Animation erweitert werden. Dadurch würde symbolisiert, dass eine Verbindung noch nicht vollständig aufgebaut ist. Zudem wäre hilfreich, wenn die ToolTip kurz eingeblendet wird, sobald eine Verbindung aufgebaut oder beendet wurde.

9.2 Gnome Applet

Das Gnome Applet könnte nach Einführung des neuen Fehlermeldungskonzepts erweitert werden. Dazu müsste das Gnome Applet nur noch auf die “VpnStateChanged()” Signale des D-Buses reagieren und die entsprechenden Fehlermeldungen durch Userinterface-Elemente anzeigen. Ein solches Fehlermeldungsfenster müsste im Gnome Applet neu entworfen werden.

9.3 KDE Bugs/Mailinglists/Reviewboard

Falls Änderungen an dem NM Plasma Applet gemacht werden sollten, ist es wichtig, dass potenzielle Entwickler die KDE Entwickler kontaktieren. Es könnte sein, dass die KDE Entwickler

zeitgleich genau an diesen Erweiterungen arbeiten. Deshalb sollte sich jeder Entwickler in die Mailingliste eintragen und seine Vorhaben beschreiben. Zudem existiert eine Bug Tracking System, in dem bekannte Fehler eingetragen werden können. Entwickler können einen solchen “Bug” (Fehler) analysieren und einen Patch[1] schreiben, der diesen Fehler berichtigt. Ein Patch wird am besten im svn erstellt. Ändert man den Sourcecode, kann über den Befehl *svn diff >patch.diff* ein Patch kreiert werden. Dieser Patch kann nun aufs Rewievboard hochgeladen werden. Die KDE Entwickler werden diesen Patch prüfen und eventuell in das KDE System integrieren.

NM Plasma Applet Mailingliste:
https://mail.kde.org/mailman/listinfo/kde-networkmanager

Bug Tracking System:
https://bugs.kde.org

Rewievboard:
http://reviewboard.kde.org

[1] Ein Patch ist ein überarbeiteter Quellcode, der einen Fehler berichtigt

10 Projektmanagement

10.1 Projektplan

In der zweiten Projektwoche wurde der Projektplan (Abbildung 10.1) entworfen. Es wurde bewusst eine lange Einarbeitungszeit von 4 Wochen gewählt, da eine umfassende Einarbeitung wesentlich den Erfolg und die Qualität des Projektes beeinflusst. Es waren zwei Versionen der Software geplant, die jeweils in einer Zeitspanne von zwei Wochen getestet und verbessert werden sollten. Vier Meilensteine wurden definiert:

Beta Version 0.5 :

- Verbindungskonfiguration strongSwan (EAP)
- Verbindung starten
- Verbindung stoppen

Version 1 :

- Beta Version 0.5 verbessert
- Beta Version 0.5 getestet

Beta Version 1.5 :

- Verbindungsaufbau mit dynamischer Passwortabfrage
- VPN Verbindungsstatus visualisieren
- Fehlermeldungsanzeige
- Verbindungskonfiguration mit Private Key und Zertifikat

Version 1 :

- Beta Version 1.5 verbessert
- Beta Version 1.5 getestet

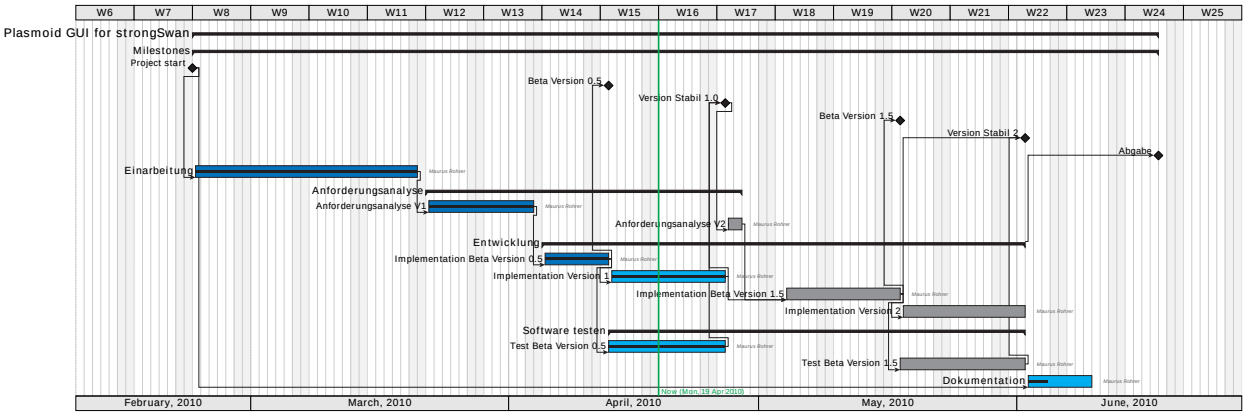


Abbildung 10.1: Ursprünglicher Projektplan

Nach der Einarbeitung und dem Erreichen des Meilensteins “Version 1”, wurde der Projektplan überarbeitet. Der Meilenstein “Version 1” wurde gut zwei Wochen früher erreicht. Somit blieb mehr Zeit, um die weiteren Funktionalitäten zu realisieren. Im ersten Entwurf des Projektplans sollten alle weiteren Funktionalitäten in einer Version, sprich zeitgleich entwickelt werden. Das Risiko, dass eine dieser Funktionalitäten nicht wie geplant entwickelt werden könnte und somit die ganze Version gefährdet hätte, war zu gross. Somit wurden mehrere kleinere Versionen geplant, die jeweils nur eine Funktionalität beinhalten. In Abbildung 10.2 ist der durchgeführte Projektplan abgebildet. In den letzten zwei Wochen der Arbeit wurde die Dokumentation vervollständigt, welche während der ganzen Dauer des Projektes geführt wurde. Im neuen Projektplan wurden neue Meilensteine definiert:

Version 1 -

- Verbindungskonfiguration strongSwan (EAP)
- Verbindung starten
- Verbindung stoppen

Version 1.2 -

- dynamische Passwortabfrage

Version 1.6 -

- Fehlermeldungskonzept

Version 2 -

- VPN Icon Anpassung
- Verbindungskonfiguration mit Private Key und Zertifikat

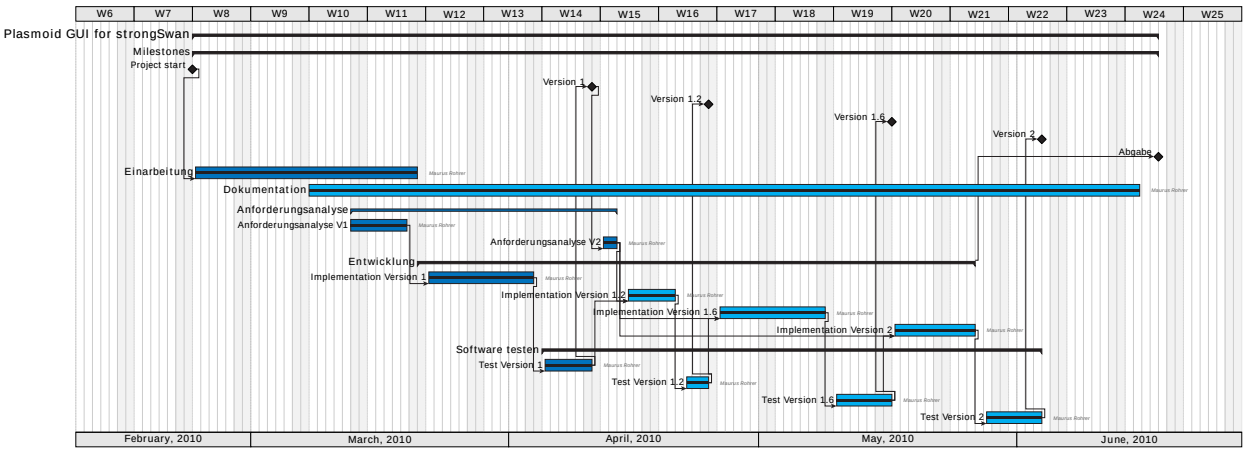


Abbildung 10.2: Überarbeiteter Projektplan

Die Projektpläne sind auf der mitgelieferten DVD im Verzeichnis “01_Dokumentation/Projektplan/“ abgelegt.

10.2 Zeitabrechnung

In Tabelle 10.1 ist die Zeitabrechnung aufgelistet. Die Zeit wurde für die unterschiedlichen Phasen im Projektplan 10.2 erfasst. Die geplante Zeit, ist die offizielle welche für eine Bachelorarbeit (12 ETCS x 30 Stunden) geleistet werden muss.

Projektphase	geplant [Std.]	realisiert [Std.]
Einarbeitung	80	90
Anforderungsanalyse	28	20
Entwicklung	132	150
Tests	60	50
Dokumentation	60	90
Total:	**360**	**400**

Tabelle 10.1: Zeitabrechnung

10.3 Qualitätssicherung

Damit die Qualität des Projektes garantiert werden konnte, wurden diverse Massnahmen ergriffen. Die Tests, eine wichtige Massnahme der Qualitätssicherung, sind in Abschnitt 8 detailliert beschrieben.

10.3.1 Dokumentation

Während der ganzen Projektdauer wurde die Dokumentation mitgeführt. Das bedeutet, das gleichzeitig mit der Anforderungsanalyse auch die Dokumentation zu diesem Teil entstanden ist. In den letzten zwei Wochen der Arbeitszeit wurde die Dokumentation vervollständigt und überarbeitet. Über die Form und den Inhalt der Dokumentation wurde regelmässig mit dem Betreuer diskutiert.

10.3.2 Sitzungen

Es wurden wöchentlich Sitzungen durchgeführt. Diese Sitzungen waren wichtig und hilfreich. Es konnten Probleme angesprochen und grundlegende Entscheide getroffen werden. Da ich die Arbeit alleine erstellte, waren die Sitzungen für mich von grosser Bedeutung. Bei einer Einzelarbeit fehlt der Austausch mit den Kommilitone über Inhalte und Probleme. Dieser Austausch fand hauptsächlich mit dem Betreuer und den Mitarbeitern des Institutes statt.

10.3.3 Codereviews

Der entwickelte Code wurde immer in den Testphasen intensiv gereviewt. Zudem wurde der Code von den KDE Entwicklern gereviewt. Die KDE Entwickler sind erfahrene Programmierer und nach ihrer Abnahme ist die Qualität des Codes garantiert. Einige Code Teile mussten nach den Reviews der KDE Entwickler überarbeitet werden.

10.3.4 Programmierrichtlinien

Das KDE Projekt hat klare Programmierrichtlinien. Diese müssen eingehalten werden, damit der geschrieben Quellcode von den KDE Entwicklern geprüft und integriert wird. [eK10b]

10.3.5 Risikomanagement

In der Tabelle 10.3 sind die relevanten Risiken aufgelistet. Die Risiken wurden in verschiedene Stufen eingeteilt. Die Stufen sind nach ihrer Aufttritswahrscheinlichkeit eingeteilt.

Risiko Stufe	Risiko Einschätzung
1	hohes Risiko
2	mittleres Risiko
3	geringes Risiko

Tabelle 10.2: Risiko Stufen

Folgende Arbeiten/Risiken wurden durch das Risikomanagement überwacht:

Risiko	Beschreibung	Stufe	Zusatzaufwand in Tage
Einarbeitung	die vorhanden Komponenten können nicht in der geplanten Zeit analysiert werden	1	5
Plugin	strongSwan Daemon kann nicht in das NM Plasma Applet integriert werden	1	5
Plasma Zukunft	das NM Plasma Applet wird nicht in die KDE Umgebung integriert	2	10
Plugin Integration	der entwickelte Quellcode wird nicht in KDE Versionsverwaltung geladen	2	10
Architekturwechsel	die Architektur des NM Plasma Applets wird Grundlegend überarbeitet, sodass das strongSwan Plugin nicht mehr funktioniert	3	10
Teamwork	durch die fehlende Teamarbeit befasst sich der Entwickler zu lange mit einem Problem	3	3
Hardware	durch Hardware Ausfall (Laptop) muss die Entwicklungsumgebung neu aufgesetzt werden	3	1
Personalausfall	der Entwickler wird durch Krankheit oder Unfall arbeitsunfähig	3	5

Tabelle 10.3: Risikomanagement

Teil III

Appendix

A Persönlicher Bericht

Spannende und herausfordernde Wochen liegen hinter mir, in denen ich eine sehr interessante Arbeit durchführen konnte. Eine Woche vor Beginn der Bachelorarbeit war ich noch in Berlin (ERASMUS Austausch) um dort die letzten Prüfungen zu schreiben. Ich bin froh einen guten Start in die Bachelorarbeit gefunden zu haben, trotz Umzug zurück in die Schweiz.
Die Entwicklung in der Linux “Welt” hat mich fasziniert. Als grosser Open Source Unterstützer und jahrelanger Linuxanwender war es mir eine Ehre für die Open Source Projektgruppen strongSwan und KDE einen Entwicklungsbeitrag leisten zu können.
Ich hatte grossen Respekt vor der Entwicklungsumgebung KDE da mir gewusst war, dass viel Quellcode vorhanden ist, den es zu verstehen gilt. Die ersten Wochen waren deshalb intensiv und teilweise ernüchternd. Ich musste die Entwicklungsumgebung erstmals aufsetzten und es dauerte lange, bis ich meine ersten Entwicklungen auf dem Bildschirm “bestaunen” konnte.
Da die KDE Entwicklungsumgebung schlecht dokumentiert ist, musste ich tagelang Quellcode lesen und daraus auf die Architektur schliessen. Wie leider oft in solchen Arbeiten, ist die Zeit gegen Ende des Projekts am produktivsten. Jedoch war es dann wichtig, mich in der verbleibenden Zeit auf die Dokumentation und den Abschluss zu konzentrieren. Sicherlich werde ich weitere Entwicklung in der KDE und strongSwan Umgebung durchführen.
Ich hatte vor dieser Arbeit noch keine Erfahrungen in der Linux Entwicklung. Jedoch haben mir meine Kenntnisse von QT und C++ geholfen, den umfangreichen Code schnell zu verstehen.
Die gute Zusammenarbeiten mit Prof. Dr. Andreas Steffen, Martin Willi und Tobias Brunner war sehr bereichernd und motivierend. Ihr Einsatz und ihr Wissen haben mich beeindruckt. Ich möchte mich an dieser Stelle ganz herzlich bei ihnen bedanken.

B NetworkManager D-Bus Interface Specification

Version 0.8

Interfaces

- `org.freedesktop.NetworkManager`
- `org.freedesktop.NetworkManager.AccessPoint`
- `org.freedesktop.NetworkManager.Connection.Active`
- `org.freedesktop.NetworkManager.VPN.Connection`
- `org.freedesktop.NetworkManager.VPN.Plugin`

org.freedesktop.NetworkManager.VPN.Connection

Represents an active connection to a Virtual Private Network.
Interface has no methods.

Signals:

PropertiesChanged (a{sv}: properties)

Parameters

`properties` - `a{sv}` **(String_Variant_Map)** A dictionary mapping property names to variant boxed values

VpnStateChanged (u: state, u: reason)

Emitted when the state of the VPN connection has changed.

Parameters

`state` - `u` **(NM_VPN_CONNECTION_STATE)** The new state of the VPN connection.

`reason` - `u` **(NM_VPN_CONNECTION_STATE_REASON)** Reason code describing the change to the new state.

Properties:

`VpnState` - `u` - `(read)` **(NM_VPN_CONNECTION_STATE)** The VPN-specific state of the connection.

`Banner` - `s` - `(read)` The banner string of the VPN connection.

Enumerated types:

NM_VPN_CONNECTION_STATE

`NM_VPN_CONNECTION_STATE_UNKNOWN = 0` The state of the VPN connection is unknown.

`NM_VPN_CONNECTION_STATE_PREPARE = 1` The VPN connection is preparing to connect.

`NM_VPN_CONNECTION_STATE_NEED_AUTH = 2` The VPN connection needs authorization credentials.

`NM_VPN_CONNECTION_STATE_CONNECT = 3` The VPN connection is being established. FIXME: Should be CONNECTING or CONNECTED.

`NM_VPN_CONNECTION_STATE_IP_CONFIG_GET = 4` The VPN connection is getting an IP address. FIXME: Should be an -ING

`NM_VPN_CONNECTION_STATE_ACTIVATED = 5` The VPN connection is active.

`NM_VPN_CONNECTION_STATE_FAILED = 6` The VPN connection failed.

`NM_VPN_CONNECTION_STATE_DISCONNECTED = 7` The VPN connection is disconnected.

NM_VPN_CONNECTION_STATE_REASON

`NM_VPN_CONNECTION_STATE_REASON_UNKNOWN = 0` The reason for the VPN connection state change is unknown.

`NM_VPN_CONNECTION_STATE_REASON_NONE = 1` No reason was given for the VPN connection state change.

`NM_VPN_CONNECTION_STATE_REASON_USER_DISCONNECTED = 2` The VPN connection changed state because the user disconnected it.

`NM_VPN_CONNECTION_STATE_REASON_DEVICE_DISCONNECTED = 3` The VPN connection changed state because the device it was using was disconnected.

`NM_VPN_CONNECTION_STATE_REASON_SERVICE_STOPPED = 4` The service providing the VPN connection was stopped.

`NM_VPN_CONNECTION_STATE_REASON_IP_CONFIG_INVALID = 5` The IP config of the VPN connection was invalid.

`NM_VPN_CONNECTION_STATE_REASON_CONNECT_TIMEOUT = 6` The connection attempt to the VPN service timed out.

`NM_VPN_CONNECTION_STATE_REASON_SERVICE_START_TIMEOUT = 7` A timeout occurred while starting the service providing the VPN connection.

`NM_VPN_CONNECTION_STATE_REASON_SERVICE_START_FAILED = 8` Starting the service starting the service providing the VPN connection failed.

`NM_VPN_CONNECTION_STATE_REASON_NO_SECRETS = 9` Necessary secrets for the VPN connection were not provided.

org.freedesktop.NetworkManager.VPN.Plugin

This interface is provided by plugins providing VPN services to the NetworkManager daemon.

Methods:

Connect (a{sa{sv}}: connection) – nothing

Tells the plugin to connect.

Parameters

`connection` - `a{sa{sv}}` **(String_String_Variant_Map_Map)** Describes the connection to be established.

Possible errors

`org.freedesktop.NetworkManager.VPN.Error.StartingInProgress` The request could not be processed because the VPN connection is already being started.*(generic description)*

`org.freedesktop.NetworkManager.VPN.Error.AlreadyStarted` The request could not be processed because a VPN connection was already active.*(generic description)*

`org.freedesktop.NetworkManager.VPN.Error.StoppingInProgress` The request could not be processed because the VPN connection is already being stopped.*(generic description)*

`org.freedesktop.NetworkManager.VPN.Error.BadArguments` Invalid arguments were passed with the request. FIXME: too general.*(generic description)*

`org.freedesktop.NetworkManager.VPN.Error.LaunchFailed` A binary providing the service failed to launch.*(generic description)*

NeedSecrets (a{sa{sv}}: settings) – s

Asks the plugin whether the provided connection will require secrets to connect successfully.

Parameters

`settings` - `a{sa{sv}}` **(String_String_Variant_Map_Map)** Describes the connection that may need secrets.

Returns

`setting_name` - `s` The setting name within the provided connection that requires secrets, if any.

Possible errors

`org.freedesktop.NetworkManager.VPN.Error.ConnectionInvalid` The request could not be processed because the VPN connection settings were invalid.*(generic description)*

Disconnect () – nothing

Disconnect the plugin.

Possible errors

`org.freedesktop.NetworkManager.VPN.Error.StoppingInProgress` The request could not be processed because the VPN connection is already being stopped.*(generic description)*

`org.freedesktop.NetworkManager.VPN.Error.AlreadyStopped` The request could not be processed because the VPN connection was already stopped.*(generic description)*

SetIp4Config (a{sv}: config) – nothing

Set IPv4 details on the connection.

Parameters

`config` - `a{sv}` **(String_Variant_Map)** Ip4Config details for the conneciton.

SetFailure (s: reason) – nothing

Indicate a failure to the plugin.

Parameters

`reason` - `s` The reason for the failure.

Signals:

StateChanged (u: state)

Emitted when the plugin state changes.

Parameters

`state` - `u` **(NM_VPN_CONNECTION_STATE)** The new state of the plugin.

Ip4Config (a{sv}: ip4config)

The plugin obtained an IPv4 configuration.

Parameters

`ip4config` - `a{sv}` **(String_Variant_Map)** The IPv4 configuration.

LoginBanner (s: banner)

Emitted when the plugin receives a login banner from the VPN service.

Parameters

`banner` - `s` The login banner string.

Failure (u: reason)

Emitted when a failure in the VPN plugin occurs.

Parameters

`reason` - `u` **(NM_VPN_PLUGIN_FAILURE)** Reason code for the failure.

Properties:

`State` - `u` - `(read)` **(NM_VPN_CONNECTION_STATE)** The state of the plugin.

Enumerated types:

NM_VPN_PLUGIN_FAILURE

`NM_VPN_PLUGIN_FAILURE_LOGIN_FAILED = 0` Login failed.

`NM_VPN_PLUGIN_FAILURE_CONNECT_FAILED = 1` Connect failed.

`NM_VPN_PLUGIN_FAILURE_BAD_IP_CONFIG = 2` Invalid IP configuration returned from the VPN plugin.

Generic types:

Enumerated types:

NM_802_11_MODE

`NM_802_11_MODE_UNKNOWN = 0` Mode is unknown.

`NM_802_11_MODE_ADHOC = 1` Uncoordinated network without central infrastructure.

`NM_802_11_MODE_INFRA = 2` Coordinated network with one or more central controllers.

Mapping types:

String_Variant_Map - a{ s: Key – v: Value }

A mapping from strings to variants representing extra key-value pairs.

Members

`Key` - `s` *(undocumented)*

`Value` - `v` *(undocumented)*

String_String_Map - a{ s: Key – s: Value }

A mapping from strings to strings representing extra key-value pairs.

Members

`Key` - `s` *(undocumented)*

`Value` - `s` *(undocumented)*

String_String_Variant_Map_Map - a{ s: Key – a{sv}: Value }

A mapping from strings to a map of string to variant.

Members

`Key` - `s` *(undocumented)*

`Value` - `a{sv}` **(String_Variant_Map)** *(undocumented)*

Errors:

org.freedesktop.NetworkManager.Error.UnknownConnection

Connection was not provided by any known settings service.

org.freedesktop.NetworkManager.Error.UnknownDevice

Unknown device.

org.freedesktop.NetworkManager.Error.InvalidService

Invalid settings service (not a recognized system or user settings service name).

org.freedesktop.NetworkManager.Error.SystemConnection

Connection was superseded by a system connection.

org.freedesktop.NetworkManager.Error.PermissionDenied

User does not have the permission to activate this connection.

Errors:

org.freedesktop.NetworkManager.VPN.Error.General

This is a drab, nondescript error.

org.freedesktop.NetworkManager.VPN.Error.StartingInProgress

The request could not be processed because the VPN connection is already being started.

org.freedesktop.NetworkManager.VPN.Error.AlreadyStarted

The request could not be processed because a VPN connection was already active.

org.freedesktop.NetworkManager.VPN.Error.StoppingInProgress

The request could not be processed because the VPN connection is already being stopped.

org.freedesktop.NetworkManager.VPN.Error.AlreadyStopped

The request could not be processed because the VPN connection was already stopped.

org.freedesktop.NetworkManager.VPN.Error.WrongState

The request could not be processed because the VPN connection is in the wrong state for this type of request. FIXME: too general?

org.freedesktop.NetworkManager.VPN.Error.BadArguments

Invalid arguments were passed with the request. FIXME: too general.

org.freedesktop.NetworkManager.VPN.Error.LaunchFailed

A binary providing the service failed to launch.

org.freedesktop.NetworkManager.VPN.Error.ConnectionInvalid

The request could not be processed because the VPN connection settings were invalid.

C IKEv2 (RFC 4306)

Notify Message Types

Notification information can be error messages specifying why an SA could not be established. It can also be status data that a process managing an SA database wishes to communicate with a peer process. The table below lists the Notification messages and their corresponding values. The number of different error statuses was greatly reduced from IKEv1 both for simplification and to avoid giving configuration information to probers.

Types in the range 0 - 16383 are intended for reporting errors. An implementation receiving a Notify payload with one of these types that it does not recognize in a response MUST assume that the corresponding request has failed entirely. Unrecognized error types in a request and status types in a request or response MUST be ignored except that they SHOULD be logged.

Notify payloads with status types MAY be added to any message and MUST be ignored if not recognized. They are intended to indicate capabilities, and as part of SA negotiation are used to negotiate non-cryptographic parameters.

NOTIFY MESSAGES - ERROR TYPES — Value

RESERVED — 0

UNSUPPORTED_CRITICAL_PAYLOAD — 1

Sent if the payload has the "critical" bit set and the payload type is not recognized. Notification Data contains the one-octet payload type.

INVALID_IKE_SPI — 4

Indicates an IKE message was received with an unrecognized destination SPI. This usually indicates that the recipient has rebooted and forgotten the existence of an IKE_SA.

INVALID_MAJOR_VERSION — 5

Indicates the recipient cannot handle the version of IKE specified in the header. The closest version number that the recipient can support will be in the reply header.

INVALID_SYNTAX 7

Indicates the IKE message that was received was invalid because some type, length, or value was out of range or because the request was rejected for policy reasons. To avoid a denial of service attack using forged messages, this status may only be returned for and in an encrypted packet if the message ID and cryptographic checksum were valid. To avoid leaking information to someone probing a node, this status MUST be sent in response to any error not covered by one of the other status types. To aid debugging, more detailed error information SHOULD be written to a console or log.

INVALID_MESSAGE_ID 9

Sent when an IKE message ID outside the supported window is received. This Notify MUST NOT be sent in a response; the invalid request MUST NOT be acknowledged. Instead, inform the other side by initiating an INFORMATIONAL exchange with Notification data containing the four octet invalid message ID. Sending this notification is optional, and notifications of this type MUST be rate limited.

INVALID_SPI 11

MAY be sent in an IKE INFORMATIONAL exchange when a node receives an ESP or AH packet with an invalid SPI. The Notification Data contains the SPI of the invalid packet. This usually indicates a node has rebooted and forgotten an SA. If this Informational Message is sent outside the context of an IKE_SA, it should be used by the recipient only as a "hint" that something might be wrong (because it could easily be forged).

NO_PROPOSAL_CHOSEN 14

None of the proposed crypto suites was acceptable.

INVALID_KE_PAYLOAD 17

The D-H Group # field in the KE payload is not the group # selected by the responder for this exchange. There are two octets of data associated with this notification: the accepted D-H Group # in big endian order.

AUTHENTICATION_FAILED 24

Sent in the response to an IKE_AUTH message when for some reason the authentication failed. There is no associated

data.

SINGLE_PAIR_REQUIRED 34

This error indicates that a CREATE_CHILD_SA request is unacceptable because its sender is only willing to accept traffic selectors specifying a single pair of addresses. The requestor is expected to respond by requesting an SA for only the specific traffic it is trying to forward.

NO_ADDITIONAL_SAS 35

This error indicates that a CREATE_CHILD_SA request is unacceptable because the responder is unwilling to accept any more CHILD_SAs on this IKE_SA. Some minimal implementations may only accept a single CHILD_SA setup in the context of an initial IKE exchange and reject any subsequent attempts to add more.

INTERNAL_ADDRESS_FAILURE 36

Indicates an error assigning an internal address (i.e., INTERNAL_IP4_ADDRESS or INTERNAL_IP6_ADDRESS) during the processing of a Configuration Payload by a responder. If this error is generated within an IKE_AUTH exchange, no CHILD_SA will be created.

FAILED_CP_REQUIRED 37

Sent by responder in the case where CP(CFG_REQUEST) was expected but not received, and so is a conflict with locally configured policy. There is no associated data.

TS_UNACCEPTABLE 38

Indicates that none of the addresses/protocols/ports in the supplied traffic selectors is acceptable.

INVALID_SELECTORS 39

MAY be sent in an IKE INFORMATIONAL exchange when a node receives an ESP or AH packet whose selectors do not match those of the SA on which it was delivered (and that caused the packet to be dropped). The Notification Data contains the start of the offending packet (as in ICMP messages) and the SPI field of the notification is set to match the SPI of the IPsec SA.

RESERVED TO IANA - Error types 40 - 8191

Private Use - Errors 8192 - 16383

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BR06] BRUNNER, TOBIAS und DANIEL RÖTHLISBERGER: *NAT-Traversal for strongSwan II*. Technischer Bericht, 2006.

[eK10a] KDE E.V.: *The KDE Source Repository - WebSVN*. `http://websvn.kde.org/`, 2010.

[eK10b] KDE E.V.: *KDE TechBase - Kdelibs Coding Style*. `http://techbase.kde.org/Policies/Kdelibs_Coding_Style`, 2010.

[eK10c] KDE E.V.: *KDE TechBase - Set up KDE 4 for development*. `http://techbase.kde.org/Getting_Started/Set_up_KDE_4_for_development`, 2010.

[eK10d] KDE E.V.: *KDE TechBase - User Interface Guidelines*. `http://techbase.kde.org/Projects/Usability/HIG`, 2010.

[eK10e] KDE E.V.: *KDE UserBase - Plasma*. `http://userbase.kde.org/Plasma`, 2010.

[fre10] FREEDESKTOP.ORG: *D-Bus specification*. `http://dbus.freedesktop.org/doc/dbus-specification.html`, 2010.

[HM05] HUTTER, JAN und WILLI MARTIN: *strongSwan II, Eine IKEv2-Implementierung für Linux.* Diplomarbeit, Hochschule für Technik Rapperswil, 2005.

[JM08] JASMIN, BLANCHETTE und SUMMERFIELD MARK: *C++ GUI Programming with QT 4*. Trolltech, Second Edition Auflage, 2008.

[Kau05] KAUFMAN, C.: *RFC4306 - Internet Key Exchange (IKEv2) Protocol*. `http://tools.ietf.org/html/rfc4306`, Dezember 2005.

[Küg10] KÜGLER, SEBASTIAN: *Vizzion Blog - Network Management*. `http://vizzzion.org/blog/2010/02/tokamak-iv-network-management/`, 2010.

[Red09] REDHAT: *NetworkManager - Architecture*. `http://people.redhat.com/dcbw/NetworkManager/architecture.html`, 2009.

[Tro10] TROLLTECH: *QT Designer Manual*. `http://doc.trolltech.com/4.6/designer-manual.html`, 2010.