
Entwurf und Implementierung eines Plug-ins für JCrypTool

S/MIME-Visualisierung

Bachelorarbeit von Rouven Röhrig

E-Mail: rroehrig(a-t)rbg.informatik.tu-darmstadt.de



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Fachgebiet Theoretische Informatik –
Kryptographie und Computeralgebra

Prof. Dr. Johannes Buchmann

An dieser Stelle möchte ich meinen Unterstützerinnen und Unterstützern danken.

Ein besonderer Dank gilt Evangelos Karatsiolis, der für jede Frage offen war und mich immer motivieren konnte.

Ein herzlicher Dank geht an meine Freundin Nadine, die Korrektur gelesen hat und mir stets zur Seite stand.

Außerdem bedanke ich mich bei meiner Familie und Korrekturleserinnen Angela, Halina und Annika.

Darüber hinaus bedanke ich mich bei Prof. Dr. Johannes Buchmann, durch den diese Arbeit ermöglicht wurde.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Rouven Röhrig. Darmstadt, am 19.03.2010.

Inhaltsverzeichnis

1	Einleitung	1
2	Abhängigkeiten	2
2.1	JCrypTool	2
2.2	Eclipse Rich Client Platform	2
2.3	Standard Widget Toolkit (SWT)	3
2.4	JFace	3
3	Theoretische Betrachtung	5
3.1	E-Learning	5
3.1.1	Definition von E-Learning	5
3.1.2	Multimedialität	5
3.1.3	Multicodalität	6
3.1.4	Multimodalität	6
3.1.5	Interaktivität	7
3.2	Benutzbarkeit (<i>Usability</i>)	8
3.2.1	Erlernbarkeit	8
3.2.2	Flexibilität	9
3.2.3	Robustheit	10
3.3	Internet-Standards	11
3.3.1	RFC 822	11
3.3.2	Multipurpose Internet Mail Extensions (MIME)	11
3.3.3	Secure MIME (S/MIME)	12
3.3.4	Cryptographic Message Syntax CMS	13
4	Anforderungsanalyse	14
4.1	Welche Konzepte sollen erklärt werden?	14
4.2	Zielgruppe	14
4.2.1	Analyse der Zielgruppen	14
4.2.2	Definition der Zielgruppen	14
4.2.3	Besondere Anforderungen der Zielgruppen	15
4.3	Globalisierung	15
4.3.1	Internationalisierung	15
4.3.2	Lokalisierung	16
4.4	Akteure im System	17
4.5	Feature-Liste	17
4.6	Anwendungsfälle (Use Cases)	17
5	Design	22
5.1	Senderansicht	22
5.1.1	E-Mail-Editor	22
5.1.2	MIME-Nachricht-Ansicht	23
5.1.3	S/MIME-Nachricht-Ansicht	23
5.2	Dynamische Zertifikat-Erzeugung	23

5.3	Empfängeransicht	24
5.3.1	S/MIME-Ansicht	24
5.3.2	E-Mail-Viewer	25
5.4	Erklärungen	25
5.4.1	Ziele	26
5.4.2	Design	26
5.5	E-Mail-Client-Konfiguration	27
6	Umsetzung	28
6.1	Integration in JCrypTool	28
6.2	Senderansicht	28
6.3	Empfängeransicht	29
6.4	Konfiguration des E-Mail-Clients	30
6.5	Erklärungsdialoge	30
7	Fazit und Ausblick	34
8	Literaturverzeichnis	35

Abbildungsverzeichnis

2.1	JCrypTool: Einstellungsdialog unter Mac OS X (10.6).	4
5.1	Erster Entwurf der Senderansicht.	22
5.2	Aktivitätsdiagramm: Senderansicht.	24
5.3	Aktivitätsdiagramm: Senden.	24
5.4	Erster Entwurf der Empfängeransicht.	25
5.5	Aktivitätsdiagramm: Empfängeransicht.	25
5.6	Dialogentwurf mit verschiedenen Anordnungen.	27
6.1	Visualisierungsmenü mit integriertem S/MIME-Plug-in.	28
6.2	Grafische Implementierung der Senderansicht (Mac OS X 10.6).	29
6.3	S/MIME-Modus-Menü (Mac OS X 10.6).	29
6.4	Grafische Implementierung der Empfängeransicht (Mac OS X 10.6).	30
6.5	Einstellungsdialog mit S/MIME-Einstellungsseite (Mac OS X 10.6).	31
6.6	Wizard mit der Erklärung für den MIME-Header (Mac OS X 10.6).	32
6.7	Wizard mit der Erklärung zu SignedData vom CMS (Mac OS X 10.6).	33

Tabellenverzeichnis

4.1	Feature-Liste.	18
4.2	Use Case 1 – Einfache E-Mail erstellen.	19
4.3	Use Case 2 – S/MIME-Nachricht mit Signatur erzeugen.	19
4.4	Use Case 3 – S/MIME-Nachricht mit Verschlüsselung erstellen.	19
4.5	Use Case 4 – S/MIME-Nachricht als E-Mail versenden.	20
4.6	Use Case 5 – Senden.	20
4.7	Use Case 6 – Signatur überprüfen.	21
4.8	Use Case 7 – S/MIME-Nachricht entschlüsseln.	21

1 Einleitung

Jeden Tag werden weltweit in Unternehmen, aber auch zunehmend im privaten Bereich E-Mails verschlüsselt und signiert. Die typische Benutzerschnittstelle beschränkt sich meistens auf nicht viel mehr als Empfänger, Betreff und einem Textfeld, sowie Knöpfe o.ä. zum Signieren und Verschlüsseln der Nachricht. Dass hinter diesen Eingabefeldern Internetstandards, wie *Multipurpose Internet Mail Extensions* (MIME) zum Darstellen der E-Mail und *Secure/Multipurpose Internet Mail Extensions* (S/MIME) zum Verschlüsseln der E-Mail stecken, bleibt dem Nutzer verborgen. Laut einer PGP Corporation-Studie von 2009 nutzen 26 Prozent der führenden deutschen IT-Unternehmen „meistens“ E-Mail-Verschlüsselung¹. In diesen Unternehmen sitzen häufig viele nicht-technische Mitarbeiter, für die E-Mail-Verschlüsselungen äußerst wichtig sind; beispielsweise in den Rechts- und Personalabteilungen.

Was wäre daher angebrachter, als die Techniken dieses Alltagsszenarios durch eine Visualisierung möglichst vielen Menschen näher zu bringen? Nutzer sollten nicht nur auf die oben beschriebene Nutzerschnittstelle blicken können, sondern den Blick hinter die „Kulisse“ einer (sicheren) E-Mail werfen dürfen. Dabei muss die Benutzerfreundlichkeit an allererster Stelle stehen, insbesondere durch eine einfache Anwendung sowie einer intuitiven Nutzung.

Diese Bachelorarbeit beschäftigt sich mit dem Entwurf und der Implementierung eines S/MIME-Visualisierungsplug-ins für JCrypTool². JCrypTool ist eine kryptographische E-Learning-Plattform, die zum einen eine graphische Oberfläche für kryptographische Algorithmen zur Verfügung stellt und zum anderen verschiedene Algorithmen visualisiert. Im Fokus einer Visualisierung stehen stets das Verständnis des Benutzers. Daher ist ein Schwerpunkt dieser Arbeit konsequent Praktiken der Benutzerfreundlichkeit in die Planung mit einzubeziehen. Der Wiedererkennungsfaktor spielt dabei eine große Rolle. Daher wird die natürliche Anwendungsumgebung, die typische Schnittstelle jedes E-Mail-Client-Programms, mit in das Plug-in integriert. Viele Techniken erfordern zwar kryptographisches Grundverständnis, doch gerade deshalb setzt sich das Plug-in zum Ziel, so detailliert wie möglich und so abstrakt wie nötig zu bleiben. In diesem Zusammenhang ist ein schrittweises Vorgehen von der typischen E-Mail-Schnittstelle über Internetstandards wie *RFC 822* und *MIME* bis hin zur Verschlüsselung und Signaturerzeugung, bzw. Entschlüsselung und Signaturprüfung mittels *S/MIME* unverzichtbar.

Nach einer Durchführung des Plug-in soll der Benutzer folgende Lernziele erreicht haben: Sehr gutes Verständnis von *S/MIME*, Basisverständnis von *RFC 822*, *MIME* und *CMS*, sowie die genannten Konzepte im Kontext von sicherer E-Mail-Kommunikation erfassen können. Diese Arbeit und das dazugehörige Plug-in behandeln nicht die kryptographischen Grundlagen der verwendeten Algorithmen und abstrahieren von der Notwendigkeit einer Public-Key-Infrastruktur. Dies ist sinnvoll, da zum einen die beachteten Konzepte bereits sehr komplex sind und zu tiefe und zu komplizierte Einblicke der Akzeptanz schaden könnten. Zum anderen existieren bereits darauf konzipierte Vorarbeiten, beispielsweise die Visualisierung von RSA, DSA und ElGamal. Zudem liegt der Fokus dieser Arbeit sehr stark auf den Abläufen, die sich auf der Seite des Benutzers abspielen, d.h. E-Mail schreiben, Signatur/Verschlüsselung erstellen und Versenden der E-Mail, bzw. E-Mails empfangen und Signatur prüfen/E-Mail entschlüsseln.

¹ PGP Corporation, Jahresstudie 2009, Verschlüsselungstrends in deutschen Unternehmen, <http://www.encryptionreports.com/>, letzter Zugriff: 10.03.2010

² <http://jcryptool.sourceforge.net/>, letzter Zugriff: 10.03.2010

2 Abhängigkeiten

Das zu entwerfende Plug-in baut auf existierenden Technologien, Frameworks und APIs auf. In diesem Kapitel werden die wichtigsten Abhängigkeiten dieses Plug-ins beschrieben.

2.1 JCrypTool

JCrypTool ist eine kryptographische E-Learning-Plattform, die auf der *Eclipse Rich Client Platform* (Abschnitt 2.2) basiert. JCrypTool bietet eine plattform-unabhängige graphische Oberfläche, die zum einen kryptographische Algorithmen der Krypto-Bibliotheken FlexiProvider¹ und Bouncy Castle² zur Verfügung stellt, und zum anderen verschiedene Algorithmen visualisiert.

JCrypTool hat sich aus dem E-Learning-Programm *CrypTool*³ für Windows entwickelt, das ursprünglich für Schulungen in Unternehmen entwickelt wurde⁴. Weitere Partner-/Nachfolgeprojekte sind *CrypTool 2.0* und *CrypTooLinux* für Linux.

Diese Arbeit beschränkt sich auf ein Plug-in für JCrypTool, da dieses zum einen plattform-unabhängig ist und durch die *Eclipse Rich Client Platform* bereits die notwendigen Schnittstellen für Entwickler bietet. Zum anderen ist JCrypTool durch FlexiProvider mit der TU Darmstadt verwurzelt und das Plug-in wird die bisher nicht veröffentlichte S/MIME-Implementierung vom Fachgebiet *Theoretische Informatik – Kryptographie und Computeralgebra* am Fachbereich Informatik der TU Darmstadt⁵ nutzen.

2.2 Eclipse Rich Client Platform

Basis der beliebten Java-Entwicklungsplattform *Eclipse* und vieler weiterer, wie beispielsweise *Eclipse IDE for C/C++ Developer*⁶ ist die *Eclipse Rich Client Platform*. Es handelt sich um ein Framework, das entwickelt wurde, um Entwicklern eine Möglichkeit zu bieten, Applikationen durch Plug-ins aufzubauen⁷. Dafür bietet sie eine Vielzahl von Erweiterungspunkten (Extension Points), um auf Standard-Services zuzugreifen, beispielsweise erweiterbare Hilfsfunktion und Einstellungsdialoge. Zur Implementierung von grafischen Oberflächen können Entwickler *SWT* und bereits vorgefertigte *JFace* Komponenten verwenden. Das wichtigste ist allerdings die Plattform-Laufzeitumgebung (Platform Runtime). Sie kümmert sich um den Lebenszyklus einer *Eclipse RCP* Instanz, verwaltet die Plug-ins, stellt sie zur Verfügung und beinhaltet die Kern-Funktionalität (vgl. International Business Machines Corp., 2006).

¹ Kryptobibliothek, entwickelt an der TU-Darmstadt, <http://www.flexiprovider.de/>, letzter Zugriff: 01.03.2010

² <http://www.bouncycastle.org/>, letzter Zugriff: 01.03.2010

³ Aktuelle Version: 1.4.x, <http://cryptool.org/>, letzter Zugriff: 01.03.2010

⁴ vgl. „Was ist CrypTool“, <http://cryptool.org/index.php/de/about-topmenu-42.html>, letzter Zugriff: 01.03.2010

⁵ Technische Universität Darmstadt, Fachbereich Informatik, Fachgebiet Theoretische Informatik – Kryptographie und Computeralgebra. <http://www.cdc.informatik.tu-darmstadt.de/>, letzter Zugriff: 01.03.2010

⁶ <http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/galileosr2>, letzter Zugriff: 01.03.2010

⁷ vgl. <http://www.eclipse.org/platform/overview.php>, letzter Zugriff: 01.03.2010

2.3 Standard Widget Toolkit (SWT)

Das *Standard Widget Toolkit* (SWT) bietet eine Java-API zur Entwicklung von plattform-unabhängigen Komponenten zur Erstellung einer Benutzerschnittstelle. Dabei greift es auf native Komponenten des Betriebssystems zu, wo immer dies möglich ist, so dass dem Benutzer die natürliche Anmutung präsentiert wird. Es bietet Standard-Komponenten wie Buttons, Textfelder, etc. Das Eclipse-Framework und seine Bestandteile benutzen *SWT* für die graphischen Schnittstellen.

2.4 JFace

Neben *SWT* kommt *JFace* zum Einsatz. Es fasst SWT-Komponenten zu typischen wiederkehrenden Aufgaben zusammen. Dazu gehören u.a. Datei-Browser-Dialoge, Speichern-Unter-Dialoge und die Einstellungsseite. *JFace* wird von Eclipse zur Verfügung gestellt, um wiederkehrende Aufgaben mit einem immer gleich oder ähnlich aussehenden Dialog abzuarbeiten. Es ist daher äußerst empfehlenswert *JFace*-Klassen anstelle eigener Implementierungen zu benutzen, wo immer diese passend verwendet werden können. Dies bietet einen Wiedererkennungsfaktor innerhalb der entwickelten Applikation und anderer *Eclipse RCP* Anwendungen.

Abbildung 2.1 zeigt den Einstellungsdialog von *JCrypTool* unter Mac OS X. Zu erkennen ist der typische Eclipse-Einstellungsdialog mit der nativen Anmutung von Mac OS X. Dieses Aussehen hat jeder Einstellungsdialog, der das Eclipse `package org.eclipse.jface.preference` verwendet.

Der Dialog (`org.eclipse.jface.preference.PreferenceDialog`) besteht aus drei Teilen: Der Kategorieauswahl (links), der dazugehörigen „*PreferencePage*“ (rechts) und dem Steuerungsbereich (unten). Der Steuerungsbereich stellt *Buttons* zur Verfügung, die in gleicher Form auch bei anderen Dialogen erscheinen. Der rechte Teil des Dialoges zeigt die „*PreferencePage*“ (`org.eclipse.jface.preference.PreferencePage`) zu der Kategorie, die in der Kategorieauswahl (links) gewählt wurde.

JCrypTool baut auf den dazugehörigen *Extension Points* von Eclipse auf und bietet den Erweiterungspunkt „`org.eclipse.ui.preferencePages`“ und Kategorienamen (Kategorie-IDs) an, um eine neue Seite (*PreferencePage*) einzufügen. So wird beispielsweise durch den Extension Point `org.eclipse.ui.preferencePages` und durch die Kategorie `org.jcryptool.preferences.editors` eine neue *PreferencePage* in die Einstellungskategorie „Editoren“ eingefügt.

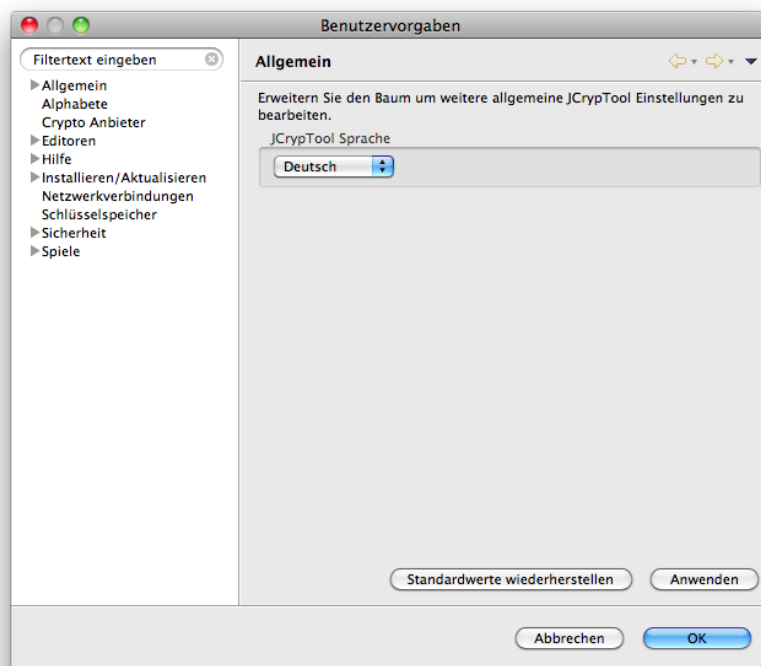


Abbildung 2.1: JCryptTool: Einstellungsdialog unter Mac OS X (10.6).

3 Theoretische Betrachtung

In diesem Kapitel werden die theoretischen Vorarbeiten geleistet, die für die erfolgreiche Planung und Implementierung eines S/MIME-E-Learning-Plug-ins erforderlich sind. In den nächsten Abschnitten wird zunächst auf den Begriff des „E-Learnings“ eingegangen. Danach wird das Thema Benutzbarkeit behandelt und anschließend werden die für S/MIME notwendigen Internet-Standards erklärt.

3.1 E-Learning

Dieser Abschnitt befasst sich mit dem Begriff „E-Learning“ und den dahinter stehenden Konzepten. Diese Konzepte werden zudem im allgemeinen Kontext der E-Learning-Plattform JCrypTool betrachtet und im speziellen Kontext des Entwurfs eines S/MIME-Plug-ins für JCrypTool.

3.1.1 Definition von E-Learning

Für E-Learning gibt es keine eindeutige Definition. Das Wort *E-Learning* ist die englische Abkürzung von *Electronic Learning*. Rey (2009) bezeichnet E-Learning als „das Lehren und Lernen mittels verschiedener elektronischer Medien“¹. Diese Definition wird auch in dieser Arbeit verwendet, hierfür muss nur noch erläutert werden, was *verschiedene elektronische Medien* (im Folgenden *Multimedia*) bedeutet. Die Definition von Multimedia ist ebenfalls umstritten. Rey bezieht sich auf die drei am häufigsten genannten Aspekte von Multimedia: „Multimedialität“, „Multicodalität“ und „Multimodalität“ und ergänzt um die Kategorie „Interaktivität“ nach Schaumburg und Issing (Schaumburg und Issing, 2009. In: Rey).

3.1.2 Multimedialität

Spricht man von Multimedia, meint man verschiedene Medien (z.B. Radio, Fernseher, Telefon, Bücher etc.). Spricht man allerdings von *Multimedialität*, sind Medien gemeint, die einen Zugang zu verschiedenen Informationsquellen bieten (vgl. Issing und Klimsa, 2002, S.483 f.). Ein Computer ermöglicht durch das Internet den Zugang zu verschiedenen Medien, beispielsweise Radio, Fernseher, Telefon, Text, (elektronische) Post und Audiospieler.

JCrypTool benutzt derzeit nur wenige verschiedene Medien. Nach dem Herunterladen der Software hat man eine Internet-unabhängige Software. Die bisher eingesetzten Medien sind Text (z.B. Hilfe-Dokumentationen), interaktive Benutzerschnittstellen (z.B. die Visualisierungen) und Spiele (z.B. Number Shark-Plug-in). Mögliche Erweiterungen wären zum einen der Einsatz von Videoplayern, beispielsweise zum Abspielen von Lehrvideos, die auch als *E-Lecture*² fungieren könnten. Des Weiteren die Verwendung von Audioplayer, beispielsweise für zusätzliche Informationen oder gar mit Anbindung an Podcasts. Zum anderen könnten zusätzliche Informationen durch Einbindungen von Wissensportalen realisiert werden. Beispielsweise könnte anstelle eines Verweisens auf einen Wikipedia-Artikel (vgl. Mul-

¹ vgl. Webseite zum Buch „E-Learning“, Rey 2009. http://www.elearning-psychologie.de/elearning_multimedia.html, letzter Zugriff: 07.01.2010

² Eine E-Lecture ist eine interaktive Vorlesungsaufzeichnung mit Folien (vgl. z.B. <http://www.lecturnity.de/>), letzter Zugriff: 07.01.2010

tipartite Key Exchange-Plug-in) dieser direkt in JCrypTool angezeigt werden.

Das geplante S/MIME-Plug-in bietet eine weitere interessante und außergewöhnliche Möglichkeit: die Anbindung von elektronischer Post. Eine signierte/verschlüsselte E-Mail (S/MIME-Nachricht) wird in der Realität durch einen E-Mail-Client verschickt und von einer anderen Person empfangen und gelesen. Daher ist ein früherer Gedanke zur Förderung der Multimedialität gewesen, dass ein *einfacher E-Mail-Client* implementiert wird, der in der Lage ist, eine erzeugte S/MIME-Nachricht zu verschicken. Der Benutzer kann diese dann in seinem eigenen E-Mail-Client entschlüsseln.

3.1.3 Multicodalität

Codierung bezeichnet die Darstellung des Lehr- und Lernmaterials (vgl. z.B. Weidenmann 2009. In: Rey). Multicodierung bedeutet daher die Darstellung von Lehr- und Lernmaterial durch verschiedene Codierungen. Mögliche Codierungen sind (Hyper-)Text, Bilder, Animationen und Simulationen³. D.h. im Folgenden ist „Lehr- und Lernmaterial“ ein gedankliches Konzept, somit ist z.B. „Text“ nicht nur eine geschriebene Information, sondern stellt bereits eine Art der Codierung von „Lehr- und Lernmaterialien“ dar.

JCrypTool setzt bereits heute schon verschiedene Formen ein, wie Text, Bilder, Animationen, und sogar Simulationen (beispielsweise in der ECC-Demonstration⁴).

Das geplante S/MIME-Plug-in soll einen erheblichen Teil an Theorie beinhalten, daher wird *Text* eine wichtige Codierung sein. Die Codierung als Text soll jedoch ansprechend wirken, daher werden alle Texte in (sehr) kurze und prägnante Paragraphen unterteilt. Zudem sollen Grafiken oder Listings als Codierung verwendet werden, die nach Möglichkeit mit Farben die Verbindung zum Text herstellen werden. Beispielsweise könnte eine Überschrift eines Paragraphen mit der gleichen Farbe unterlegt werden, wie der dazugehörige Teil in einem daneben stehenden Listing.

3.1.4 Multimodalität

Im Kontext von E-Learning bezeichnet eine Modalität eine *Sinnesmodalität* (auch als *Sinneskanal* bekannt). Eine Sinnesmodalität ist ein Sinnesorgan. Bei den heutigen medialen Angeboten beschränken sich diese allerdings auf visuelle und auditive Organe - sprich: Augen und Ohren (vgl. Issing und Klimsa, 2002. S.44 ff.). Mit Multimodalität ist somit die auditiv-visuelle Wahrnehmung gemeint. Wichtig hierbei ist, dass nur die Kombination von auditiver und visueller Wahrnehmung (z.B. ein Video) als multimodal bezeichnet wird. Ein auditiver Teil und ein visueller Teil, die unabhängig von einander auftreten, sind *monomodal*.

JCrypTool bietet nur visuelle Monomodalitäten. Auch das geplante S/MIME-Plug-in soll den Benutzer momentan nur visuell unterstützen. Die Monomodalität muss das Plug-in nicht zwingend schlechter machen.

Issing und Klimsa (2002, S.48) hierzu:
„Die am meisten verbreitete naive Annahme in diesem Bereich lautet: ‚Multimedia spricht mehrere Sinneskanäle an; das verbessert das Behalten‘“,
Issing und Klimsa kritisieren zum einen, dass Sinnesmodalität und Codierung miteinander vermischt

³ vgl. z.B. http://www.elearning-psychologie.de/multicodalitaet_i.html und http://www.elearning-psychologie.de/multicodalitaet_ii.html, letzter Zugriff: 10.01.2010

⁴ vgl. z.B. JCrypTool Version 1.0.0. (RC1)

werden. Zum anderen kritisieren sie die naive Summierungstheorie (hauptsächlich nach Ballstaedt, 2002. In: Issing und Klimsa), die davon ausgeht, dass sich der Lerneffekt von Sinnesmodalitäten aufaddiert. Zum Beispiel 20 % Wirkung durch Hören und 30 % Wirkung durch Sehen ergeben 50 % Wirkung durch Hören und Sehen.

Im Weiteren geben Issing und Klimsa auch Vorschläge für eine sinnvolle Nutzung von Multimodalität und Multicodalität. Es sollten insbesondere Studien beachten, die zeigen, dass Sinnesmodalitäten auf Reizüberflutungen anfällig reagieren (vgl. Issing und Klimsa, 2002, S.53 f.). Die Überflutung kann man durch eine Verteilung auf verschiedene Modalitäten und Codierungen reduzieren (vgl. Engelkamp und Zimmer, 2002. In: Issing und Klimsa, S.53 f.).

Die Visualisierungen von JCrypTool sind zwar nur monomodal, trotzdem wurde immer Wert auf verschiedene Codierungen gelegt. Zwar wird das geplante S/MIME-Plug-in zunächst nur auf die visuelle Unterstützung achten, aber durch die bekannten Probleme der Reizüberflutung besonderen Wert auf eine Verteilung auf verschiedene Codierungen legen (vgl. Abschnitt 3.1.3).

3.1.5 Interaktivität

Auch bei der Interaktivität gibt es verschiedene Definitionen. Diese Arbeit verwendet die Definition von Issing und Klimsa (2002), nach denen der Begriff *Interaktivität* im Bereich von Computer „[...] die Eigenschaft von Software beschreibt, dem Benutzer eine Reihe von Eingriffs- und Steuerungsmöglichkeiten zu eröffnen“ (Issing und Klimsa, 2002, S.128). Umstritten ist insbesondere die Grundform von Interaktivität. Während die einen selbst einfache „vor“- und „zurück“-Aktionen als Interaktivität verstehen, blendet z.B. Bétrancourt diese Formen als „Benutzerkontrolle“ aus (Bétrancourt 2009. In: Rey). Es gibt Merkmale an denen man steigende Interaktivität erkennen kann. Diese sind z.B. Auswählen, Umblättern, Ja/Nein- und Multiplechoice-Fragen, etwas Markieren zum Aktivieren von Zusatzfunktionen, über Fragesysteme mit der Möglichkeit freiformulierte Antworten zu geben bis hin zum freien Dialog (vgl. Issing und Klimsa, 2002).

Nach Issing und Klimsa (2002) hat die Interaktivität zwei Funktionen: individualisiertes Lernen und motiviertes Lernen. Ersteres bedeutet, dass die Software praktisch individuell die Lernbedürfnisse eines Lernenden berücksichtigen kann. Letzteres bedeutet, dass der Lernende aktiv miteinbezogen wird, was sich motivierend auswirkt.

Die Funktion von Interaktivität ist daher entscheidend für den Erfolg von E-Learning. Denn nur wenn Software mehr bieten kann als klassische Lehr- und Lerntechniken findet sie Akzeptanz.

Die Visualisierungen von JCrypTool haben immer viel Wert auf ein hohes Maß an Interaktivität gesetzt. Betrachtet man z.B. Das Plug-in „Ameisenkolonie Optimierung“, so sind sehr viele *Benutzer präventive* Interaktionsmöglichkeiten zu erkennen. Es gibt verschiedene Eingabearten (Freier Text, Selektionen, stufenlose Regler, etc), verschiedene Ansichten und steuerbare Animationen. Die „Diffie-Hellman-Schlüsselaustausch“-Visualisierung (mit elliptischen Kurven) bietet auch eine Vielzahl an Interaktionselementen: Zum Beispiel kann die elliptische Kurve relativ frei gewählt werden durch Selektion verschiedener Kurventypen, der Kurvengröße (zweistufig), Auswahl der Domänenparameter und der Wahl eines Generatorpunktes auf der Kurve.

Das S/MIME-Plug-in wird unter dem Aspekt der Interaktivität und mit Bezug zu den vorhandenen Plug-ins interaktive Elemente einbauen. So wird die E-Mail frei geschrieben werden können. Dadurch kann man beispielsweise mit Sonderzeichen die Zeichencodierung beeinflussen. Des Weiteren werden

optionale Elemente das individualisierte Lernen fördern. Der Benutzer wird darüber hinaus durch Auswahlmöglichkeiten (Ja/Nein, Multiplechoice) in möglichst viele Schritte involviert und kann durch die Möglichkeit E-Mails zu versenden und zu speichern zur Eigeninitiative über das Plug-in hinaus angeregt werden.

3.2 Benutzbarkeit (*Usability*)

Die Implementierung der Benutzerschnittstelle wird sich nach folgenden Design-Prinzipien richten:

- Erlernbarkeit
- Flexibilität
- Robustheit

Die Prinzipien der folgenden Abschnitte richten sich nach der Vorlesung *Human Computer Systems* (Schiele, 2007).

3.2.1 Erlernbarkeit

Im Folgenden werden die Prinzipien der Erlernbarkeit genannt. Zudem wird jeweils ein Beispiel gegeben, wie man die Prinzipien in dieser Arbeit verwenden und berücksichtigen kann.

- **Vorhersagbarkeit**

Die Vorhersagbarkeit soll Benutzern die Möglichkeit bieten, das Ergebnis einer Aktion schon vor deren Ausführung zu wissen.

Vorhersagbarkeit ist sehr wichtig, um einer sehr heterogenen Zielgruppe gerecht zu werden. Daher werden durch eindeutige Kennzeichnung alle Aktionen annotiert (z.B. sinnvolle Buttonbeschriftungen). So können Benutzer mit viel Erfahrung für sie uninteressante Aktionen (z.B. Erklärungen) überspringen.

- **Synthetisierbarkeit**

Auswirkungen von Aktionen sollen Benutzern sichtbar gemacht werden. Der Benutzer kann den Systemzustand beurteilen.

Ziel des Plug-ins ist, dass Benutzer jeder Zeit beurteilen können, in welchem Zustand sie sich befinden. Dies wird durch ausführliche Beschreibungen des jeweiligen Zustands erreicht. Die Zustände werden sich iterativ durch getätigte Operationen ändern, so dass die fortlaufende Änderung ersichtlich wird. Beispielsweise wird nach der Verfassung einer E-Mail im nächsten Fenster die MIME-Nachricht angezeigt.

- **Bekanntheit**

Bekanntheit beschäftigt sich damit, bekannte Elemente der realen Welt und anderer Anwendungen in der Benutzerschnittstelle zu verwenden. Die Bekanntheit ist entscheidend für den ersten Eindruck.

Für das geplante Plug-in wird daher auf Bekanntheit besonderen Wert gelegt. Benutzer sollen ihren vertrauten E-Mail-Client wiedererkennen; aus Sender- und Empfängersicht. Des Weiteren wird

durch SWT und JFace das native und bekannte Eclipse/JCrypTool-„Look-and-Feel“ realisiert. Darüber hinaus wird sich die Bedienung stark an anderen Plug-ins orientieren.

- **Generalisierbarkeit**

Generalisierbarkeit bezeichnet die Übertragbarkeit von einem System/einer Operation/einer Aktion etc. auf andere Systeme, Operationen und Aktionen (z.B. „Drag-And-Drop“).

Die Benutzung des Plug-ins wird mit vorhandenen Plug-ins vergleichbar sein. Zusätzlich werden Teilschritte wie z.B. „Verschlüsselung“ und „Signaturerzeugung“ sehr ähnliche Dialogfenster erhalten. Des Weiteren werden die Teilschritte ähnlich gestaltet. Darüber hinaus schafft JFace ein Höchstmaß an Ähnlichkeit in Bezug auf das Aussehen und das Verhalten (vgl. Abschnitt 2.4).

- **Konsistenz**

Unter Betrachtung der Erlernbarkeit, bedeutet Konsistenz, dass das System in ähnlichen Situationen ähnlich reagiert. Dies bezieht sich auch auf Ausgaben, Interaktionen und das Layout.

Im Allgemeinen wird die Konsistenz innerhalb des Plug-ins, anderer Plug-ins und der *Eclipse Rich Client Platform* (Eclipse RCP) sehr hoch gehalten werden. Im Speziellen werden ähnliche Ansichten und ähnliche Dialoge gestaltet. Das Layout und das Verhalten werden ebenfalls Ähnlichkeiten zu bestehenden Systemen aufweisen.

3.2.2 Flexibilität

Dieser Unterabschnitt listet die Prinzipien der Flexibilität auf und beschreibt den möglichen Einsatz in dieser Arbeit.

- **Dialog Initiative**

Bei der Dialog Initiative unterscheidet man zwischen *Benutzer präventiv*, d.h. der Benutzer bzw. die Benutzerin initiiert eine Aktion, und *System präventiv*, das bedeutet, dass das System Benutzer zur Eingabe auffordert. *Benutzer präventiv* gilt als flexibler und besser, jedoch lässt sich ein *System präventives* Verhalten nicht immer vermeiden.

Das Plug-in wird überwiegend *Benutzer präventiv* agieren. An bestimmten Stellen ist für die weitere Verarbeitung ein *System präventives* Verhalten notwendig, z.B. wenn der Nutzer sich zwischen Signaturerstellung und Verschlüsselung entscheiden muss.

- **Multithreading**

Multithreading ermöglicht parallele Abarbeitung verschiedener Aufgaben.

Dies wird im Allgemeinen bereits durch die Eclipse Plattform gewährleistet. So können mehrere Plug-ins ohne Beeinträchtigungen parallel ausgeführt werden. Innerhalb des Plug-ins wird dies nicht notwendig sein.

- **Migrationsmöglichkeit von Aufgaben**

Die Migrationsmöglichkeit von Aufgaben bedeutet, dass Teilaufgaben vom System oder vom Nutzer erledigt werden können. Hier sollte derjenige die Aufgabe übernehmen, der sie besser lösen kann (z.B. Rechtschreibkontrolle).

Generell sollen bei diesem E-Learning-Projekt möglichst viele Aufgaben vom Benutzer ausführbar sein. Allerdings können die Berechnungen der kryptographischen Verfahren nicht vom Benutzer übernommen werden. Außerdem wird das System die Eingaben des Nutzers prüfen (z.B. die Eingabe einer syntaktisch korrekten E-Mail-Adresse).

- **Ersetzbarkeit**

Ersetzbarkeit bedeutet, dass Aufgaben auf alternativen Wegen erledigt werden können. Beispielsweise unterschiedliche Ausgabeformate und verschiedene Ausführungsmöglichkeiten, die zum gleichen Ziel führen.

Die Ersetzbarkeit wird an mehreren Stellen realisiert. Zum einen wird ein E-Mail-Client die Standardausgabe ersetzen. Zudem können S/MIME-Nachrichten gespeichert werden. Zum anderen werden mehrere Aktionen optional sein, so dass unterschiedliche Ausführungswege ermöglicht werden.

- **Konfigurierbarkeit**

Konfigurierbarkeit heißt, dass die Benutzeroberfläche angepasst werden kann.

Dies wird im Umfang der typischen Eclipse-RCP-Eigenschaften ermöglicht. Beispielsweise kann die Fensteranordnung frei gestaltet werden. Innerhalb des Plug-ins werden die Dialog- und Wizard-Fenster flexibel gehalten. Veränderungen werden nach Möglichkeit adaptiert (z.B. durch Scroll-Bars).

3.2.3 Robustheit

Das dritte UI-Design Prinzip ist die „Robustheit“, die sich in die folgenden Prinzipien unterteilen lässt.

- **Beobachtbarkeit**

Beobachtbarkeit bedeutet, dass Benutzer erkennen können, in welchem Zustand sie sich befinden.

Besonders in einem E-Learning-Projekt bietet sich an, Benutzer schrittweise durch den Ablauf zu führen. Durch die unterschiedlichen Schritte (z.B. E-Mail schreiben, MIME-Format anzeigen, Verschlüsseln) kann der Benutzer genau erkennen, in welchem Zustand er sich befindet.

- **Wiederherstellbarkeit (Recoverability)**

Wiederherstellbarkeit heißt, dass Benutzer einen Zustand wiederherstellen können oder den derzeitigen Zustand verbessern können. Ferner wird zwischen *backward recovery* und *forward recovery* unterschieden. Ersteres heißt, dass es möglich ist Fehler rückgängig zu machen. Letzteres wiederum, dass etwas korrigiert werden kann. Die Schwierigkeit einer Wiederherstellung sollte sich am Aufwand der getätigten Aktion orientieren.

Eingabe-Schnittstellen werden nach der Eingabe nicht blockiert, somit wird ermöglicht, Fehler durch *forward recovery* zu beheben. Beispielsweise kann die Eingabe des E-Mail-Textes jederzeit wiederholt werden und eine Verschlüsselung oder eine Signatur nochmals erstellt werden.

- **Ansprechbarkeit**

Die Ansprechbarkeit bezeichnet die subjektive Interaktionsgeschwindigkeit aus Benutzersicht.

Die Ausführzeiten von JCrypTool sind sehr schnell. Der einzige Engpass kann während der Erzeugung der S/MIME-Nachricht auftreten. Das Plug-in ermöglicht nur Signatur-Erstellung und Verschlüsselung für Text. Somit wird die Ausführungszeit maximal der eines gewöhnlichen E-Mail-Clients entsprechen und für den Benutzer kein Problem darstellen.

- **Aufgabenanpassung**

Die Aufgabenanpassung verlangt, dass Benutzer möglichst alle Aufgaben erledigen können, die sie vom System erwarten. Außerdem sollen die Aufgaben in der erwarteten Art und Weise erfüllt werden. Das Prinzip der Aufgabenanpassung umfasst auch das Verständnis darüber, wie Aufgaben mit dem System erledigt werden können. Darüber hinaus, ob Benutzer eigene Aufgaben definieren können.

Dieses Prinzip lässt sich nicht ohne Weiteres erfüllen. Ein Aspekt ist die Betrachtung der Zielgruppe und das Verständnis über deren Anforderungen (siehe auch Abschnitt 4.2). Ein weiterer Gesichtspunkt ist eine ausführliche Evaluation, die möglichst die ganze Zielgruppe abdeckt. Dies ist im Rahmen dieser Arbeit nicht zu leisten, sondern erst, wenn das Plug-in durch die Veröffentlichung an die gesamte Zielgruppe herangeführt wird.

3.3 Internet-Standards

Folgende Internet-Standards werden während der S/MIME-Visualisierung durch das Plug-in erklärt.

3.3.1 RFC 822

Der RFC 822-Standard ist der klassische E-Mail-Standard, auf den bis heute trotz seiner Defizite nicht vollkommen verzichtet werden kann. Die Konzepte sind sehr an die Realität gebunden. So stellt man sich eine einfache Nachricht vor, wie einen Brief mit Briefumschlag (vgl. Tanenbaum 2003, S.649). Der Umschlag mit Anschrift wird durch *Header* beschrieben. Der *Text* der nach dem Header kommt stellt den Brief dar. Außerdem beschränkt sich dieses einfache E-Mail-Format auf den ASCII-Zeichensatz, der nur englische Standardzeichen enthält und in der heutigen global-vernetzten Welt nicht mehr Bestand haben kann. Die Gründe hierfür sind, dass der Standard bereits aus dem Jahr 1982 stammt und für ARPANET⁵ geschaffen wurde⁶. Zu dieser Zeit wusste man noch nicht, dass das Internet in seiner heutigen multimedialen Form (Text, Video, Audio, etc.) genutzt werden wird. Durch seine große Verbreitung spielt der Standard bis heute eine wichtige Rolle und daher baut der heutige Nachfolger (MIME) nur ergänzend auf dem RFC 822 Standard auf. Die wichtigsten Header des Standards sind: *To:*, *Cc:* und *Bcc:* für die Adressaten, *From:* und *Sender:* für den Absender sowie *Subject:* für den Betreff und *Date:* für das Datum. Es existieren weitere Header (z.B. *Received:*), die in dieser Arbeit nicht beschrieben, da sie für das grundlegende Verständnis nicht benötigt werden.

3.3.2 Multipurpose Internet Mail Extensions (MIME)

Der RFC 822-Standard definiert Header und behandelt sonstigen Text nicht. Allerdings besteht die Einschränkung durch ASCII-Zeichen. Ein Nachfolgestandard, der auf RFC 822 aufbaut muss daher den Textteil verwenden um abwärtskompatibel zu sein. Ferner muss er die Einschränkung auf den ASCII-Zeichensatz beachten. Die am weitesten verbreitete und heute gängige Lösung ist der als *Multipurpose*

⁵ ARPANET ist einer der Vorgänger des Internets.

⁶ <http://tools.ietf.org/html/rfc822>, letzter Zugriff: 15.02.2010

Internet Mail Extension (MIME) bekannte Standard. Er basiert auf dem RFC 1341, der durch RFC 2045-2049 überarbeitet wurde (vgl. Tanenbaum 2003, S.649). Im wesentlichen definiert der Standard zwei Konzepte. Zum einen zusätzliche Header (z.B. MIME-Version und Content-Type) und zum anderen Kodierregeln für nicht-ASCII-Zeichen (quoted-printable, Base64). Zwei weitere Besonderheiten sind, dass MIME ein erweiterbares Typ- und Untertyp-Konzept hat (z.B. text/plain) und mehrere Nachrichten transportiert werden können. Dies bildet die Grundlage für S/MIME (siehe auch Abschnitt 3.3.3). Im Plug-in zu dieser Arbeit werden diese Konzepte erklärt. Speziell sollen die Benutzer das Konzept der Erweiterbarkeit, die Notwendigkeit von Kodierregeln und das Typ- und Untertypsystem verstehen.

3.3.3 Secure MIME (S/MIME)

Der Internet-Standard RFC 5751 definiert die derzeit aktuelle Version von S/MIME (Version 3.2)⁷. Er wurde erstellt, um sichere MIME-Daten zu senden. Er definiert Regeln zur Erzeugung signierter MIME-Nachrichten und verschlüsselter MIME-Nachrichten bzw. verschlüsselte und signierte MIME-Nachrichten. Insbesondere ist S/MIME unabhängig von konkreten kryptographischen Verfahren. Darüber hinaus können Daten komprimiert werden, aber in dieser Arbeit wird nicht weiter auf dieses Thema eingegangen.

S/MIME macht sich die Erweiterbarkeit von MIME zu Nutze und definiert neue Content-Types. Definiert wurden die Untertypen *pkcs7-mime* und *pkcs7-signature* für den Basistyp *application*. Der Content-Type *application/pkcs7-mime* von S/MIME kann genutzt werden, um CMS-Nachrichten vom CMS-Content-Type *EnvelopedData*, *SignedData* und *CompressedData* zu verschicken (siehe auch *Cryptographic Message Syntax*, Abschnitt 3.3.4). Der Content-Type *application/pkcs7-signature* ist nur für CMS-Nachrichten vom CMS-Content-Type *SignedData* vorgesehen. Dieser Content-Type ist somit für MIME-Nachrichten gedacht, die nur signiert werden sollen.

Die S/MIME-Verschlüsselung verwendet ein Hybridverfahren, d.h., dass Nachrichten mit einem symmetrischen Schlüssel verschlüsselt werden, und der asymmetrische öffentliche Schlüssel verschlüsselt den symmetrischen Schlüssel. Dieses Verfahren wird angewendet, da es wesentlich schneller ist als ein reines asymmetrisches Verfahren.

Die Signatur-Erzeugung von S/MIME ermöglicht es, mehrere Attribute zusätzlich zur Nachricht zu signieren. Dies wird durch das Verwenden von CMS realisiert. Die signierten Attribute (z.B. *signingTime*) geben dem Empfänger zusätzliche Informationen, die schützenswert sind.

Das Plug-in zu dieser Arbeit wird die grundlegenden Konzepte von S/MIME erklären. Darüber hinaus wird auf das Hybridverfahren und die spezielle Bedeutung der signierten Attribute eingegangen. Ferner soll der Benutzer verstehen, welche Teile der Nachricht geschützt werden (MIME-Nachricht und signierte Attribute) und welche Teile ungeschützt bleiben (z.B. die RFC 822-Header).

RFC 5751 wurde erst im Januar 2010 veröffentlicht und löst die S/MIME-Version 3.1 ab (RFC 3851). Daher gab es einen Wechsel während der Erstellung dieser Arbeit. Allerdings gab es keine Änderungen, die Auswirkungen auf diese Arbeit oder das Plug-in haben.

⁷ <http://tools.ietf.org/html/rfc5751>, letzter Zugriff: 09.03.2010

3.3.4 Cryptographic Message Syntax CMS

Die *Cryptographic Message Syntax* (CMS) definiert ein Format für kryptographische Nachrichten. Die aktuelle Version wird in RFC 5652 spezifiziert und wurde im September 2009 veröffentlicht⁸. Wichtig zu wissen ist, dass der CMS-Standard der Nachfolger von PKCS7 ist, daher beginnen die Namen der Untertypen von S/MIME mit „pkcs7-“. Dies wird auch zur Kompatibilität bestehen bleiben und wurde daher nicht in der aktuellen Version von S/MIME geändert.

CMS ermöglicht das digitale Signieren und Verschlüsseln von Daten sowie die Anwendung von Hash- und MAC-Funktionen. Dafür wurden Strukturen definiert, die viele verschiedene Arten von kryptographischen Nachrichten unterstützen. Ferner werden Kodierregeln für die Bestandteile festgelegt (z.B. *Distinguished Encoding Rules* (DER)).

S/MIME nutzt die *Cryptographic Message Syntax*, um seine Nachrichten zusammenzustellen. Von besonderem Interesse sind die Strukturen von *EnvelopedData* und *SignedData*, die für verschlüsselte bzw. signierte Nachrichten verwendet werden. Diese Strukturen und deren Bestandteile werden vom Plug-in erklärt werden.

⁸ <http://tools.ietf.org/html/rfc5652>, letzter Zugriff: 09.03.2010

4 Anforderungsanalyse

Im Folgenden werden die einzelnen Schritte der Planung beschrieben.

4.1 Welche Konzepte sollen erklärt werden?

Das Hauptziel des Plug-ins ist das Verständnis des Benutzers über S/MIME. S/MIME ist ein komplexer Standard und hat sehr viele Abhängigkeiten. So funktioniert S/MIME nur in Verbindung mit anderen Standards, wie z.B. RFC 822, MIME, CMS, kryptographischen Verfahren, Public-Key-Infrastrukturen, dem Internet und vielem mehr. Dahinter stehen wiederum komplexe und komplizierte Internetstandards. Daher wurde sich für eine Auswahl dieser Konzepte entschieden, die mit Fokus auf die Prozesse auf Benutzerseite ausgewählt wurden. Die gewählten Konzepte sind RFC 822, MIME, CMS, S/MIME-Verschlüsselung (*EnvelopedData*) und S/MIME-Signatur (*SignedData*). Diese Konzepte werden ausschließlich hinsichtlich S/MIME erläutert.

4.2 Zielgruppe

Folgender Abschnitt beschäftigt sich mit der Analyse der Ziel- und Interessengruppen einer S/MIME-Visualisierung für JCrypTool.

4.2.1 Analyse der Zielgruppen

Bei JCrypTool handelt es sich um ein frei verfügbares Open-Source-Projekt, dass von jedem Internetnutzer weltweit heruntergeladen werden kann. JCrypTool bietet Visualisierungs-Plug-ins für verschiedene Altersgruppen. Für Einsteiger, mit geringen mathematischen Kenntnissen, eignet sich beispielsweise die RSA-Visualisierung. Komplexere Visualisierungen, wie beispielsweise die ECC-Demonstration, die mit elliptischen Kurven arbeitet, sowie das Plug-in „Graphenisomorphie“, erfordern zumindest weitergehende mathematische Kenntnisse bzw. Grundkenntnisse der Graphentheorie.

Bekannt ist, dass (J)CrypTool von Schullehrern für den Unterricht genutzt wird¹. Außerdem sind die Schnittstellen zu den Algorithmen der Kryptobibliotheken (z.B. RSA, AES, DSA, SHA1) auch für Kryptographie-Vorlesungen interessant². Die Vorgänger-Software CrypTool wurde ursprünglich für IT-Sicherheitsschulungen entwickelt. Daher ist insbesondere eine S/MIME-Visualisierung ein Kandidat für derartige Schulungen.

4.2.2 Definition der Zielgruppen

Aufgrund der Analyse der Zielgruppen deckt die geplante Visualisierung insbesondere folgende Zielgruppen ab:

- 1. Schülerinnen und Schüler mit wenig mathematischen und kryptographischen Erfahrungen

¹ vgl. Cryptoportal für Lehrer, <https://www.cryptoportal.org/>, letzter Zugriff: 18.02.2010

² vgl. z.B. Stoffplan der Vorlesung „Einführung in die Kryptographie“ an der TU Darmstadt, vgl. Modulhandbuch <http://www.mhb.informatik.tu-darmstadt.de/>, letzter Zugriff: 18.02.2010

-
- 2. Studenten und Studentinnen mit wenigen bis fortgeschrittenen mathematischen und kryptographischen Erfahrungen
 - 3. Mitarbeiterinnen und Mitarbeiter in Unternehmen, die an IT-Sicherheitsschulungen teilnehmen oder auf sichere E-Mail-Übertragung angewiesen sind.

Lehrer und Lehrerinnen können als weitere Interessengruppe definiert werden. Sie haben ein hohes Interesse daran, dass JCrypTool Visualisierungen beinhaltet, die für Schüler geeignet und motivierend sind.

4.2.3 Besondere Anforderungen der Zielgruppen

Insbesondere die letzte Zielgruppe umschließt eine große Gruppe von Menschen aus technischen und aus nicht-technischen Bereichen, sowie Personen mit viel und Personen mit wenig Computerumgang. Daher muss auf die *intuitive Nutzung* und die Erklärung *grundlegender Begriffe* Wert gelegt werden. Bei der ersten und zweiten Gruppe kann man von Computererfahrungen ausgehen, daher muss die Benutzbarkeit durch einen *hohen Wiedererkennungswert* unterstützt werden. Speziell die erste Gruppe muss durch besondere Lernanreize unterstützt werden. Beispielsweise durch kurze verständliche Texte sowie durch Farben. Der Gruppe der fortgeschrittenen Studenten und der Gruppe der Mitarbeiter mit Erfahrungen muss es ermöglicht werden, bekannte Themen zu überspringen. Daher müssen *alternative Navigationsflüsse* modelliert werden.

4.3 Globalisierung

So wie in der Informatik üblich, ist auch bei JCrypTool Englisch die Standard-Sprache. Trotzdem ist eine deutsche Übersetzung stark nachgefragt und wird für die Implementierung eines Plug-ins als Zweitsprache empfohlen. Dieser und die folgenden Abschnitte beschäftigen sich daher mit der Umsetzung einer internationalisierten Software.

Der Begriff *Globalisierung* von Software umfasst die beiden Disziplinen *Internationalisierung* und *Lokalisierung* von Software. Diese Disziplinen ermöglichen das Nutzen einer Software unabhängig von der Sprache der Nutzer. Es vergrößert somit nicht nur die Zielgruppe, sondern ist ein hohes Qualitätsmerkmal und ermöglicht den internationalen Zugang.

4.3.1 Internationalisierung

Internationalisierung von Software bedeutet, dass eine Software Zeichen und Eigenheiten von Sprachen unterstützt. Dies bedeutet z.B., dass man eine Zeichenkodierung verwendet, die auch beispielsweise japanische und hebräische Zeichen unterstützt. Letzteres ist ein Spezialfall. Hebräisch ist nämlich eine *bidirektionale Sprache*. So schreibt man hebräische Zeichen zwar von rechts nach links, jedoch werden Zahlen von links nach rechts geschrieben. Das bedeutet für eine internationalisierte Software, dass sie bei Eingabe von Zeichen den Cursor von rechts nach links bewegen muss, und von links nach rechts, wenn Ziffern eingegeben werden. Zusätzlich muss der Cursor „springen“, wenn während der Zeicheneingabe Ziffern eingegeben werden.

Ein weiterer Bestandteil der Internationalisierung ist die Ermöglichung von phonetischen Eingaben. Anstelle der Eingabe von Schriftzeichen kann man auch die Phoneme eingeben. Beispielsweise ermöglicht Googles Übersetzungsdienst eine phonetische Eingabe. Wählt man als Ausgangssprache Arabisch, kann man Phoneme eingeben. Hat man Deutsch als Zielsprache gewählt und gibt man „marhaban“ ein

und betätigt die Leertaste, so wird die Eingabe in die arabischen Zeichen umgewandelt und in der Übersetzungszeile erscheint „Hallo“³.

Viele Eigenheiten, insbesondere bei bidirektionalen Sprachen, erfordern lange Erfahrung und gute Kenntnisse und werden in diesem Projekt nicht unterstützt. Dieses Plug-in wird aber auf eine internationale Kodierung (UTF8) Wert legen. Im Speziellen werden auch internationale Zeichen bei der Erstellung von S/MIME-Nachrichten erlaubt, so wie dies auch durch „normale“ E-Mail-Clients erfüllt werden sollte.

4.3.2 Lokalisierung

Lokalisierung von Software bedeutet, dass eine Software in verschiedenen Sprachen angezeigt werden kann. Um dies zu ermöglichen müssen alle Zeichenketten exportierbar sein. Dies wird in Java durch .properties-Dateien ermöglicht. Eclipse bietet dafür einen speziellen Mechanismus an, der bereits bei anderen Plug-ins verwendet wurde. Daher wird dieses Plug-in kompatibel zu diesem Mechanismus implementiert.

Bei der Lokalisierung muss man allerdings auf ein paar einfache Regeln beachten:

1. Zeichenketten dürfen nicht zusammengefügt werden.

Trennt man zur Einfachheit beispielsweise „Please select a language“ und „Please select a color“ in „Please select“, „a language“ und „a color“, und lässt sie übersetzen, so sind die Übersetzungen beispielsweise im Deutschen „Bitte auswählen“, „eine Sprache“ und „eine Farbe“. Somit wären die zusammengefügte Sätze in der Software: „Bitte auswählen eine Sprache“ und „Bitte auswählen eine Farbe“, die im Deutschen nicht korrekt sind.

2. Sätze dürfen nicht parametrisiert werden.

Die Sätze „Please select the title“ und „Please select the author“ könnten zum Wiederverwenden durch „Please select {0}“, „the title“ und „the author“ parametrisiert werden. Würde man allerdings dies ins Deutsche übersetzen, so hieße es: „Bitte wählen Sie {0} aus“, „der Titel“ und „der Autor“. Dies würde zu den inkorrekten Sätzen „Bitte wählen Sie der Titel aus“ und „Bitte wählen Sie der Autor aus“ führen.

Ein Parameter soll nur verwendet werden, wenn man einen Namen, eine Zahl, einen Typ oder ähnliches einfügen muss. Z.B. „The file '{0}' has been deleted.“ oder „Hard drive free space required for install: {0} MB.“.

3. Parameter dürfen nicht von der Reihenfolge abhängig sein.

Javas .properties-Dateien erfüllen bereits diese Regel durch die Nummerierung von Parametern (z.B. {0},{1},{2},...). Bei anderen Programmiersprachen, die eigene Dateikonzepte für die Lokalisierung erfordern, hat man oft mehr Freiheit zum Benennen. Beispielsweise kann man den Satz „%string% MB free space on drive %string%.“ exportieren und den ersten String immer durch die MB-Anzahl und den zweiten String durch den Laufwerksnamen (z.B. C:\) ersetzen. Diese Reihenfolge kann aber durch einen Übersetzer vertauscht werden. Beispielsweise ist eine mögliche deutsche Übersetzung: „Laufwerk %string% hat %string% MB freien Speicherplatz.“. In dieser Übersetzung wurden die Parameter vertauscht, somit würden die Strings falsch ersetzt werden.

In diesen Fällen muss man eindeutige Bezeichner wählen. Beispielsweise: „%freespace% MB free space on drive %drive%.“.

³ <http://translate.google.de>, letzter Zugriff: 07.02.2010

.properties-Dateien haben eine Schwachstelle: Der Übersetzer kennt nicht die Bedeutung der nummerierten Parameter. Diese können aber für die Übersetzung entscheidend sein. Beispielsweise kennt ein Übersetzer die Bedeutung von „{0} {1}“ nicht. Handelt es sich aber um „first name“ und „last name“ (dt. Vorname und Nachname), so weiß der Übersetzer, dass die Parameter unter Umständen umgedreht werden müssen.

Solche Fälle demonstrieren, dass man zur professionellen Lokalisierung auf weitere Werkzeuge angewiesen ist. Diese Werkzeuge können dann verwendet werden, um die zu übersetzenden Nachrichten zu pflegen, mit Zusatzinformationen zu verwalten und daraus Sprachpakete (d.h. alle benötigten .properties-Dateien einer Sprache) zu generieren.

Die Plug-in-Version zu Ende dieser Arbeit soll alle zu übersetzenden Zeichenketten in .properties-Dateien exportiert haben. Zusätzlich werden alle Zeichenketten in deutscher und englischer Sprache vorhanden sein. Dabei wird besonderen Wert auf die oben beschriebenen Regeln gelegt. Die beim Starten von JCrypTool gewählte Sprache wird vom Plug-in adaptiert und entsprechend verwendet. Sollte die gewählte Sprache nicht vorhanden sein, verwendet das Plug-in Englisch als Standard-Sprache.

JCrypTool benutzt keine zusätzliche Software zum Lokalisieren und keine zentrale Datenbank. Der verwendete Eclipse-Mechanismus sorgt lediglich dafür, dass die Zeichenketten aus den Quellcode-Dateien exportiert und als .properties-Dateien gepflegt werden können. Somit wird das geplante Plug-in auf keine weitere Software zurückgreifen und keine eigenen Datenbanken erstellen. Die bekannten Schwachstellen von Javas .properties-Dateien werden nach Möglichkeit umgangen.

4.4 Akteure im System

Das System interagiert mit der Benutzerin bzw. dem Benutzer von JCrypTool. Darüber hinaus existieren keine weiteren Akteure. Für die weitere Anforderungsanalyse wird die Benutzerin bzw. der Benutzer durch die Rolle *der Benutzer* bezeichnet⁴.

4.5 Feature-Liste

Aus der Anforderungsanalyse ergeben sich die in Tabelle 4.1 dokumentierten Features, die realisiert werden sollen.

4.6 Anwendungsfälle (Use Cases)

Tabelle 4.2 bis Tabelle 4.8 dokumentieren die geplanten Anwendungsfälle (im Folgenden: *Use-Cases*).

⁴ Der Verfasser dieser Arbeit stellt den Grundsatz der Gleichberechtigung damit in keiner Weise in Frage.

Tabelle 4.1: Feature-Liste.

Name	Beschreibung
Mail-Ausgangsserver	Eine erzeugte S/MIME-Nachricht kann per E-Mail verschickt werden. Notwendige Serverparameter sind persistent konfigurierbar.
Signaturprüfung/Entschlüsselung im eigenen E-Mail-Client	Eine im Plug-in erstellte S/MIME-Nachricht ist im eigenen E-Mail-Client überprüfbar/entschlüsselbar. Export-Möglichkeit für das CA-Zertifikat ist vorhanden.
JCrypTool-typische Anmutung	Das Plug-in hat eine durch SWT und JFace grafische Benutzeroberfläche. Das Plug-in soll in die JCrypTool-Plattform integrierbar sein.
Lokalisierung Englisch und Deutsch	Das Plug-in ist Deutsch und Englisch lokalisiert.
E-Mail-Client Look-and-Feel	Das Plug-in zeigt den Weg von der E-Mail-Erzeugung über MIME und S/MIME bis zur Empfängerseite. Wiedererkennungswert durch typischen E-Mail-Client-Aufbau.
E-Mail-Schnittstelle	Eine typische E-Mail-Schnittstelle mit Standardfeldern (An, Betreff, Text) steht zur Verfügung.
MIME-Nachrichten	Aus der E-Mail-Schnittstelle kann eine MIME-Nachricht erzeugt werden.
S/MIME-Nachrichten	Aus einer MIME-Nachricht kann eine S/MIME-Nachricht erzeugt werden (Verschlüsselung und Signaturerzeugung).
S/MIME-Entschlüsselung/Signaturprüfung	Eine durch das Plug-in erzeugte S/MIME-Nachricht kann entschlüsselt werden bzw. die Signatur kann geprüft werden.
Alternative Abläufe	Erklärungen können optional angezeigt werden. Erklärungen können übersprungen werden.
Kontextsensitive Erklärungen	Verschlüsselung/Signaturerzeugung liefern jeweils kontextsensitive Erklärungen.
S/MIME-Nachrichten abspeichern	Erzeugte S/MIME-Nachrichten können auf dem Dateisystem gespeichert werden.
RFC 822- und MIME-Erklärung	Die Internetstandards RFC 822 und MIME werden grundlegend erklärt. RFC 822- und MIME-Anteile können vom Nutzer unterschieden werden. Erweiterbarkeitsprinzip von MIME-Nachrichten wird erklärt.
S/MIME-Signatur-Erklärung	Der Ablauf zur Erzeugung einer S/MIME-Nachricht mit Signatur wird erklärt. Es werden die Signatur-relevanten Bestandteile von CMS grundlegend erklärt. Insbesondere werden die signierten Attribute (engl. Signed Attributes) erläutert.
S/MIME-Verschlüsselung-Erklärung	Der Ablauf zur Erzeugung einer verschlüsselten S/MIME-Nachricht wird beschrieben. Es werden die Verschlüsselungs-relevanten Bestandteile von CMS grundlegend erklärt. Es wird die Hybrid-Verschlüsselung im Allgemeinen erklärt. Das Hybrid-Verfahren für S/MIME wird erklärt.

Tabelle 4.2: Use Case 1 – Einfache E-Mail erstellen.

Use Case 1	Einfache E-Mail erstellen
Beschreibung	Die E-Mail-Schnittstelle ermöglicht die Erstellung einer einfachen E-Mail im MIME-Format.
Akteure	Benutzer.
Vorbedingungen	Senderansicht geöffnet.
Nachbedingungen	Erzeugte MIME-Nachricht wird im mittleren Editor angezeigt.
Normaler Ablauf	Die Felder „An“, „Betreff“ und „Textfeld“ werden gültig ausgefüllt. Die Eingabe wird durch den „Weiter“-Button bestätigt.
Alternativer Ablauf	Die Felder „An“, „Betreff“ und „Textfeld“ werden ungültig (ungültige E-Mail-Adresse) ausgefüllt. Die Eingabe wird durch den „Weiter“-Button bestätigt. Es erscheint eine Fehlermeldung.
Status	Implementiert.

Tabelle 4.3: Use Case 2 – S/MIME-Nachricht mit Signatur erzeugen.

Use Case 2	S/MIME-Nachricht mit Signatur erzeugen.
Beschreibung	Aus einer MIME-Nachricht wird eine S/MIME-Nachricht mit Signatur erstellt.
Akteure	Benutzer.
Vorbedingungen	Senderansicht geöffnet, MIME-Nachricht erstellt.
Nachbedingungen	Vorbedingung bleibt erhalten. Zusätzlich wird die erzeugte S/MIME-Nachricht im rechten Editor angezeigt.
Normaler Ablauf	Der MIME-Editor-„Weiter“-Button wird betätigt. Ein Wizard öffnet sich. „Signieren“ wird ausgewählt. Auswahl wird durch Bestätigung des „Fertig“-Button bestätigt.
Alternativer Ablauf	-
Status	Implementiert.

Tabelle 4.4: Use Case 3 – S/MIME-Nachricht mit Verschlüsselung erstellen.

Use Case 3	S/MIME-Nachricht mit Verschlüsselung erstellen.
Beschreibung	Aus einer MIME-Nachricht wird eine S/MIME-Nachricht mit Verschlüsselung erstellt.
Akteure	Benutzer.
Vorbedingungen	Senderansicht geöffnet, MIME-Nachricht erstellt.
Nachbedingungen	Vorbedingung bleibt erhalten. Zusätzlich wird die erzeugte S/MIME-Nachricht im rechten Editor angezeigt.
Normaler Ablauf	Der MIME-Editor-„Weiter“-Button wird betätigt. Ein Wizard öffnet sich. „Verschlüsseln“ wird ausgewählt. Auswahl wird durch Bestätigung des „Fertig“-Buttons bestätigt.
Alternativer Ablauf	-
Status	Implementiert.

Tabelle 4.5: Use Case 4 – S/MIME-Nachricht als E-Mail versenden.

Use Case 4	S/MIME-Nachricht als E-Mail versenden.
Beschreibung	Eine erzeugte S/MIME-Nachricht wird als E-Mail versendet.
Akteure	Benutzer.
Vorbedingungen	Senderansicht geöffnet, S/MIME-Nachricht erstellt und E-Mail-Client gültig konfiguriert.
Nachbedingungen	Vorbedingung bleibt erfüllt. E-Mail wurde versendet. „Senden“-Button ist deaktiviert. Bestätigungsnachricht wird angezeigt.
Normaler Ablauf	<ol style="list-style-type: none"> 1. „Senden“-Button drücken. 2. Im geöffneten Wizard „Als E-Mail versenden“ auswählen. 3. Bis zur „E-Mail-Senden“-Seite weiter klicken. 4. Mit Senden bestätigen. 5. Passwort eingeben und bestätigen. 6. E-Mail wurde gesendet, Bestätigung wird angezeigt.
Alternativer Ablauf	-
Status	Implementiert.

Tabelle 4.6: Use Case 5 – Senden.

Use Case 5	Senden
Beschreibung	Erzeugte S/MIME-Nachricht wird „virtuell“ gesendet, d.h. in der Empfängeransicht angezeigt.
Akteure	Benutzer.
Vorbedingungen	Senderansicht geöffnet, S/MIME-Nachricht erstellt.
Nachbedingungen	Empfängeransicht geöffnet, S/MIME-Nachricht wird im linken Editor der Empfängeransicht angezeigt.
Normaler Ablauf	<ol style="list-style-type: none"> 1. „Senden“-Button drücken. 2. Im geöffneten Wizard „Entschlüsseln/Verifizieren“ auswählen. 3. Mit dem „Fertig“-Button bestätigen. 4. S/MIME-Nachricht wird in der Empfängeransicht angezeigt.
Alternativer Ablauf	<ol style="list-style-type: none"> 1. „Senden“-Button drücken. 2. Im geöffneten Wizard „Entschlüsseln/Verifizieren“ auswählen. 3. Zudem wird „Speichern unter“ und/oder „Als E-Mail senden“ ausgewählt. 4. E-Mail wird gespeichert und/oder als E-Mail versendet. 5. Mit dem „Fertig“-Button bestätigen. 6. S/MIME-Nachricht wird in der Empfängeransicht angezeigt.
Status	Implementiert.

Tabelle 4.7: Use Case 6 – Signatur überprüfen.

Use Case 6	Signatur überprüfen.
Beschreibung	Eine S/MIME-Nachricht mit Signatur wird überprüft.
Akteure	Benutzer.
Vorbedingungen	Empfängeransicht geöffnet. S/MIME-Nachricht mit Signatur im linken Editor geöffnet.
Nachbedingungen	Empfängeransicht geöffnet. Rechter Editor zeigt überprüfte E-Mail und Feedback zur Signatur an.
Normaler Ablauf	<ol style="list-style-type: none"> 1. „Weiter“-Button wird betätigt. 2. S/MIME-Nachricht mit Signatur wird überprüft. 3. E-Mail wird im rechten Editor angezeigt. 4. Der Benutzer bekommt durch „signiert“ angezeigt, dass die Nachricht signiert wurde.
Alternativer Ablauf	-
Status	Implementiert.

Tabelle 4.8: Use Case 7 – S/MIME-Nachricht entschlüsseln.

Use Case 7	S/MIME-Nachricht entschlüsseln.
Beschreibung	Eine erzeugte S/MIME wird entschlüsselt und präsentiert.
Akteure	Benutzer.
Vorbedingungen	Empfängeransicht geöffnet. S/MIME-Nachricht mit Verschlüsselung im linken Editor geöffnet.
Nachbedingungen	Empfängeransicht geöffnet. Rechter Editor zeigt entschlüsselte E-Mail an.
Normaler Ablauf	<ol style="list-style-type: none"> 1. „Weiter“-Button wird betätigt. 2. S/MIME-Nachricht wird entschlüsselt. 3. Entschlüsselte E-Mail wird angezeigt.
Alternativer Ablauf	-
Status	Implementiert.

5 Design

Das Plug-in soll die natürliche Umgebung des E-Mail-Client-Benutzers widerspiegeln. Diese besteht aus zwei Teilen: Erstens, aus dem Erstellen und Versenden von E-Mails. Dieser Teil kann auch das Signieren und Verschlüsseln beinhalten. Zweitens, aus dem Empfangen von E-Mails. Empfangene E-Mails können verschlüsselt, signiert oder signiert und verschlüsselt sein. Es bietet sich daher an, beide Abschnitte getrennt voneinander zu behandeln. Das Plug-in wird daher in zwei Teile aufgeteilt, die durch zwei verschiedene Sichten klar unterteilt sind. Die beiden Sichten werden *Senderansicht* und *Empfängeransicht* genannt und nachfolgend beschrieben.

5.1 Senderansicht

Die Senderansicht besteht aus drei Hauptbestandteilen. Einem E-Mail-Editor, einer MIME-Nachricht-Ansicht und einer S/MIME-Nachricht-Ansicht. Diese drei Bestandteile werden gleichzeitig angezeigt und vom Benutzer schrittweise abgearbeitet. Dabei wird der Benutzer langsam von der gewohnten E-Mail-Editor-Schnittstelle, über die Erklärung einer MIME-Nachricht, bis hin zur Erzeugung einer S/MIME-Nachricht geführt werden. Ferner werden optionale und alternative Ablaufmöglichkeiten geboten, die jeder Person der Zielgruppe gerecht werden. Abbildung 5.1 zeigt den ersten Entwurf der Senderansicht. Diese Gestaltung ergab sich aus verschiedenen Aspekten der Benutzbarkeit. Im Hinblick auf die Erlernbarkeit (siehe Abschnitt 3.2.1) wird insbesondere die Vorhersagbarkeit und Generalisierbarkeit gefördert. Abbildung 5.2 zeigt das Aktivitätsdiagramm der Senderansicht. Die Aktivität „Senden“ wird durch das Aktivitätsdiagramm in Abb. 5.3 genauer spezifiziert.

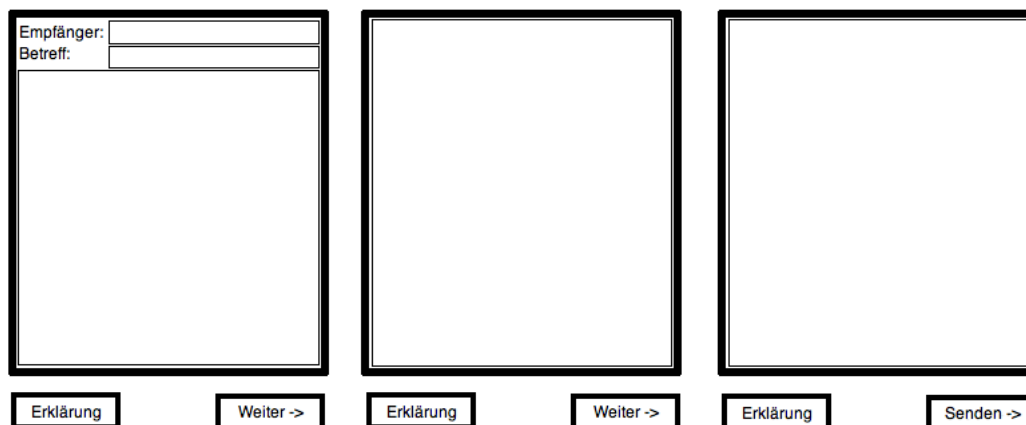


Abbildung 5.1: Erster Entwurf der Senderansicht.

5.1.1 E-Mail-Editor

Die natürliche und bekannte Schnittstelle des E-Mail-Editors ermöglicht den intuitiven Einstieg. Er bietet ein Adressaten-, Betreff- und Texteingabefeld. Für unerfahrene Benutzer wird trotzdem ein Erklärungsbutton angeboten, der einen Dialog mit der Beschreibung des gewünschten Ablaufs öffnet. Der Editor wird links angeordnet, einschließlich eines Erklärungsbuttons und eines Weiter-Buttons, der das Schrei-

ben der E-Mail abschließt und eine MIME-Nachricht erzeugt.

In diesem Abschnitt werden noch keine Konzepte erklärt. Ziel ist der vertraute Einstieg durch die bekannte Schnittstelle.

5.1.2 MIME-Nachricht-Ansicht

Bestätigt der Benutzer per Weiter-Button die Eingabe der E-Mail, wird in der mittleren Ansicht eine gültige MIME-Nachricht dazu erzeugt und dargestellt. Angezeigt werden die Standard *RFC 822*-Header, die MIME-Nachricht mit Content-Type *multipart/mixed* und der im Editor geschriebene Text wird als *Bodypart* mit dem Content-Type *text/plain* erzeugt. Der geschriebene Text wird bereits codiert als *quoted-printable* dargestellt.

Diese Ansicht wird daher genutzt, um den *RFC 822*-Standard und den MIME-Standard grundlegend zu erklären. Beim *RFC 822* bekommt der Benutzer die wichtigsten Bestandteile und die Einschränkungen des Standards (z.B. Beschränkung auf ASCII-Zeichen) erklärt. Beim MIME-Standard sollen insbesondere die Konzepte des Content-Types (Typen und Untertypen), des Content-Transfer-Encoding, der Bodyparts und das Prinzip der Erweiterbarkeit verstanden werden.

Im Anschluss kann aus der MIME-Nachricht eine S/MIME-Nachricht mit Signatur oder Verschlüsselung erstellt werden.

5.1.3 S/MIME-Nachricht-Ansicht

Die dritte Ansicht wird ganz rechts angeordnet und vervollständigt die Senderansicht. Eine erzeugte S/MIME-Nachricht wird in dieser Ansicht angezeigt. Der dazugehörige Erklärungswizard soll sich kontextsensitiv zur Nachricht verhalten. Das heißt, wurde eine S/MIME-Nachricht mit Signatur erzeugt, so beinhaltet der Erklärungswizard Erklärungen zur Signaturerzeugung und -darstellung. Wohingegen Erklärungen zum Verschlüsselungsverfahren erläutert werden, wenn die Nachricht zuvor verschlüsselt worden ist.

Bei einer Signatur wird der Aufbau der CMS-Nachricht in den dafür nötigen Bestandteilen erklärt. Außerdem wird dem Benutzer bewusst gemacht, welche Teile geschützt werden und welche Teile ungeschützt bleiben. Besonders soll auf die signierten Attribute eingegangen werden.

Im Unterschied zur Signatur werden bei einer verschlüsselten Nachricht nur die relevanten CMS-Bestandteile erklärt, die für eine Verschlüsselung notwendig sind. Die Hybrid-Verschlüsselung wird im Allgemeinen erklärt und im Speziellen für S/MIME. Zusätzlich wird noch einmal auf die Kodierung (Base64) eingegangen. Dem Benutzer wird erklärt, weshalb der S/MIME-Teil unleserlich dargestellt wird.

5.2 Dynamische Zertifikat-Erzeugung

Während der Ausführung des Plug-ins werden dynamisch Zertifikate erzeugt. D.h. es wird eine *Certification Authority* (CA) erstellt, durch die der Benutzer ein Benutzerzertifikat ausgestellt bekommt. Hierfür bietet die S/MIME-Implementierung von Flexiprovider ein einfaches Interface.

Wird der E-Mail-Client vom Plug-in zum Versenden einer S/MIME-Nachricht verwendet, kann der Benutzer die notwendigen Zertifikate speichern und sie zur Verifizierung und Entschlüsselung im eigenen Client verwenden (vgl. Aktivitätsdiagramm „Senden“ in Abb. 5.3). Dies wird durch einen einfachen „Speichern Unter“-Button und einem JFace-„Speichern Unter“-Dialog realisiert werden.

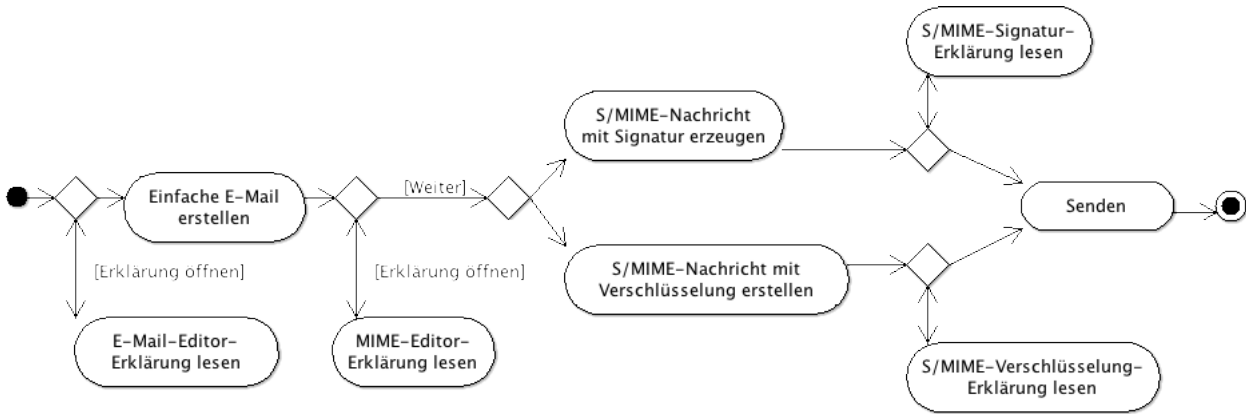


Abbildung 5.2: Aktivitätsdiagramm: Senderansicht.

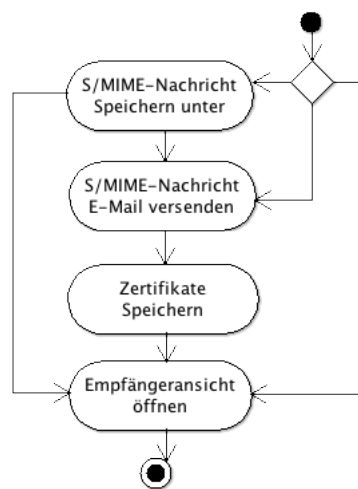


Abbildung 5.3: Aktivitätsdiagramm: Senden.

5.3 Empfängeransicht

Die Empfängeransicht visualisiert die Verarbeitung einer E-Mail aus der Sicht eines Empfängers. Diese Sicht wird nach Abschluss der Senderansicht geöffnet. Zur Visualisierung werden zwei speziellere Ansichten dargestellt. Zum einen die S/MIME-Ansicht, die noch einmal die unveränderte S/MIME-Nachricht auf der Seite des Empfängers darstellt. Zum anderen wird die typische Schnittstelle eines E-Mail-Clients für eine empfangene E-Mail grafisch dargestellt. Der erste Entwurf dieser Ansicht wird in Abbildung 5.4 dargestellt. Ein Aktivitätsdiagramm zur Empfängeransicht ist in Abbildung 5.5 dargestellt.

5.3.1 S/MIME-Ansicht

Die S/MIME-Ansicht zeigt noch mal die erstellte S/MIME-Nachricht an, dieses Mal aber auf Empfängerseite. Sie wird links angeordnet und bildet den ersten Schritt der Empfängeransicht (bzw. den vierten Schritt der Visualisierung).

Der Benutzer wird erfahren, dass seine E-Mail genauso empfangen wird, wie er sie versendet hat. Darüber hinaus wird erklärt, dass bei der Übertragung durch die Server und den Client weitere Header hinzu kommen (z.B. „Received:“).

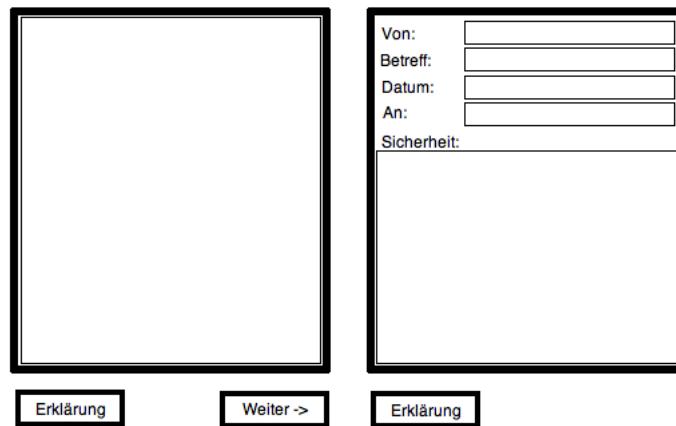


Abbildung 5.4: Erster Entwurf der Empfängeransicht.

5.3.2 E-Mail-Viewer

Die zweite Ansicht stellt einen typischen Viewer dar. Sie besteht aus den häufigen Feldern „Von“, „Betreff“, „Datum“ und „An“. Zusätzlich wird ein Feld mit dem Titel „Sicherheit“ hinzugefügt, um dem Benutzer anzuzeigen, ob die Nachricht signiert wurde oder nicht.

Der Benutzer bekommt in dieser Ansicht vermittelt, wie eine S/MIME-Signatur verifiziert wird, bzw. eine Verschlüsselung entschlüsselt wird.

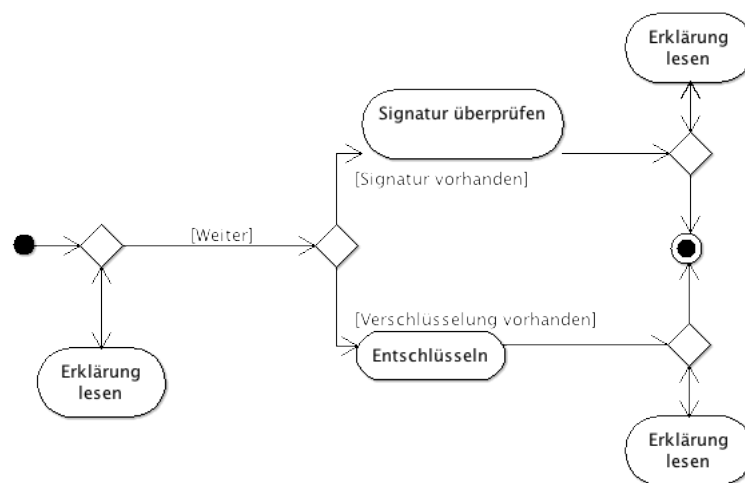


Abbildung 5.5: Aktivitätsdiagramm: Empfängeransicht.

5.4 Erklärungen

Ein sehr wichtiger Bestandteil dieses E-Learning-Projekts sind die Erklärungen. Daher haben mehrere Faktoren Einfluss auf das Design. So haben zunächst die theoretischen Erkenntnisse zum Thema E-Learning aus Abschnitt 3.1 Auswirkungen auf das Design. Zudem die breite Zielgruppe, die angesprochen werden soll. Des Weiteren die Zielsetzungen aus dem Benutzbarkeitsabschnitt (Abschnitt 3.2). Und letztlich die Abhängigkeiten, die durch Eclipse, JCrypTool, SWT und JFace gegeben sind.

5.4.1 Ziele

In Abschnitt 3.1.3 wurde Multimodalität, Multicodalität und Reizüberflutung diskutiert. Es wurden die wissenschaftlichen Erkenntnisse zur Reizüberflutung angegeben. Mögliche Reizüberflutungen lassen sich danach durch Verteilung auf verschiedene Codierung reduzieren und vermeiden. Als Ziel (1) wurde daher definiert: Erklärungen immer auf mehrere Codierungen zu verteilen.

Im Weiteren wurde bereits das Ziel definiert: Prägnante kurze Paragraphen zu schreiben, anstelle von langen theoretischen Texten (vgl. Abschnitt 3.1.3 zur Multicodalität) .

Ein weiterer ausschlaggebender Faktor war die Zielgruppe und die Benutzbarkeit. Die Zielgruppe umfasst viele verschiedene Menschen mit unterschiedlichem Fachwissen und Kenntnisstand (vgl. Abschnitt 4.2). Die Benutzbarkeit kann in Bezug auf die Flexibilität durch einen *Benutzer präventive* Dialog Initiative erhöht werden. Somit wird als Ziel (2) definiert: Alle Erklärungen sollen *Benutzer präventiv* sein. Dies kommt der sehr heterogenen Zielgruppe zugute. Benutzer können selbst entscheiden, ob sie genügend Fachwissen haben oder eine Erklärung durchlesen möchten. Außerdem können dadurch Schritte ausgelassen werden. Führt der Benutzer das Plug-in zweimal durch, kann er z.B. die MIME-Erklärung überspringen und erst bei der S/MIME-Erklärung anfangen.

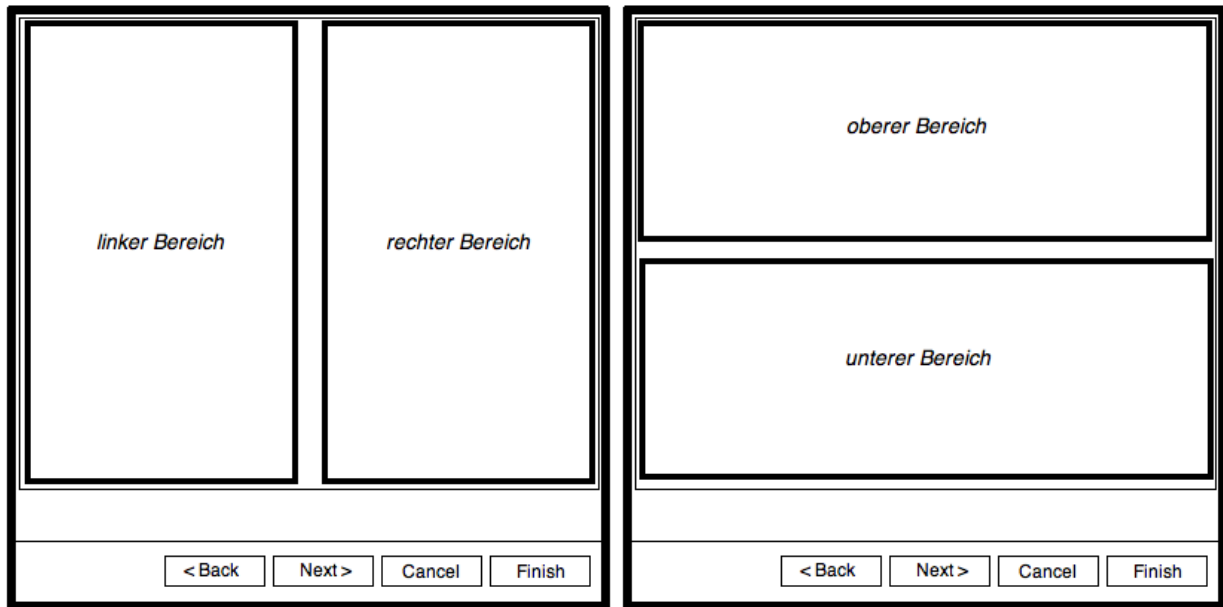
Der nächste Gedanke in Bezug auf die Benutzbarkeit war, die Erlernbarkeit zu fördern. Insbesondere spielen die Erlernbarkeitsprinzipien Synthetisierbarkeit, Bekanntheit, Generalisierbarkeit und Konsistenz eine wichtige Rolle. Daher wurde sich entschieden alle Erklärungen als Dialoge zu gestalten (Ziel (3)), die sich im Aufbau generell sehr ähnlich sein werden (fördert Bekanntheit, Generalisierbarkeit und Konsistenz). Der Aufruf soll außerdem über Erklärungsbuttons erfolgen, die zu jeder Ansicht (d.h. insgesamt fünf) an gleicher Position erscheinen sollen (fördert ebenfalls Bekanntheit, Generalisierbarkeit und Konsistenz). Und schließlich sollen die Erklärungen immer auf die jeweilige Ansicht zugeschnitten sein, das heißt beispielsweise werden zur MIME-Ansicht MIME-bezogene Erklärungen angezeigt. Dadurch weiß der Benutzer immer zu welchem Zustand die jeweilige Erklärung gehört (erhöht insbesondere die Synthetisierbarkeit).

5.4.2 Design

Bereits in Abbildung 5.1 und 5.4 können die konsistent angeordneten Erklärungsbuttons erkannt werden. Dies erfüllt Ziel (2): Alle Dialoge sind durch die optionalen Erklärungsbuttons *Benutzer präventiv*. Abbildung 5.6 zeigt zwei Entwürfe des Erklärungsdialoges die beide angewendet werden sollen. Durch die Gestaltung als Dialog (Wizard) wird Ziel (3) erfüllt.

Ziel (1) wird durch die Unterteilung in zwei Bereiche verwirklicht, die jeweils eine unterschiedliche Codierung des zu lernenden Inhalts haben werden. Dialogentwurf (a) zeigt eine links/rechts Anordnung der Bereiche, Entwurf (b) veranschaulicht eine oben/unten Unterteilung. Einer der beiden Bereiche wird immer Paragraphen mit prägnanten Überschriften enthalten. Der andere Bereich veranschaulicht die geschriebenen Erklärungen durch eine andere Codierung. Beispielsweise könnten links Erklärungen zur MIME-Nachricht sein, während rechts der MIME-Header dargestellt wird.

Die Dialoge (Wizards) sind durch SWT und JFace beeinflusst. Daher haben beide Dialoge garantiert einen „Cancel“- und einen „Finish“-Button, sowie optional (falls es mehrere Seiten gibt) einen „Back“- und einen „Next“-Button.



(a) links/rechts Anordnung

(b) oben/unten Anordnung

Abbildung 5.6: Dialogentwurf mit verschiedenen Anordnungen.

5.5 E-Mail-Client-Konfiguration

Bestandteil des Plug-ins soll ein einfacher E-Mail-Client sein. Dieser ermöglicht dem Benutzer erstellte S/MIME-Nachrichten zu versenden und diese dann anschließend im eigenen E-Mail-Client zu empfangen, zu überprüfen und zu entschlüsseln. Zur Realisierung wird die persistente Speicherung der Konfigurationsdaten ermöglicht. Eclipse bietet die Möglichkeit Einstellungsdaten (Preferences) persistent für ein Plug-in zu speichern. Zu diesem Zweck wird eine neue PreferencePage in den Einstellungsdialog von JCrypTool (vgl. Abb. 2.1) integriert werden. Die Kategorie „Visualisierungen“ bekommt die Unterkategorie „S/MIME“. Durch die dazugehörige PreferencePage können E-Mail-Adresse, Name, Ausgangsserver, Port und Benutzername angegeben werden. Das Passwort wird nicht persistent gespeichert, da eine geschützte Speicherung nicht garantiert werden kann. Der Benutzer wird vor dem Senden zur Passworteingabe aufgefordert.

6 Umsetzung

Dieses Kapitel beschreibt die Umsetzung des Plug-ins. In den folgenden Abschnitten werden die wichtigsten Implementierungen präsentiert und erklärt.

6.1 Integration in JCrypTool

Das Plug-in wurde analog zu den anderen Visualisierungen in JCrypTool integriert. Die Integration geschieht in zwei Schritten:

Zunächst muss eine neue *View* implementiert werden. Zu diesem Zweck stellt Eclipse die abstrakte Klasse `org.eclipse.ui.part.ViewPart` zur Verfügung. Sie beinhaltet die Basis-Implementierung für *Workbench Views*. Durch den Eclipse Extension Point `org.eclipse.ui.views` wird die implementierte Klasse in das Plug-in eingebunden.

Anschließend muss die Möglichkeit geboten werden, die *View*, d.h. das Plug-in aufzurufen. Bei einer Visualisierung geschieht dies durch den Menüpunkt „Visualisierungen“. Hierfür stellt JCrypTool den Extension Point `org.jcryptool.core.operations.visuals` bereit. Analog können auch andere Plug-ins eingebunden werden. Wird z.B. ein neues Spiel entworfen und implementiert, wird der Extension Point `org.jcryptool.core.operations.games` verwendet, um es in das Spiele-Menü zu integrieren.

Abbildung 6.1 zeigt das Visualisierungsmenü von JCrypTool mit dem Menüpunkt „S/MIME“ für das entworfene Plug-in.

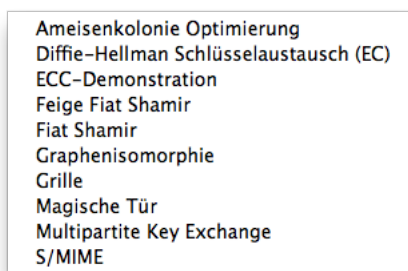


Abbildung 6.1: Visualisierungsmenü mit integriertem S/MIME-Plug-in.

6.2 Senderansicht

Die grafische Implementierung der Senderansicht ist in Abbildung 6.2 zu sehen. Deutlich ist die Ähnlichkeit zum ersten Entwurf zu erkennen (vgl. Abb. 5.1). Die Abbildung zeigt das Plug-in bereits in Aktion. So wurde zunächst eine E-Mail geschrieben (links). Anschließend wurde die E-Mail in eine MIME-Nachricht umgewandelt (Mitte). Und schließlich wurde die MIME-Nachricht signiert und als S/MIME-Nachricht angezeigt (rechts).

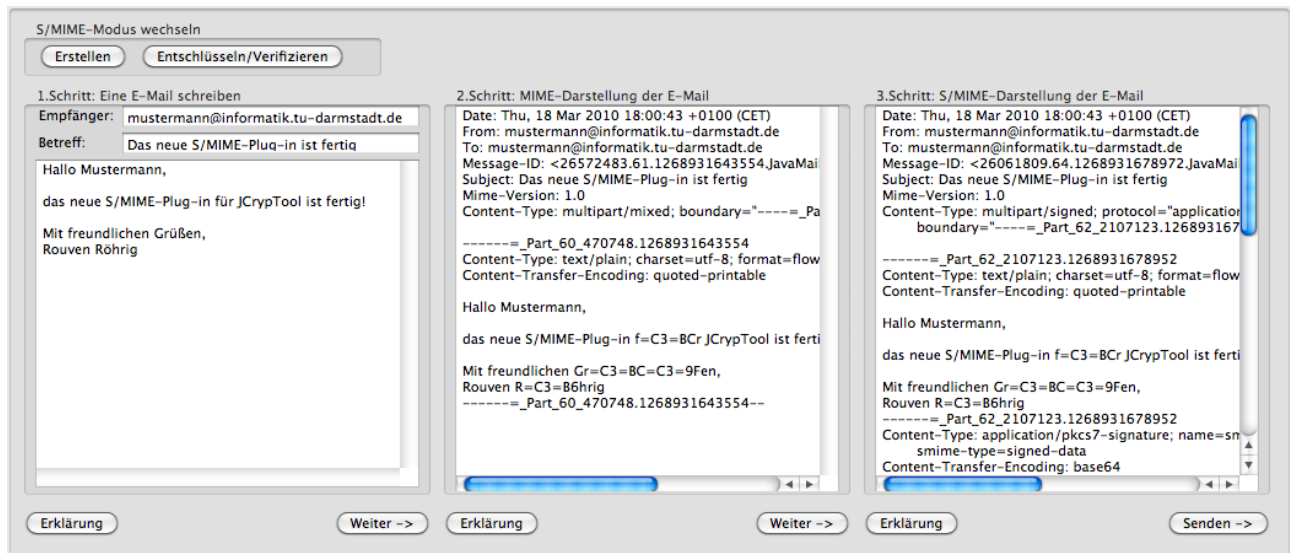


Abbildung 6.2: Grafische Implementierung der Senderansicht (Mac OS X 10.6).

Die fertige Implementierung unterscheidet sich jedoch in zwei Punkten. Zum einen wurden zwei Buttons eingefügt, um problemlos zwischen der Sender- und Empfängeransicht zu wechseln (vgl. Abbildung 6.3). Es wurde sich dafür entschieden dieses Element als „S/MIME-Modus wechseln“ zu bezeichnen, anstelle von „Ansicht wechseln“. Der Grund dafür ist, dass der Benutzer zwar intuitiv erkennen kann in welcher Sicht er sich befindet, nicht aber wo er S/MIME-Nachrichten erstellen kann und auch nicht wo er sie entschlüsseln/verifizieren kann. Durch die Kennzeichnung als „S/MIME-Modus“ wird dieses Problem gelöst und zusätzlich eine Brücke zwischen Anwendersicht und Implementierungssicht erstellt. Die Knöpfe wurden mit „Erstellen“ und „Entschlüsseln/Verifizieren“ bezeichnet.

Zum anderen wurde jedes Editor-Fenster nummeriert und beschriftet. Dies wirkt sich unter mehreren Gesichtspunkten positiv auf die Benutzbarkeit aus. Im Entwurf wurde bereits das hohe Maß an Vorhersagbarkeit und Generalisierbarkeit beschrieben (vgl. Abschnitt 5.1). Die Nummerierung und Beschriftung wirkt sich insbesondere auf die *Synthetisierbarkeit* aus, da der Benutzer zuvor nicht genau wusste, in welchem Schritt er sich gerade befindet (z.B. „MIME-Darstellung der E-Mail“). Dies verbessert ebenfalls die *Vorhersagbarkeit*, der Benutzer weiß bereits vorher, was er im nächsten Schritt erhalten wird.

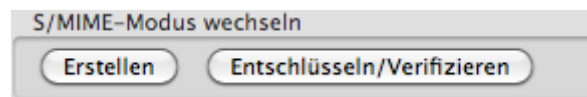


Abbildung 6.3: S/MIME-Modus-Menü (Mac OS X 10.6).

6.3 Empfängeransicht

Abbildung 6.4 zeigt die grafische Umsetzung der Empfängeransicht. Auch diese Ansicht unterscheidet sich in ihrer Implementierung nur durch die Beschriftungen der S/MIME-Ansicht und des E-Mail-Viewers, sowie durch das Element „S/MIME-Modus wechseln“ (vgl. Entwurf in Abb. 5.4). Die Abbildung zeigt die Empfängeransicht im Betrieb. Es ist eine verschlüsselte Nachricht in der S/MIME-Ansicht zu erkennen und diese Nachricht wieder entschlüsselt, befindet sich auf der rechten Seite im E-Mail-Viewer.

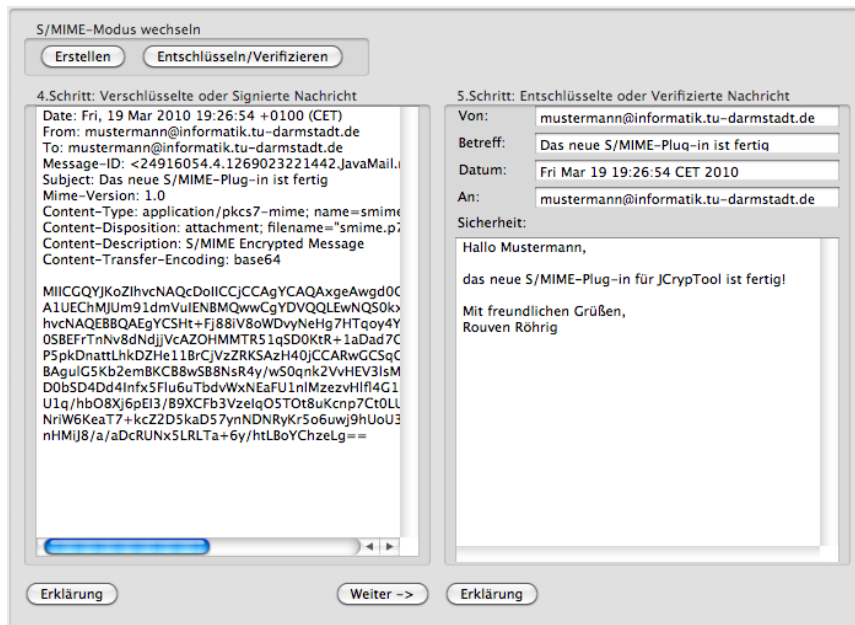


Abbildung 6.4: Grafische Implementierung der Empfängeransicht (Mac OS X 10.6).

6.4 Konfiguration des E-Mail-Clients

Analog zur Planung in Abschnitt 5.5 wurde die neue Unterkategorie „S/MIME“ in die dafür vorgesehene Kategorie „Visualisierungen“ des Einstellungsdialoges hinzugefügt. Die dazugehörige PreferencePage bietet eine Schnittstelle für die Konfigurationsfelder E-Mail-Adresse, Name, Ausgangsserver, Port und Benutzername. Abbildung 6.5 zeigt die nahtlose Integration in den JCryptool-Einstellungsdialog.

6.5 Erklärungsdialoge

Die Implementierung der Erklärungsdialoge wird anhand einiger Beispiele beschrieben. Dies ist durch hohe Konsistenz möglich. Die Dialoge weisen durchgehend den im Entwurf beschriebenen Aufbau auf, der eine Aufteilung in zwei Bereiche vorsieht (vgl. Abb. 5.6). Abbildung 6.6 zeigt den Erklärungsdialog für MIME-Nachrichten. Zu erkennen sind die *Back*- und *Next*-Buttons, die dem Benutzer signalisieren, dass es sich um mehrere Seiten handelt. Im Speziellen wird die Erklärung für den MIME-Header angezeigt. Daher wird auf der rechten Seite der Header der erstellten MIME-Nachricht genutzt und durch verschiedene Farben hervorgehoben. Auf der linken Seite befinden sich kurze Paragraphen mit prägnanten Überschriften. Die Überschriften erklären ein Element aus dem MIME-Header und tragen die gleiche Farbe, wie der dazugehörige Bestandteil im Header. Beispielsweise hat *RFC 822* die gleiche Farbe wie *From:*, *To:*, *Subject:* und *Date:*. Die Farben haben eine nichtfunktionale Eigenschaft. Sie helfen Lernenden durch eine offensichtlichere Zuordnung. Gleichzeitig haben sie keine notwendige Funktion und stellen daher kein Hindernis für farbenblinde Menschen dar.

In Abbildung 6.7 wird der Screenshot des Erklärungsdialoges zur *SignedData*-Struktur vom CMS abgebildet. Dieser Dialog verwendet eine oben/unten Anordnung (vgl. Abbildung 5.6 (b)). Auch in dieser Erklärung werden die zu erklärenden Konzepte durch zwei Codierungen dargestellt (vgl. Multicodalität, Abschnitt 3.1.3). Darüber hinaus wird eine Verbindung der beiden Codierungen durch die farbigen Überschriften und farbigen Bestandteile der *SignedData*-Struktur hergestellt.

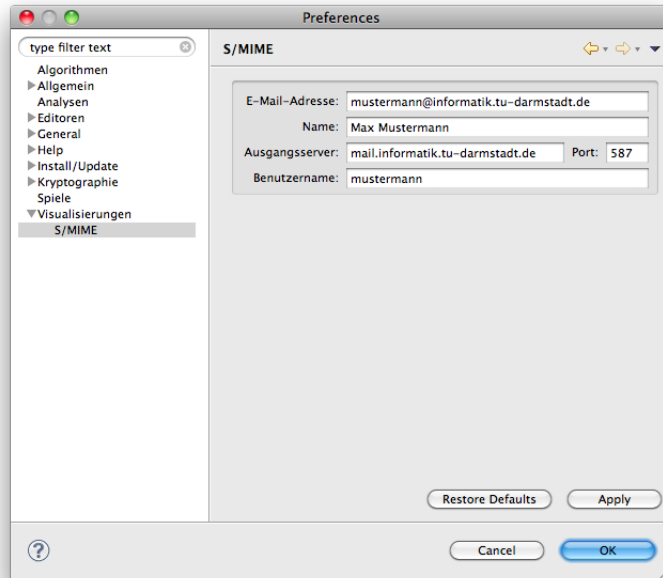


Abbildung 6.5: Einstellungsdialog mit S/MIME-Einstellungsseite (Mac OS X 10.6).

Wirft man einen genaueren Blick auf Abbildung 6.6 und 6.7, erkennt man, dass der *Back*-Button einmal aktiv und einmal inaktiv dargestellt wird. Dieser Mechanismus ist durch die Verwendung von JFace gegeben und soll dem Benutzer sagen, ob weitere Seiten vorhanden sind und in welcher Richtung sich diese befinden. Dieses Verhalten hat einen positiven Effekt auf die Vorhersagbarkeit und Synthetisierbarkeit in Bezug auf die Erlernbarkeit.

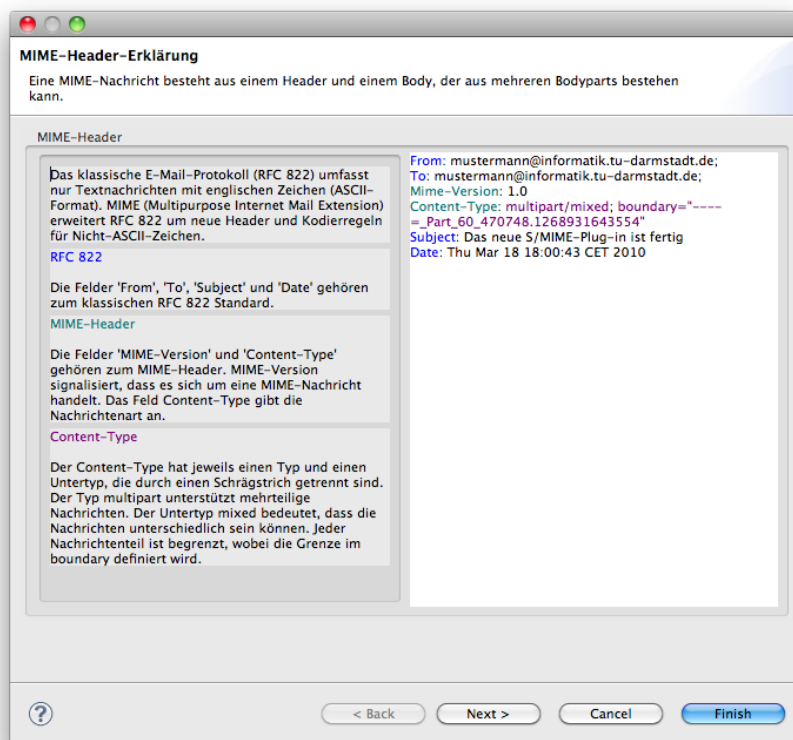


Abbildung 6.6: Wizard mit der Erklärung für den MIME-Header (Mac OS X 10.6).

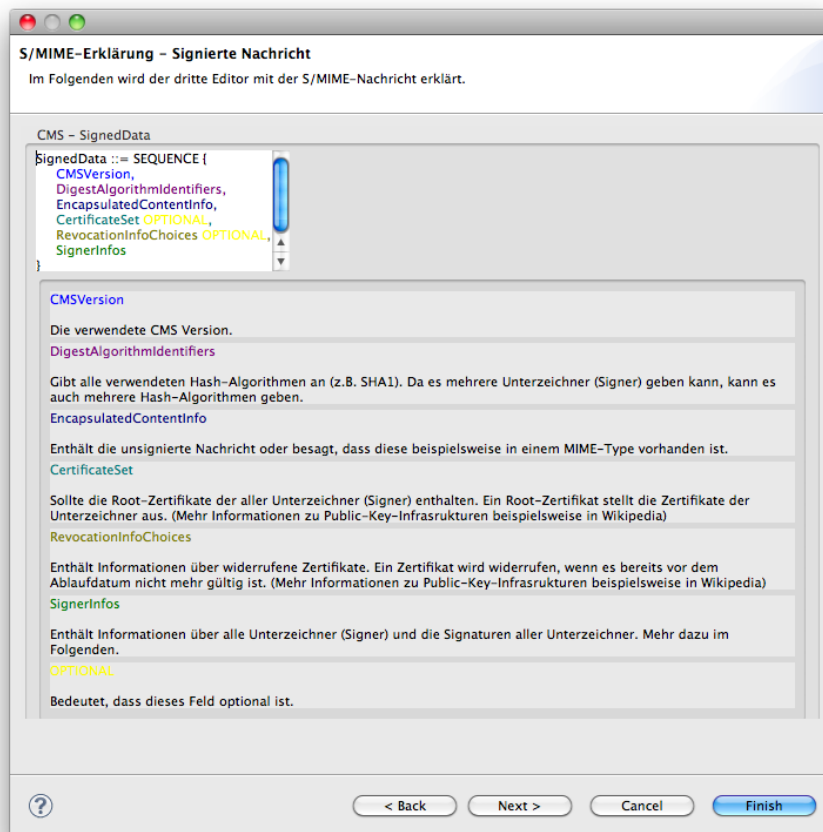


Abbildung 6.7: Wizard mit der Erklärung zu SignedData vom CMS (Mac OS X 10.6).

7 Fazit und Ausblick

Das Plug-in zu dieser Arbeit konnte in einer guten Version erfolgreich fertig gestellt werden. Alle geplanten Konzepte wurden implementiert. Durch die ausführliche Diskussion der Benutzbarkeitsprinzipien (Abschnitt 3.2) und dem Begriff des E-Learnings und dessen Konzepte (Abschnitt 3.1) konnte ein hohes Maß an Benutzerfreundlichkeit erreicht werden und viele Aspekte einer guten E-Learning-Software realisiert werden. Darüber hinaus wurde auf unterschiedliche Ziel- und Interessengruppen geachtet, so dass dieses Plug-in vielen Menschen gerecht wird. Im Speziellen wurden die unterschiedlichen Fähigkeiten und Fertigkeiten der heterogenen Zielgruppe berücksichtigt und dadurch die Möglichkeit des motivierenden individuellen Lernens geschaffen.

Der Abschluss dieser Arbeit wird nicht der Abschluss dieses Projektes sein. Im nächsten Schritt wird das Plug-in vorbereitet, um ein offizielles JCrypTool-Plug-in zu werden, so dass es in naher Zukunft für alle JCrypTool-Nutzer verfügbar sein wird.

Das Plug-in realisiert bereits viele E-Learning-Konzepte, trotzdem wurden einige Erweiterungsmöglichkeiten angesprochen. So ist es möglich weitere Elemente der Interaktivität einzubauen, neue Codierungen zu nutzen und multimodale Bestandteile zu implementieren. Ferner könnten sich zukünftige Versionen dieses Plug-ins mit Angriffen auf S/MIME beschäftigen. Die aktuelle Version zeigt bereits die Schwachstellen auf, wie beispielsweise die ungeschützten RFC 822-Header und die Austauschmöglichkeit einer verschlüsselten Nachricht, die nicht durch eine Signatur geschützt wurde. Daher bietet das Plug-in viel Potenzial für weitere Arbeiten.

8 Literaturverzeichnis

Schiele, B. (2007).

Human Computer Systems, Wintersemester 2007/08, Stand 25.11.2009

http://tahiti.mis.informatik.tu-darmstadt.de/oldmis/Education/Courses/hcs/index_html/index.html

Issing, L. J. und Klimsa, P. (Hrsg.) (2002).

Information und Lernen mit Multimedia und Internet: Lehrbuch für Studium und Praxis.

Weinheim 3. Auflage Weinheim: BeltzPVU.

Rey, G. D. (2009).

E-Learning. Theorien, Gestaltungsempfehlungen und Forschung. Webseite zum Buch unter:

<http://www.elearning-psychologie.de/literatur.html>, letzter Zugriff: 12.03.2009.

Schaumburg, H. und Issing, L. J. (2009). Interaktives Lernen mit Multimedia.

In: G. D. Rey. E-Learning. Theorien, Gestaltungsempfehlungen und Forschung. Webseite zum Buch unter:

<http://www.elearning-psychologie.de/>, letzter Zugriff: 12.03.2009.

Weidenmann, B. (2009).

Multicodierung und Multimodalität im Lernprozeß.

In: G. D. Rey, E-Learning. Theorien, Gestaltungsempfehlungen und Forschung. Webseite zum Buch unter:

<http://www.elearning-psychologie.de/>, letzter Zugriff: 12.03.2009.

Bétrancourt, M. (2009).

The animation and interactivity principles in multimedia learning.

In: R. E. Mayer (Hrsg.), The Cambridge Handbook of Multimedia Learning (S. 287-296).

Cambridge, MA: Cambridge University Press.

In: G. D. Rey, E-Learning. Theorien, Gestaltungsempfehlungen und Forschung. Webseite zum Buch unter:

<http://www.elearning-psychologie.de/>, letzter Zugriff: 12.03.2009.

Ballstaedt, S.-P. (2002).

Integrative Verarbeitung bei audiovisuellen Medien.

In: K. Böhme-Dürr, J. Emig & Seel (Hrsg.), Wissensveränderung durch Medien, (189-196). München: Saur.

In: Ludwig J. Issing und Paul Klimsa (Hrsg.).

Information und Lernen mit Multimedia und Internet: Lehrbuch für Studium und Praxis.

3. Auflage. Weinheim: BeltzPVU.

Engelkamp J. und Zimmer, H.D. (2002).

The Human Memory. A Multi-Modal Approach.

Bern: Hogrefe & Huber Publishers.

In: L. J., Issing und P. Klimsa (Hrsg.).

Information und Lernen mit Multimedia und Internet: Lehrbuch für Studium und Praxis (S.53-54).

3. Auflage. Weinheim: BeltzPVU.

Tanenbaum, A. S. (2003).

Computernetzwerke (S.647-654).

4. überarbeitete Auflage. München: Person Studium.

International Business Machines Corp. (2006).

Eclipse Platform Technical Overview

<http://eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf>

letzter Zugriff: 01.03.2010.
Crocker, D. H. (August 1982).
RFC 822.
Standard for the Format of ARPA Internet Text Message.
<http://tools.ietf.org/html/rfc822>, letzter Zugriff: 15.02.2010.

Ramsdell, B. (Januar 2010).
RFC 5751.
Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification.
Internet Engineering Task Force (IETF): <http://tools.ietf.org/html/rfc5751>, letzter Zugriff: 09.03.2010.

Ramsdell, B. (Juli 2004).
RFC 3851.
Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification.
Network Working Group: <http://tools.ietf.org/html/rfc3851>, letzter Zugriff: 09.03.2010.

Housley, R. (September 2009).
RFC 5652.
Cryptographic Message Syntax (CMS).
Network Working Group: <http://tools.ietf.org/html/rfc5652>, letzter Zugriff: 09.03.2010.

Von Loewenich, C. und Werner, J. (2009).
Corporate Design der TU Darmstadt: LaTeX-Vorlage.
<http://exp1.fkp.physik.tu-darmstadt.de/tuddesign/>, letzter Zugriff: 28.12.2009