Autoren: Graber Johnny, Müdespacher Andreas
Thema: Sicherheitstool für USB-Sticks
Datum: 15. Dezember 2006

geht an: Dubois Jean-Paul
Fuhrer Claude
Voirol Pierre-Yves

Endbericht

AtaraxiS

Dok.-Nr:	3
Version:	1.0
Status:	Abgeschlossen
Klassifizierung:	intern
Ausgabedatum:	15. Dezember 2006
Projektbeginn:	23. Oktober 2006
Projektende:	15. Dezember 2006
Dok.-Name:	Endbericht_AtaraxiS.pdf
SVN-Ablage:	/documents/

1 Inhaltsverzeichnis

2 Zweck des Dokuments

Der Endbericht liefert einen Überblick über die in der Diplomarbeit geleistete Arbeit sowie Erklärungen für die ausgewählten Technologien. Er liefert zudem eine detaillierte Erklärung zur implementierten Lösung und zeigt auf, wo und wie sich die Semesterarbeit von der Diplomarbeit abgrenzt.

3 Sinn und Zweck von AtaraxiS

Durch die stetige Verbreitung von USB-Sticks werden diese auch immer mehr für den Datentransport eingesetzt. Ihre Grösse macht sie nicht nur sehr angenehm zum mitführen, sondern auch zum verlieren. Je wichtiger die Daten auf dem USB-Stick, desto gravierender der Schaden bei deren Verlust.

Das verlieren eines USB-Stick kann man mit Software nicht verhindern. Wohl aber kann man die Daten schützen. Genau hier setzt AtaraxiS an. AtaraxiS bietet einem die Möglichkeit, mit einem weit verbreiteten und sicheren Verfahren (AES - American Encryption Standard) seine Daten für den Transport zu verschlüsseln.

AtaraxiS kann direkt auf dem USB-Stick abgelegt werden und kommt bis auf Java ohne zusätzliche Software aus. Java selbst kann ebenfalls auf dem USB-Stick abgelegt werden, was AtaraxiS unabhängig von der Software auf dem Computer macht.
Neben der Ver- und Entschlüsselung von Dateien und Verzeichnissen bietet AtaraxiS auch Möglichkeiten, Daten sicher zu löschen und bietet über dies hinaus eine integrierte Passwortverwaltung an. Eine einfache Benutzerführung, die Mehrbenutzer-Fähigkeit und die Unterstützung der Sprachen Deutsch, Englisch und Französisch runden AtaraxiS ab.

Alles Notwendige für den sicheren Transport der eigenen Daten wird mit AtaraxiS mitgeliefert. Für den Datenaustausch zwischen mehreren Benutzern bietet AtaraxiS derzeit nur eine indirekte Möglichkeit über ein gemeinsam genutztes Benutzerkonto.

Die Passwort Verwaltung und der Datenschredder sind 2 Teile, die auch von Benutzern ohne USB-Stick häufig nachgefragt werden. AtaraxiS muss nicht zwingend auf einem USB-Stick installiert werden, sondern kann auch auf dem lokalen Computer, einer mobilen Festplatte oder allen anderen Datenträgern benutzt werden.

4 Zusammenfassung der geleisteten Arbeit

Die Diplomarbeit AtaraxiS setzt auf der gleichnamigen Semesterarbeit auf. In der Semesterarbeit wurden die für die Verschlüsselung und das komplette Schlüssel-Management notwendigen Grundlagen erarbeitet und ein einfacher Prototyp einer grafischen Oberfläche (GUI) erstellt.

In der Diplomarbeit wurden diese Grundlagen ergänzt und mit zusätzlichen Funktionen erweitert. Neben Dateien können nun ganze Verzeichnisse verschlüsselt werden. Dateien und Verzeichnisse lassen sich vor dem Verschlüsseln komprimieren. Eine Möglichkeit zum Verwalten seiner Passwörter wurde neu erstellt, ebenso ein Passwort-Generator und ein Datenschredder zum sicheren löschen von Dateien und ganzen Verzeichnissen. Um all diese Funktionen dem Benutzer zur Verfügung zu stellen wurde ein komplett neues GUI entwickelt. AtaraxiS wurde vom Einbenutzer-System zu einem Multibenutzer-System umgebaut und ist durchgehend mehrsprachig.
Neben diesen für den Benutzer sichtbaren Erweiterungen gibt es zusätzlich auch weniger offensichtliche Hilfsfunktionen. Eine im Hintergrund laufende Backup-Funktion hilft dem Benutzer, falls er seinen AES-Schlüssel verliert. Eine Lösung zum automatischen kopieren von Policy-Dateien, um unter Java die volle Stärke der Verschlüsselung zu aktivieren, befreit den Benutzer davor, sich selber die Dateien zu besorgen und einzurichten.

Mit diesen grösseren und unzähligen kleineren Erweiterungen haben wir am Ende der Diplomarbeit nun eine Lösung, die für den täglichen Gebrauch eingesetzt werden kann.

5 Projektplanung

5.1 Entwicklungsmodell

Das gewählte Spiralmodell hat sich für die Diplomarbeit bewährt. Durch die Erfahrungen aus der Semesterarbeit konnten wir das Modell an unsere Anforderungen anpassen und ohne grossen Verwaltungsaufwand entwickeln.

5.2 Projektziele

Alle als Muss-Ziel definierten Aufgaben konnten in der Diplomarbeit erstellt werden. Von den Kann-Zeilen wurde allerdings keines realisiert.

Iteration	Ziel	Art	Erreicht
1.1	Mehrsprachiges GUI	Muss	Ja
1.2	Taskbar-Icon mit den wichtigsten Funktionen	Muss	Ja
2.1	Passwort- und Accountverwaltung	Muss	Ja
2.2	Passwort-Generator	Muss	Ja
3	Datenschredder (sicheres Löschen von Dateien)	Muss	Ja
4.1	Funktion zur Verschlüsselung von Verzeichnissen	Muss	Ja
4.2	Komprimierte Verschlüsselung von Dateien und Verzeichnissen	Muss	Ja
5	Einfaches Deployment-Verfahren für Endbenutzer	Muss	Ja
6.1	Public-/Private-Key Verschlüsselung für Texte / E-Mails	Kann	Nein
6.2	Import- / Exportmöglichkeit für Public-Key Zertifikate	Kann	Nein
7.1	Hashbasierte Dateiverifikation	Kann	Nein
7.2	Kontrolle des Ver- und Entschlüsselungsvorgangs	Kann	Nein
7.3	Einbinden von portabler Applikationen	Kann	Nein

Es hat sich im Verlauf der Diplomarbeit gezeigt, dass ein Einbinden von portablen Applikationen zu wenig Vorteile für den Benutzer bringt und zurzeit nicht umsetzbar ist. Zu einem späteren Zeitpunkt und nach einer Konsolidierung im Bereich der portablen Applikationen sollte man diesen Entscheid überdenken.

Die anderen Kann-Ziele werden im Kapitel Ausbaumöglichkeiten genauer betrachtet und könnten als nächste Schritte angegangen werden.

5.3 Zeitplanung

Die Qualität unserer Aufwandsschätzung hat sich gegenüber der Semesterarbeit deutlich verbessert. Für das GUI benötigten wir mehr Zeit, ebenso für den Datenschredder und die Integration der Passwortverwaltung ins GUI. Durch eine parallele Bearbeitung des Datenschredders und der Verschlüsselung von Verzeichnissen konnte dieser Rückstand aber aufgeholt werden.

5.4 Anmerkung zum Projektmanagement

In der Projektmitte sollte man eine Kontroll- und Aufräum-Iteration einbauen. In dieser Iteration sollte man sich die Zeit nehmen, alles genau zu überprüfen und alle Kleinigkeiten abzuschliessen, die man bisher immer auf später verschoben hat.

6 Übersicht zur implementierten Lösung

6.1 Grundlegende Technologien

In der Semesterarbeit haben wir die Entscheidungen für die grundlegenden Technologien getroffen. Hier folgt daher nur eine kurze Begründung. Die Evaluationsdokumente der Semesterarbeit liefern weiterführende Angaben, falls diese gewünscht sind.

6.2 Einsatz von Java

Wir haben uns für Java entschieden, da unser Programm unabhängig vom Betriebssystem laufen soll. Java ist zu diesem Zweck die ideale Sprache.

6.3 Verschlüsselung auf Dateiebene

Die Verschlüsselung kann auf unterschiedlichen Ebenen stattfinden. Wir wollten ein Plattform- und USB-Stick-unabhängiges Tool entwickeln, daher erschien uns die Verschlüsselung auf der Ebene der Dateien am sinnvollsten.

6.4 Verschlüsselungsalgorithmus AES

AES steht für American Encryption Standard und ist der Nachfolger von DES. AES setzt sich immer mehr durch und wird mittlerweile von fast allen Verschlüsselungs-Tools unterstützt.

6.5 Security Provider Bouncy Castle und KeyStore-Typ UBER

Die Auswahl an frei verfügbaren Security Providern ist sehr beschränkt und bei der Evaluierung zeigte sich, dass nur Bouncy Castle unseren Anforderungen genügt. Durch diese Wahl standen uns fünf KeyStore-Typen zur Auswahl. Wir wählten den UBER KeyStore, da nur dieser auch die Aliasse und Zertifikatstexte verschlüsselt auf die Festplatte abspeichert.

6.6 Verwendete Bibliotheken

Name	Version	Lizenz/Einschränkung	Webseite
Bouncy Castle	1.34	Keine Einschränkung	http://www.bouncycastle.org
Log4J	1.2.14	Apache License V. 2	http://logging.apache.org/log4j/docs/
JDOM	1.0	Apache-style Open Source Lizenz	http://jdom.org
Jaxen	1.1 B9	Apache-style Open Source Lizenz	http://jaxen.org
SWT	3.2.1	Eclipse Public License (EPL)	http://www.eclipse.org/swt/

Zur Verwendung der Bibliotheken in AtaraxiS waren keine Modifikationen notwendig. Bei allen gewählten Bibliotheken können wir frei entscheiden, wie wir AtaraxiS lizenzieren. Diesbezüglich gibt es keine Einschränkungen. Die Einschränkungen der Lizenzen kommen erst dann zu tragen, wenn wir die Bibliotheken modifizieren oder unter unserem Namen herausgeben wollten.

Um eine kommerzielle Version von AtaraxiS zu vertreiben, muss man bei einigen der Lizenzen auf Verlangen der Benutzer den Quellcode der Bibliothek zur Verfügung stellen. Um sich die allfällige Arbeit zu ersparen sollte man am Besten den Quellcode aller Bibliotheken gleich von Anfang an auf der CD mitliefern oder auf der Webseite zum Download anbieten.

6.7 Verwendete Icons und Grafiken

Name	Lizenz	Download
Crystal SVG	LGPL	http://www.kde-look.org
Linspire Clear (for Debian)	LGPL	http://www.kde-look.org

Die GNU Lesser General Public License (LGPL) erlaubt im Gegensatz zur GNU General Public License (GPL) das einbinden in proprietäre (Closed Source) Software. Somit sind wir auch bezüglich der Icons und Grafiken frei in der Wahl unserer Lizenz für AtaraxiS.

6.8 Abgrenzung zur Semesterarbeit

6.8.1 Unveränderte Klassen und Interfaces
Ohne Änderungen wurden aus der Semesterarbeit übernommen:

- crypt.KeyCreator
- crypt.NotImplementedException
- crypt.CryptoMethodError
- gui.IllegalPasswordException

6.8.2 Geringfügig veränderte Klassen und Interfaces
Diese Klassen wurden für das Logging oder den Wechsel des Passwort-Typs (String/char[]) editiert:

- crypt.AESKeyCreator
- crypt.KeyStoreHandler
- crypt.RSAKeyCreator
- crypt.UBERKeyStoreCreator
- crypt.UBERKeyStoreHandler

6.8.3 Stark veränderte Klassen und Interfaces
Durch zusätzliche Verschlüsselungsarten (komprimiert, Verzeichnisse) und weiteren Anforderungen (fehlende Policy-Rechte entdecken) wurde diese Klasse stark erweitert:

- crypt.AtaraxisCrypter

6.8.4 Neue Klassen und Interfaces
Neu erstellt wurden diese Klassen und Interfaces:

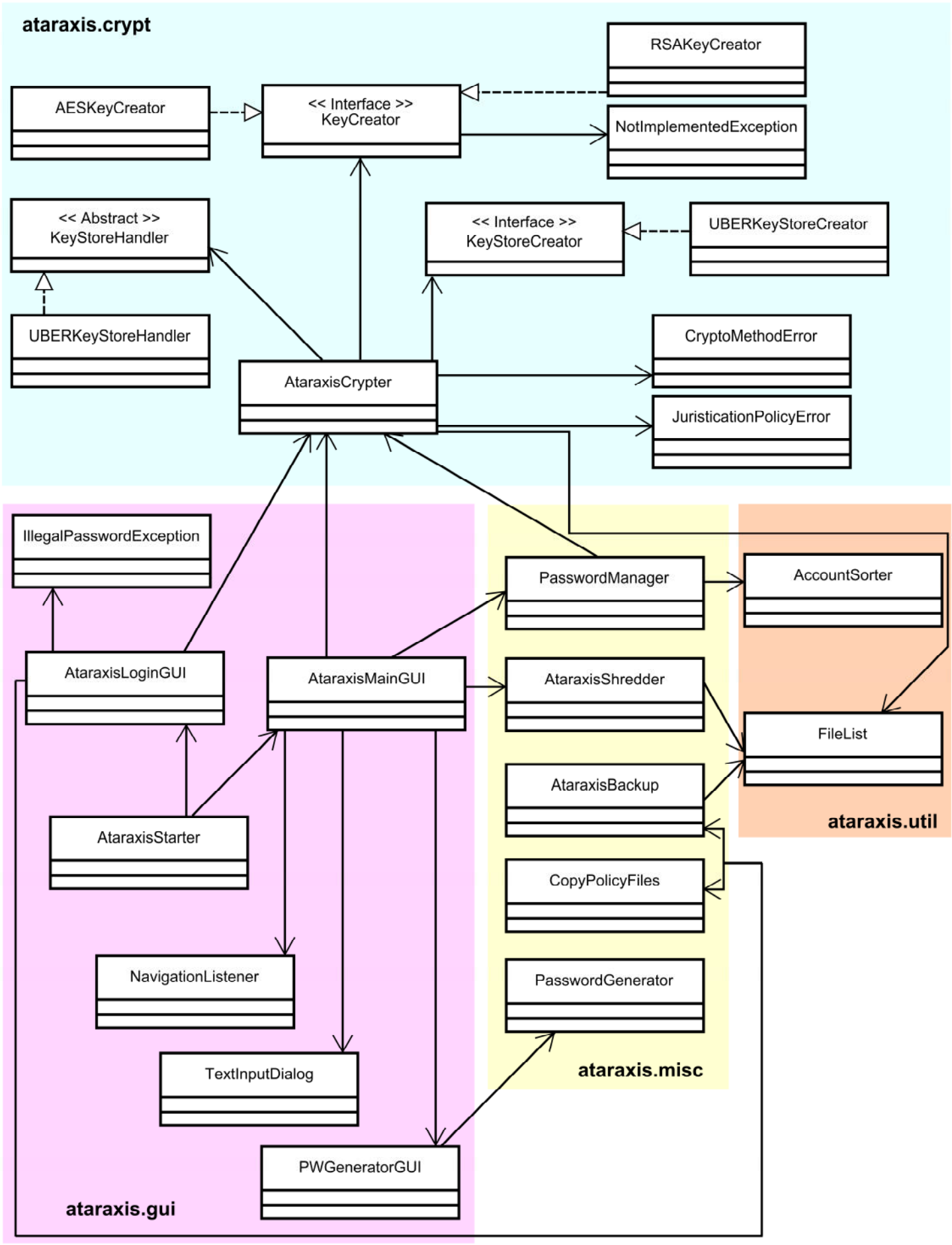
- crypt.JurisdictionPolicyError
- gui.AtaraxisLoginGUI
- gui.AtaraxisMainGUI
- gui.AtaraxisStarter
- gui.NavigationListener
- gui.PWGeneratorGUI
- gui.TextInputDialog
- misc.AtaraxisBackup
- misc.AtaraxisShredder
- misc.CopyPolicyFiles
- misc.PasswordGenerator
- misc.PasswordManager
- util.AccountSorter
- util.FileList

6.8.5 Entfernte Klassen und Interfaces
Keine weitere direkte Verwendung gab es für diese Klassen:

- cli.AtaraxisCLI
- gui.AtaraxisGUI

Klassendiagramm AtaraxiS

ch.bfh.hti.projects.ataraxis.*



7 Das GUI von Ataraxis

7.1 Anforderungen gemäss Pflichtenheft

Der Benutzer soll die Sprache des GUI auswählen können. Die Sprachänderung muss spätestens beim nächsten Neustart des Programms aktiviert sein. Sprachen: Deutsch, Englisch, Französisch

Über ein Taskbar-Icon sollen diese Bereiche des GUI direkt aufgerufen werden können:
Passwort/Accountverwaltung, Datenverschlüsselung, Konfiguration

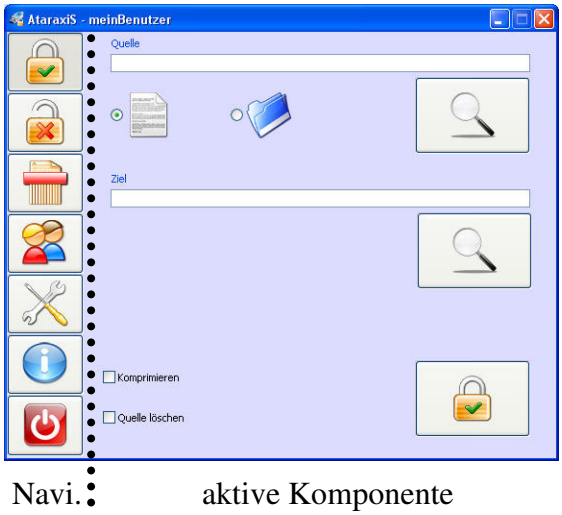
7.2 Lösungssuche für das GUI

Da wir bei der Semesterarbeit mit SWT bereits vorwiegend positive Erfahrungen gemacht haben und wir zudem gerne ein Tray Menu für unsere Applikation anbieten wollten, haben wir uns für SWT entschieden. Im Verlaufe der Diplomarbeit haben wir herausgefunden, dass mit Java 6 neuerdings auch Tray Menus mit AWT und Swing erzeugt werden können. Java 6 wurde allerdings erst wenige Tage vor dem Ende der Diplomarbeit veröffentlicht.

7.3 Implementation des GUI

Zuerst haben wir die Navigation und eine Komponente möglichst übersichtlich und ansprechend entworfen. Diese Arbeit dauerte einige Zeit, da wir uns zuerst über das grundsätzliche Layout und die Anzahl der Komponenten unterhalten mussten. Als wir dann eine Komponente implementiert hatten (vorerst ohne Ereignissteuerung), haben wir diese Komponente kopiert und den Bedürfnissen der neuen Funktionalität angepasst. So hatten wir ein GUI-Skelett, bei dem die meisten Komponenten bereits zu sehen waren.
Nun machten wir uns daran, je nach Fortschritt der technischen Klassen (Verschlüsselung, Schredder usw.) diese in das GUI zu integrieren und bei bedarf kleiner Anpassungen am Layout vorzunehmen. Gegen Ende der Diplomarbeit haben wir das GUI dann nochmals leicht überarbeitet, da je nach Benutzersprache einige Felder zu klein respektive zu gross waren.

Zum Starten der Applikation wird die Klasse AtaraxisStarter ausgeführt. Diese ruft die Klasse AtaraxisLoginGUI auf, welche bei erfolgreicher Anmeldung oder Erstellung eines neuen Benutzers eine Instanz des AtaraxisCrypter zurückliefert. Dieser wiederum wird beim Aufruf von AtaraxisMainGUI mitgegeben.



Navi. aktive Komponente

Das GUI ist in zwei Hauptbereiche aufgeteilt, links die Navigation zum aufrufen der einzelnen Komponenten, rechts die jeweils aktivierte Komponente.

Komponenten:

- Verschlüsselung
- Entschlüsselung
- Sicheres Löschen (Daten Schredder)
- Passwortverwaltung
- Einstellungen
- Information

7.3.1 Die Funktionen des GUI

Die ersten drei Komponenten haben alle ein ähnliches Layout, welches dem Benutzer die Arbeit mit der Applikation möglichst einfach machen soll. So kann der Benutzer wenn er einen Pfad einer Datei oder eines Ordners angegeben muss dies auf drei verschiedene Arten tun. Die aufwändigste

Variante ist es, den Pfad von Hand im Textfeld einzutippen. Etwas einfacher geht es, wenn man den Knopf „Durchsuchen“ (Lupe) benutzt. Dann wird auch automatisch ein adäquater Pfad für die Zieldatei vorgeschlagen. Die „Drag&Drop“-Funktionalität ist die dritte Variante. Hierzu ziehen sie einfach eine Datei oder einen Ordner in den Quell- beziehungsweise Zielbereich. Auch bei dieser Variante, wird automatisch ein adäquater Pfad für die Zieldatei vorgeschlagen.

7.3.2 Das Tray Menu

Das Tray Menu in der Taskbar bietet den Vorteil, dass das minimierte Fenster nicht erst wieder hergestellt („Restore“) und dann in die gewünschte Komponente gewechselt werden muss. Man kann damit direkt die gewünschte Komponente anwählen und das Applikationsfenster wird automatisch wieder hergestellt und in den Vordergrund gebracht.

7.3.3 Benutzereinstellungen

Jeder Benutzer kann beim Registrieren seine gewünschte Sprache auswählen. In der Konfigurationskomponente kann er diesen Wert wieder ändern und zudem denn zu verwendende Löschalgorithmus auswählen. Dieser wird bei der Ver- und Entschlüsselung verwendet und ist die vorgegebene Auswahl bei der Benutzung des Datenschredders.
Diese zwei Werte werden in der Datei `user.props` im benutzerabhängigen Verzeichnis `AtaraxiS/user_data/'Benutzernamen'/` gespeichert.

7.3.4 Applikationseinstellungen

Die Log Stufe ist die einzige Einstellung, welche applikationsweit gültig ist. Der Wert wird direkt im „property file“ von Log4j gespeichert. Diese Datei befindet sich im Verzeichnis `AtaraxiS/application_data/config/log4j.properties`.

7.3.5 Sprachabhängigkeit

Da beim Starten des LoginGUI noch nicht bekannt ist, welcher Benutzer sich anmelden will, kann die Sprache nicht vom Benutzer abhängig gesetzt werden. Deshalb haben wir uns dazu entschieden, als Sprache die Systemsprache des PCs zu verwenden.
Bei erfolgreicher Anmeldung ist der Benutzer bekannt, und im MainGUI kann die Sprache aus dem „property file“ des Benutzers gelesen. Dann wird das entsprechende „property file“ mit den sprachabhängigen Texten geladen.
Die Implementation der Sprachabhängigkeit ist deutlich zeitraubender als hart codierte Werte zu benutzen. Hinzu kommt, dass der Platzbedarf der Felder (Labels, Buttons) sich von Sprache zu Sprache sehr stark unterscheiden kann. Dies erschwert es ein klares, übersichtliches Design des GUI beizubehalten.

7.3.6 Sicherung der Benutzerdaten

Zur Sicherung der Benutzerdaten kann die Datei `Backup_Windows.bat` beziehungsweise `Backup_Linux.sh` ausgeführt werden. Der Benutzer kann wählen, wo er die komprimierten Benutzerdaten ablegen möchte. Damit verhindert werden kann, dass beim versehentlichen Löschen der Applikation sämtliche AES-Keys der Benutzer unwiderruflich verloren gehen, sollte die Sicherung der Benutzerdaten auf einem anderen Datenträger gespeichert werden. Sonst können sämtliche, verschlüsselte Daten nie mehr entschlüsselt werden!
Um nicht nur auf die Vernunft des Benutzers angewiesen zu sein, wird bei jeder neuen Benutzerregistrierung eine Sicherheitskopie erzeugt, jedoch nicht auf einem anderen Datenträger, sondern im Verzeichnis `AtaraxiS/application_data/` abgelegt. Diese Sicherungskopie kann helfen, falls ein Benutzer fälschlicherweise die Benutzerdaten, jedoch nicht sämtliche Applikationsdaten gelöscht hat. Zudem läuft dieser Vorgang im Hintergrund, ohne Einwirken des Benutzers ab, welcher so nicht unnötig gestört wird.

8 AtaraxiS Passwort Manager

Mit dem Passwort-Manager soll AtaraxiS in der Lage sein, Informationen zu Benutzerkonten (wie Benutzername, Passwort usw.) zu verwalten.

8.1 Anforderungen gemäss Pflichtenheft

Der Benutzer soll mindestens 50 Passwörter verwalten können. Zu jedem Eintrag sollen diese Informationen gespeichert werden können:

- Accountbezeichnung
- Benutzernamen
- Passwort
- Link zur Login-Seite (falls vorhanden)
- Kurze Beschreibung zum Account (200 Zeichen)

Diese Informationen sollen verschlüsselt gespeichert werden.

8.2 Lösungssuche für den Passwort Manager

8.2.1 Zusätzliche Anforderungen

Bei der Suche nach einer Lösung stellten wir sehr schnell fest, dass es nicht sehr übersichtlich sein wird, wenn wir eine Liste von rund 50 Einträgen anzeigen lassen würden. Es erschien uns sinnvoll, dem Benutzer die Möglichkeit zu bieten, ähnliche Einträge in Gruppen zusammen zu fassen.

Eine Gruppe sollte beliebig viele Passworteinträge umfassen können, allerdings keine weiteren Gruppen aufnehmen. Eine endlose Verschachtelung von Gruppen beeinträchtigt nach unserer Meinung die Usability für die Gelegenheitsnutzer zu stark ohne dabei den regelmässigen Benutzern deutliche Vorteile zu bieten.

Nachdem wir einige bestehende Programme zur Verwaltung von Passwörtern getestet hatten, legten wir die Anforderungen an die Oberfläche unserer Passwort Verwaltung fest.
Das GUI sollte aus einem Baum (SWT-Tree) bestehen, in dem man den Eintrag auswählen konnte. Der so ausgewählte Eintrag sollte neben dem Baum angezeigt werden. Über einen Knopf neben dem jeweiligen Feld sollte der Benutzername und das Passwort in die Zwischenablage kopiert und der Link zu einer allfälligen Login-Website im Browser geöffnet werden.
Beim erstellen neuer Einträge oder beim editieren bestehender soll der Benutzer die Möglichkeit haben, den Passwort-Generator aus der gleichen Maske heraus aufrufen zu können.

8.2.2 XML zur Datenspeicherung

Mit den zusätzlichen Anforderungen und der geplanten Baumstruktur zur Darstellung der einzelnen Einträge fiel unsere Wahl auf XML als Format für die Persistenz der Daten. XML speichert die Daten in einem Baumformat, ist sehr flexibel und kann nahezu beliebig erweitert und angepasst werden.

Ein direktes Abspeichern von Objekten in eine Datei haben wir verworfen, da die Verschlüsselung der Objekte sowie deren Entschlüsselung im Vergleich zur XML-Lösung wesentlich langsamer und aufwändiger wären.
Ebenfalls verworfen wurde die Verwendung einer Datenbank zum abspeichern der Daten. Mit HSQLDB [HSQLDB] gibt es eine vollkommen in Java geschriebene Datenbank, die Abfragen sehr schnell beantworten kann. Angesichts der geringen Anzahl von Einträgen die wir erwarten, erschien uns eine Datenbank jedoch eindeutig übertrieben.

8.3 Die Implementierte Lösung

8.3.1 JDOM für XML-Bearbeitung

In der inneren Auswahl für die Bearbeitung von XML standen das mit dem JDK direkt mitgelieferte DOM, die java-spezifische Erweiterung JDOM sowie DOM4J.

Die Implementierung von DOM durch SUN hält sich sehr genau an den XML Spezifikation des World Wide Web Consortiums [w3c]. Dadurch ist DOM mit Implementierungen in anderen Sprachen kompatibel, ermöglicht aber keine Java-spezifischen Vereinfachungen.

JDOM [JDOM] bietet genau diese Vereinfachungen und ermöglicht es den Java-Programmierern, sehr einfach und in gewohnter Weise mit XML zu arbeiten.
Im Rahmen der Abwägung auf welche Library wir setzen sollten, fanden wir einen interessanten Vergleich zwischen DOM und JDOM [DOM/JDOM]. Vor allem der Vergleich bezüglich dem extrahieren von Informationen aus dem XML-Dokument (Anhang 1) hat uns davon überzeugt, das wir auf JDOM setzen sollten.

Dom4J bietet wie JDOM eine einfache und für Java-Programmierer vertraute Arbeitsweise mit XML-Dokumenten. Die unserer Meinung nach geringere Verbreitung von Dom4J gegenüber JDOM sowie der grössere Platzbedarf für das dom4j-Jar führten zur Entscheidung für JDOM und gegen DOM4J.

8.3.2 DTD für die XML-Datei

Die Document Type Definition sieht folgendermassen aus:

```
<!ELEMENT ataraxis (group,account)*>
<!ELEMENT group (account)*>
<!ATTLIST group id ID #REQUIRED>
<!ELEMENT account (name?,password?,link?,comment?)>
<!ATTLIST account id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT link (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```

Das AtaraxiS XML-Dokument für die Passwort Verwaltung besteht aus den Hauptelementen `group` und `account`. Sowohl `group` als auch `account` haben als Attribut das Feld `id`, welches Dokumentenweit eindeutig sein muss. Eine `group` kann beliebig viele `account` Elemente beinhalten. Jedes `account` Element kann die Unterelemente `name`, `password`, `link` und `comment` enthalten.

8.3.3 Klasse PasswordManager.java

Die Klasse PasswordManager.java kümmert sich um das Management der Passwort-Einträge. Sie lädt und entschlüsselt die XML-Datei, erstellt und löscht Einträge und speichert die Daten wieder verschlüsselt ab.

Um den GUI-Code zu reduzieren erstellt der PasswordManager auch den SWT-Tree, der das XML-Dokument repräsentiert.

8.3.4 Erweiterungen von AtaraxisCrypter

Damit die Klasse PasswortManager.java die verschlüsselte Account-Datei ohne Zwischenschritt einer Entschlüsselung an den SAX-Reader übergeben kann, musste AtaraxisCrypter um Funktionen zum „on-the-fly“ ver- und entschlüsseln ergänzt werden.

Die gewünschte Funktionalität bestand darin, einen Stream zu erzeugen, der Daten liest, entschlüsselt und das Ergebnis weiter gibt. Die Umkehrfunktion mit Daten verschlüsseln und speichern musste ebenfalls implementiert werden, um diesen Stream dem XMLOutputter zu übergeben. Mit diesen Lösungen könnten wir ohne weitere und komplexe Ergänzungen direkt die XML-Klassen verwenden.

Da es mit CipherInputStream und CipherOutputStream bereits entsprechende Klassen gibt, haben wir auf diese zurückgegriffen und keine eigene Stream-Klasse geschrieben.

9 AtaraxiS Passwort Generator

Mit dem Passwort Generator können zufällige Passwörter mit einer beliebigen Länge erstellt werden.

9.1 Anforderungen gemäss Pflichtenheft

Damit das generierte Passwort den jeweiligen Anforderungen genügt, soll der Benutzer diese Auswahlmöglichkeiten haben:

- Länge des Passwort in Anzahl Zeichen
- Grossbuchstaben A-Z
- Kleinbuchstaben a-z
- Zahlen 0-9
- Einfache Sonderzeichen .,-

9.2 Die Implementierte Lösung

9.2.1 Klasse PasswordGenerator.java

Um zufällige Passwörter zu erstellen, nutzen wir die von SUN gelieferte Klasse java.security.SecureRandom. Mit dieser Klasse können sehr schnell sehr viele zufällige Werte generiert werden. Eine spezielle Initialisierung muss hier im Gegensatz zur Klasse java.util.Random nicht explizit erfolgen.

Vom SecureRandom–Objekt bekommen wir Integer-Werte geliefert. Um daraus ein Zeichen für das Passwort zu bekommen, haben wir ein Array mit allen vom Benutzer gewollten Zeichen gefüllt und liefern das Zeichen an der jeweiligen Position zurück. Standardmässig besteht dieser Zeichenvorrat aus den Gross- und Kleinbuchstaben, den Ziffern 0 bis 9 und den Sonderzeichen (`.:,*+/()=@`).
Die Sonderzeichen wurden von uns um zusätzliche Zeichen erweitert, damit der Auswahlbereich noch ein wenig grösser wird. Da man die Passwörter in die Zwischenablage kopieren kann, spielt es keine grosse Rolle, ob diese Zeichen auf der Tastatur einfach zu erreichen sind.

Ändert der Benutzer die Auswahl des Zeichenvorrats, wird ein neues Array mit den aktuell gültigen Werten erstellt. Die Werte werden dabei sequentiell und immer in der gleichen Reihenfolge in das Array übernommen. Da das SecureRandom–Objekt über den ganzen Auswahlbereich mit der gleichen Zufälligkeit seine Werte liefert, bringt es keine zusätzliche Sicherheit, wenn man Zeichen selber nochmals zufällig anordnet.

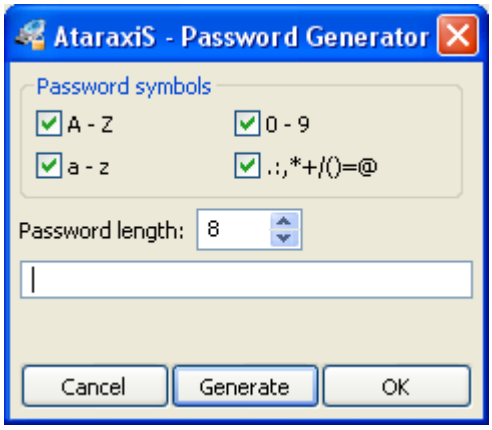
9.2.2 Klasse PWGeneratorGUI.java

Um dem Benutzer eine einfache Möglichkeit zum erstellen von Passwörtern zu bieten, haben wir eine grafische Oberfläche zum PasswordGenerator geschrieben. So können alle wichtigen Methoden einfach aufgerufen werden.

Dieses GUI ist ebenfalls Mehrsprachenfähig. Die Texte haben den Präfix „PWG.*“ und werden über die entsprechende Sprachdatei angepasst.

Ändert der Benutzer nichts an den Einstellungen, wird ihm ein Passwort mit der Länge von 8 Zeichen sowie dem gesamten Zeichenvorrat aus dem PasswordGenerator erstellt.
Will der Benutzer ein Passwort kürzer als 8 Zeichen erstellen, bekommt er eine Informationsmeldung, dass ein Passwort mit weniger als 8 Zeichen nicht sicher sei. Will der Benutzer dennoch fortfahren, muss er dies explizit verlangen.

Im GUI beträgt der maximalen Wert für den SWT.Spinner 2048 und der minimale 4. Die Länge des Passworts ist beim GUI somit auf einen Bereich von 4-2048 Zeichen limitiert. Den Maximalwert haben wir so gewählt, dass der Benutzer ohne Einschränkung einen WLAN-Key erstellen könnte.
Passwörter mit weniger als 4 Zeichen machen unserer Meinung keinen Sinn. Will der Benutzer trotzdem ein kürzeres, kann er das Passwort immer noch selber verkürzen.



Die Klasse PWGeneratorGUI kann sowohl als eigenständiges GUI aufgerufen werden wie auch als Teil einer grösseren Applikation. Um dies zu ermöglichen, haben wir einen zusätzlichen Konstruktor geschrieben, der das Shell-Objekt der umgebenden Applikation als Parameter übernimmt. Würden wir immer eine neue Shell erzeugen, würde dies zu einem Absturz der umgebenden Applikation führen. So aber kann man ohne grossen Aufwand die „Parrent-Shell“ weiter geben und alles wirkt besser integriert.

10 AtaraxiS Datenschredder

Mit dem Datenschredder von AtaraxiS sollen Daten gelöscht und deren Wiederherstellung so schwer wie möglich gemacht werden.

10.1 Anforderungen gemäss Pflichtenheft

Vor dem Löschen soll die ausgewählte Datei mindestens fünfmal abwechselnd mit einer binären Null und einer binären Eins überschrieben werden.

Zudem ist abzuklären, welche aktuellen Standards in diesem Bereich existieren und in wie weit diese implementiert werden können.

10.2 Lösungssuche für den Datenschredder

Zuerst verschafften wir uns einen Überblick über die bestehenden Löschalgorithmen. Dazu haben wir unter anderem im Wikipedia [DataRemanence] nach Informationen und im Internet nach Produkten, die sicheres Löschen von Daten anbieten, gesucht.
Besonders die von Peter Gutmann veröffentlichte Arbeit hat uns einen tiefen Einblick in die Welt der Datenschredder und der Datenwiederherstellung gewährt. Zudem ist es die einzige uns bekannte freizugängliche wissenschaftliche Arbeit in diesem Bereich. Wir gehen davon aus, dass in Geheimdienstkreisen ein Vielfaches an Informationen zu diesem Thema vorhanden wäre, welche jedoch für Aussenstehende nicht zugänglich sind.
Die Internetseite von Michael Seibold [Cipherbox] mit Informationen zu Themen rund um IT-Sicherheit hat uns auch einen guten Überblick über die bestehenden Standards verschafft.

10.2.1 Übersicht über die bekanntesten Löschalgorithmen und -standards

- VSITR: Richtlinien des BSI zum Geheimschutz von Verschlusssachen beim Einsatz von IT
- Standard 5220.22-M des US-Verteidigungsministeriums
- DSX-Methode der kanadischen Bundespolizei
- Russischer Standard, GOST P50739-95
- Bruce-Schneier-Algorithmus
- Peter-Gutmann-Algorithmus

10.2.2 VSITR des Bundesamts für Sicherheit in der Informationstechnik (Deutschland)

Beim VSITR-Standard, herausgegeben vom BSI, werden die zu löschenden Daten bei insgesamt sieben Durchläufen in den ersten sechs Durchläufen abwechselnd mit den Werten 0x00 und 0xFF und im letzten Durchlauf mit dem Wert 0xAA überschrieben. Leider konnten wir kein offizielles Dokument des BSI finden, welches die VSITR definiert, doch sämtliche Produkte, die diese Richtlinie implementieren, verwenden die oben genannten Durchgänge in dieser Reihenfolge.

10.2.3 Standard 5220.22-M des US-Verteidigungsministeriums (Department of Defense)

Der (angeblich) von der NSA entwickelte Standard [5220.22-M] zum sicheren Löschen von Daten ist im „National Industrial Security Program Operating Manual“ – genannt NISPOM – enthalten. Der Standard definiert den Umgang mit vertraulichen und geheimen Daten der Regierungsbehörden und des Militärs (US Army, Navy und Air Force).
Bei genauerer Betrachtung stellt man fest, dass es zwei unterschiedliche Standards gibt, den „DoD 5220.22-M (E)“ und den wohl neueren Standard „DoD 5220.22-M (ECE)“. Diverse Produkte auf dem Markt implementieren jedoch noch immer den älteren und weniger sicheren Standard. Beim älteren Standard werden die Daten insgesamt dreimal überschrieben. Zuerst mit einem beliebigen Wert (z.B. 0x00), dann mit dessen Komplement (z.B. 0xFF) und zuletzt mit Zufallsdaten. Gemäss dem erweiterten Standard werden die Daten zuerst mit Zufallsdaten überschrieben, dann folgen zwei Durchgänge des älteren Standards.

Gemäss Peter Gutmann muss davon ausgegangen werden, dass dieser Algorithmus eine Pseudo-Sicherheit vorgaukeln soll und keineswegs so sicher ist, wie behauptet wird. Er vermutet, dass die NSA Daten, die mit dem offiziellen Standards des DoD gelöscht wurden, wieder herstellen kann.

10.2.4 DSX-Methode der kanadischen Bundespolizei

Die "Hard Drive Secure Information Removal and Destruction Guidlines" [DSX] der “Royal Canadian Mounted Police” schreiben ein dreifaches Überschreiben der Daten vor. Datenträger mit hoch geheimen Daten müssen zusätzlich physisch geschreddert werden (max. Kantenlänge: ¼ Zoll).

10.2.5 Russischer Standard, GOST P50739-95

Die russische GOST-Norm p50739-95 (GOST: „gosudarstvennyj standart“) von 1995 verlangt für das Löschen von Daten der Sicherheitseinstufung sechs bis vier lediglich das einfache Überschreiben der Sektoren mit dem Wert 0x00 und in der Sicherheitseinstufung drei bis eins das einfache Überschreiben der Daten mit einem zufälligen Wert. Dieses Verfahren kann als nicht sicher angesehen werden.

10.2.6 Bruce-Schneier-Algorithmus

Der international anerkannte Security-Experte und Buchautor Bruce Schneier empfiehlt das siebenmalige überschreiben zur sicheren Löschung von Daten. Im ersten Durchgang sollen die Daten mit 0x00, im zweiten mit 0xFF und anschliessend fünfmal mit Zufallsdaten überschrieben werden. Durch die fünf Durchgänge mit Zufallsdaten wird eine Wiederherstellung dank Datenresten an den Spurrändern und auf Bit-Übergängen massiv erschwert.

10.2.7 Peter Gutmann-Algorithmus

Der von Peter Gutmann entwickelte Algorithmus ist mit 35 Durchgängen die mit Abstand aufwändigste und sicherste Methode zum sicheren Löschen von Daten. In seiner 1996 veröffentlichten Arbeit „Secure Deletion of Data from Magnetic and Solid-State Memory“ [PeterGutmann] erklärt er zuerst die verschiedenen Methoden der Datenwiederherstellung und widmet sich danach ausführlich der Problematik wie dieses Wiederherstellung verhindert werden kann. Dazu hat er sich überlegt welche Bitmuster das Wiederherstellen der Daten am stärksten erschweren. Da die Festplatte zum besseren Schutz der Validität (ähnlich wie bei der Datenübertragung) kodiert gespeichert werden, hat Peter Gutmann für alle derzeit gebräuchlichen Kodierungsverfahren [HD-Encodings] die Bitmuster herausgesucht, welche dann kodiert auf der Festplatte die alten Daten möglichst gründlich überschreiben. So ist er auf insgesamt 27 Bitmuster gekommen.

Zuerst werden die Daten in vier Umgängen mit Zufallsdaten überschrieben, dann werden in zufälliger Reihenfolge die Daten in 27 Umgänge mit dem jeweiligen Bitmuster überschrieben und zuletzt werden die Daten nochmals in vier Umgängen mit Zufallsdaten überschrieben.

Wie Peter Gutmann vermerkt, sollen, falls das Kodierungsverfahren bekannt ist, auch nur die dazu passenden Bitmuster (Durchgänge) geschrieben werden und nicht sämtliche 35 Durchgänge. Bei modernen Festplatten mit einer E-/PRML Kodierung schlägt er vor nur mit Zufallsdaten zu überschreiben.

10.3 Implementation des Datenschredders

Wir haben nun die bekanntesten und sinnvollsten Algorithmen ausgewählt und diese implementiert. Da Peter Gutmann selber zum Schluss kommt, dass auf modernen Festplatten mit E-/PRML Kodierung die sicherste Variante das Überschreiben mit Zufallsdaten ist, wollen wir diese Möglichkeit dem Benutzer nicht vorenthalten. Ausserdem haben wir eine Abgekürzte Variante des Gutmann Algorithmus implementiert, die nur die Durchgänge, die für Disketten (FM und MFM Kodierung) nützlich sind, ausführt.

Da der russische Standard keine hohe Sicherheit bietet und auch einfach mit den von uns implementierten Methoden (Byte, Zufallsdaten) benutzt werden kann, bieten wir keine explizite Implementation dieses Standards an.
Auch die DSX-Methode bietet aus unserer Sicht keine genügend hohe Sicherheit und ist zudem allgemein nur wenig bekannt, deshalb haben wir auch diesen Standard nicht implementiert.

10.3.1 Übersicht über die implementierten Löschalgorithmen

Byte:	Ein Durchgang mit einem Bytemuster z.B. 0x00
DoD:	Implementierung von „DoD 5220.22-M (E)“, 3 Durchgänge
DoD erweitert:	Implementierung von „DoD 5220.22-M (ECE)“, 7 Durchgänge
VSITR - BSI:	Implementierung des Standards, 7 Durchgänge
Bruce Schneier:	Implementierung der Empfehlung, 7 Durchgänge
Peter Gutmann für Disketten:	Implementierung eines Teils des Algorithmus, 18 Durchgänge
Peter Gutmann:	Implementierung des ganzen Algorithmus, 35 Durchgänge
Zufallsdaten:	Zufallsdaten bestehend aus Zufallsmuster, Zufallsbytes und „sicheren Zufallsbytes“; Anzahl Durchgänge frei wählbar, vorgeschlagen werden 10 Durchgänge

10.4 Grenzen des Datenschredders

Das sichere Löschen kann bei einem Dateisystem mit Journaling Funktion nicht gewährleistet werden. Auch bei RAID und Cluster-Systemen kann die sichere Löschung der Daten (und ihrer Kopien) nicht garantiert werden.

Auch die Methode von Peter Gutmann kann keinen hundertprozentigen Schutz vor Wiederherstellung der Daten bieten. Dazu muss der Datenträger entweder bis auf die Curie-Temperatur [Curie] erhitzt werden, bei welcher ein Element sämtliche Magnetisierung verliert, oder der Datenträger wird mit einem starken Magnet (Degausser mit Feldstärken bis zu 11'000 Gauss und einer Flussdichte von 0.8 Tesla) so stark magnetisiert, dass keine Datenrückstande mehr zurückbleiben. Zudem wird empfohlen, dass der Datenträger auch physisch zerstört (geschreddert, gepresst, gewalzt oder verbrannt) wird.

Daten, die sich auf Flash Memory (z.B. USB-Stick) befinden, können weniger gut gelöscht werden, als Daten auf Festplatten. Besonders wenn sie lange Zeit gespeichert waren, da sich dann die Daten „einbrennen“. Das sicherste ist, die Daten mehrmals mit Zufallsdaten zu überschreiben, und zwischen den Durchgängen möglichst Lange zu warten.

10.5 Möglichkeiten und Entwicklung der Datenwiederherstellung

Wiederherstellungsprogramme, die im Handel angeboten werden, sind selbst nach einem einzigen Überschreibungsvorgang nicht mehr in der Lage die Daten wiederherzustellen. Deshalb bieten wird dem Benutzer auch die relativ unsichere Methode mit lediglich einem Überschreibungsdurchgang an. Professionelle Firmen aus dem Bereich der Datenwiederherstellung verwenden zum Einlesen der Daten nicht den original Lesekopf, sondern hochwertigere, genauere Leseköpfe. Eine noch aufwändigere Wiederherstellungsmethode besteht darin, dass mit einem hochauflösenden Magnetkraftmikroskop oder mit einem neueren Rastertunnelmikroskop die elektrische Ladung präzise ausgelesen und anschliessend aufgearbeitet wird.

Da sich die Datendichte in den letzten Jahren massiv erhöht hat, ist die Wiederherstellung stark erschwert worden. Auch die Mittel zur Wiederherstellung werden weiterentwickelt, doch sie können wohl mit dem Fortschritt der hohen Datendichte nicht ganz mithalten.

Fazit: Die einzig wirklich hundertprozentig sichere Methode um Daten unwiderruflich zu löschen besteht in der physischen Zerstörung des Datenträgers!

11 Komprimierte Datei- und Verzeichnis-Verschlüsselung

AtaraxiS konnte bisher nur Dateien verschlüsseln. Mit der gleichnamigen Iteration soll AtaraxiS auch Verzeichnisse verschlüsseln können. Darüber hinaus soll, sofern dies vom Benutzer gewünscht wird, vor dem verschlüsseln die Daten komprimiert werden. Die Komprimierung kann vor allem auf langsameren USB-Sticks zu einer deutlichen Beschleunigung des Datentransfers führen.

11.1 Anforderungen gemäss Pflichtenheft

11.1.1 Verschlüsselung von Verzeichnissen

Der Benutzer muss die Möglichkeit haben, neben Dateien auch ganze Verzeichnisse auszuwählen und verschlüsseln zu lassen. Zur einfacheren Handhabung bei der Entschlüsselung soll das ganze Verzeichnis in einer verschlüsselten Datei zusammengeführt werden.

11.1.2 Komprimierte Verschlüsselung von Dateien und Verzeichnissen

Um Speicherplatz zu sparen und um weniger grosse Dateien auf den USB-Stick schreiben zu müssen soll es möglich sein, die Dateien und Verzeichnisse vor der Verschlüsselung zu komprimieren.

11.2 Lösungssuche für die Komprimierung

Mit dem zurückgreifen auf bestehende Technologien machten wir bisher gute Erfahrungen. Für die Datenkomprimierung haben wir uns daher auch nach bestehenden Lösungen umgesehen. Die gängigsten Formate um Daten zu komprimieren sind nach unseren Erfahrungen *.zip, *.tar.gz, *.ace und *.rar.

SUN selbst liefert mit dem JDK die Möglichkeit, ZIP-Dateien zu erstellen und zu bearbeiten. Von Apache Commons gibt es mit compress [compress] eine API, um *.tar, *.zip und *.bzip2 zu bearbeiten und zu erstellen. Allerdings ist diese API nicht offiziell veröffentlicht worden und liegt in der Sandbox. Dort kann sie gemäss der Warnung auf der Webseite jederzeit ohne Vorankündigung modifiziert oder umgebaut werden.
Für *.rar und *.ace haben wir keine frei verwendbare Java-API gefunden.

Nach einigen Tests mit den Zip-Klassen von SUN haben wir uns für das ZIP-Format entschieden. So erreichen wir zwar nicht die maximal mögliche Komprimierungsrate, haben dafür aber ein Format, das sich durchgesetzt hat und mit den Bordmitteln der meisten Betriebssysteme direkt bearbeitet werden kann.

Mit LZMA von 7-zip [7zip] könnte in Zukunft eine gute Alternative aufkommen. Eine offene Architektur, integrierte Verschlüsselung mit 256-Bit AES sowie eine sehr schnelle Komprimierung sprechen für sich. Dem steht momentan leider eine kaum vorhandene Dokumentation entgegen. Sobald sich diesbezüglich etwas ändert, sollte man sich diese API genauer anschauen.

11.3 Lösungssuche für die Verzeichnis-Verschlüsselung

Der Nachteil des Ansatzes im Pflichtenheft ist, dass der Benutzer immer das ganze Verzeichnis entschlüsseln muss und nicht nur eine einzige Datei herauslesen kann. Nach eingehender Analyse haben wir uns aber dennoch für diesen Ansatz entschieden. Die einfache Handhabung bei der gleichzeitig bestehenden Möglichkeit, nur einzelne Dateien zu verschlüsseln, sollte es jedem Benutzer ermöglichen, den für ihn passenden Ablauf zu wählen.

Um nur eine Zieldatei zu haben, hat sich nach den Tests für die Komprimierung das ZIP-Format hervorgetan. Um Verzeichnisse zu verschlüsseln benötigt es so nur eine Prüfung, ob es sich bei der Quelldatei um ein Verzeichnis handelt. Falls ja, wird auf jeden Fall komprimiert und eine verschlüsselte ZIP-Datei erstellt.

11.4 Die Implementierte Lösung

11.4.1 Benutzerführung

Im GUI kann der Benutzer wählen, ob die Datei beim verschlüsseln komprimiert werden soll. Wählt der Benutzer ein Verzeichnis aus, wird auf jeden Fall komprimiert. Sowohl der AtaraxisCrypter wie auch das GUI prüfen, ob man ein Verzeichnis verschlüsseln will. Dies machen wir aus dem Grund, da nur das GUI den Benutzer auf die notwendige Komprimierung aufmerksam machen kann.

Beim Entschlüsseln kann der Benutzer auf dem GUI wählen, ob die verschlüsselte und komprimierte Datei entschlüsselt und dekomprimiert oder ob nur entschlüsselt werden soll. Wird beides gewählt, hat der Benutzer am Ende die gleiche Datei wie vor dem entschlüsseln. Wählt er für eine komprimierte Datei nur das entschlüsseln, wird die Dateiendung auf *.zip geändert und der Benutzer hat eine entschlüsselte ZIP-Datei im Zielverzeichnis. Diese kann er dann mit jedem beliebigen ZIP-Programm bearbeiten.

Um zu unterscheiden, ob die verschlüsselte Datei komprimiert ist oder nicht, haben wir eine weitere Dateiendung eingeführt:

*.ac	verschlüsselte Datei
*.acz	verschlüsselte und komprimierte Datei oder Verzeichnis

11.4.2 Erweiterungen von AtaraxisCrypter

Um Dateien zu komprimieren mussten die Methoden encryptFile() und decryptFile() erweitert werden. Die Methodensignatur wurde von (String, String) auf (String, String, boolean) angepasst. Der Boolean-Wert steht auf true wenn zusätzlich komprimiert oder dekomprimiert werden soll und auf false, falls nur verschlüsselt oder entschlüsselt werden soll.

Die Methoden mit der alte Signatur wurde beibehalten, allerdings rufen diese nur die neuen Methoden auf. Dabei wird bei encryptFile() als boolean false mitgeliefert und bei decryptFile() true. Nutzt man diese Methoden, kann man sich ohne um die Komprimierung zu kümmern Daten ver- und entschlüsseln. Soll eine Datei bearbeitet werden, wird diese unkomprimiert verschlüsselt und danach wieder entschlüsselt. Bei einem Verzeichnis wird dies zuerst komprimiert und beim entschlüsseln wieder dekomprimiert.
Dieses Verhalten funktioniert deshalb, weil der AtaraxisCrypter selber komprimiert, falls es sich bei der übergebenen Quelldatei um ein Verzeichnis handelt (bei encryptFile()). Beim entschlüsselt wird nur dekomprimiert, wenn die Datei die Endung *.acz hat. Ändert der Benutzer die Endung von *.acz nach *.ac, hat er nach dem entschlüsseln ein Zip-File auf der Disk. Ändert er die Endung von *.ac nach *.acz, wirft AtaraxisCrypter eine ZipException.

11.4.3 Erstellen einer verschlüsselten Zip-Datei im AtaraxisCrypter

Die Erstellung einer Zip-Datei ist in Java sehr einfach. Um eine verschlüsselte Zip-Datei zu erstellen, muss man nur sehr wenig anpassen.
Als erstes wird ein AtaraxisCrypter.encryptedOutputStream() erzeugt, dieser übergibt man einem ZipOutputStream() und von da an erstellt man eine gewöhnliche Zip-Datei. Für jede Datei, die in der Zip-Datei abgelegt werden soll, wird ein ZipEntry erzeugt. Dieser ZipEntry hat im Konstruktor einen Namen, der den Pfad in der Zip-Datei repräsentiert. Der Zip-Entry wird der Zip-Datei hinzugefügt und danach die zu komprimierende Datei in den ZipOutputStream kopiert. Soll ein Verzeichnis in der Zip-Datei erstellt werden, muss dessen Namen mit „/“ enden.

12 Einfaches Deployment-Verfahren

Damit unser Programm von möglichst vielen Personen verwendet werden kann, muss es so einfach wie möglich zu bedienen sein. Auch die Installation und Konfiguration dürfen nicht komplexer als nötig sein.

12.1 Anforderungen gemäss Pflichtenheft

Der Endbenutzer soll maximal den Speicherort der Konfigurationsdateien und die Sprache konfigurieren müssen, bevor er nach dem entpacken die Applikation starten kann.

Nach dem Start der Applikation muss der Benutzer einen Benutzernamen sowie ein Passwort auswählen und bestätigen.

12.2 Probleme des Deployment-Verfahrens

Als mögliche Probleme oder Hürden für ein einfaches Deployment-Verfahren stellen sich unserer Meinung nach diese Punkte heraus:

- Verteilen der Software
- Installation der Software und der benötigten Bibliotheken
- Installation der „unrestricted policy files“ für das JRE
- Backup der Benutzerdaten
- Automatischer Start von AtaraxiS beim Einstecken des USB-Sticks

12.3 Die Implementierte Lösung

Zum Verteilen der Software ist es wohl am einfachsten, wenn der Benutzer nur eine gepackte Datei bekommt, die er entweder selber entpacken kann oder die ihm mit einem Assistenten geführt die einzelnen Schritte der Installation abnimmt.
Da wir eine möglichst portable Lösung gesucht haben, entschieden wir uns für eine Zip-Datei, die unsere Software enthält. Mit Hilfe von Ant konnten wir den Task `distGeneric` erstellen, der nach dem kompilieren der Applikation eine fix fertige Zip-Datei erstellt. Diese Zip-Datei muss für die Verteilung von AtaraxiS auf einen Download-Server oder eine CD gebracht werden und der Benutzer muss diese Zip-Datei nur noch entpacken.

Mit der so erstellten Zip-Datei bekommt der Benutzer alle benötigten Bibliotheken und unsere Software. Was noch fehlen kann, ist ein JRE. Die Zip-Datei enthält bereits Start-Skripte für ein mitgeliefertes JRE, wegen der Grösse des JRE ist dies aber nicht in dieser Zip-Datei enthalten. Mit dem zusätzlichen jre-win.zip kann der Benutzer sich ein JRE für Windows herunterladen, das ebenfalls auf dem USB-Stick abgelegt werden kann. Gleiches gilt bei jre-linux.zip für das Betriebssystem Linux.

Für die „unrestricted policy files“ haben wir 2 Lösungen zur Auswahl. Der AtaraxisCrypter testet bei der Initialisierung, ob er eine Cipher mit einem 256-Bit AES-Key erstellen kann. Falls nicht, wirft er eine Exception, die im GUI abgefangen wird. Mit einem Dialog wird der Benutzer angefragt, ob er die „unrestricted policy files“ kopieren wolle. Hat der Benutzer genügend Rechte, werden die benötigten Dateien kopiert. Fehlen ihm diese Rechte, muss er die 2 Jar-Dateien manuell vom Administrator kopieren lassen – falls er das auf dem System installierte JRE nutzen will. Alternativ kann er das JRE auf dem USB-Stick nutzen, wo er die benötigten Rechte haben sollte.

Verliert der Benutzer seine Datei mit dem AES-Schlüssel, kann er die verschlüsselten Dateien nicht mehr entschlüsseln. Aus diesem Grund haben wir eine Backup-Möglichkeit implementiert, mit der die Benutzerdaten innerhalb von user_data gesichert werden können. Bei jedem erstellen eines neuen Benutzers wird automatisch ein Backup erstellt. Über dies hinaus kann der Benutzer explizit

ein Backup erstellen lassen, in dem er die Datei Backup_Win.bat oder Backup_Linux.sh ausführt. Der Benutzer kann hierbei selber wählen, wo dieses Backup abgelegt werden soll. Da es sich bei der erzeugten Backup-Datei um eine gewöhnliche Zip-Datei handelt, kann er es ohne spezielles Recovery-Tool wieder herstellen. Ein eigenes Tool für die Wiederherstellung haben wir daher nicht implementiert.

Für den automatischen Start von AtaraxiS nach dem einstecken des USB-Sticks haben wir uns nach einer bestehenden Lösung umgesehen. Leider konnten die meisten Tools unsere Erwartungen nicht erfüllen, da es derzeit keine direkte Autostart-Möglichkeit in Windows XP gibt. Unter Linux sieht es diesbezüglich derzeit ebenfalls schlecht aus.
Um dennoch einen automatischen Start zu ermöglichen, haben wir zwei Lösungen integriert, die unseren Wünschen sehr nahe kommen:

- Das Magazin C't hat in der Ausgabe 26/2002 einen Überwachungsdienst [C't USB Agent] vorgestellt, der unter Windows 2000 und Windows XP das einstecken und abmelden von USB-Sticks überwacht und ein Event auslösen kann. Damit dies funktioniert, muss ein Programm auf dem PC laufen und im Root-Verzeichnis des USB-Sticks muss sich eine Datei mit dem Namen usbagent.inf befinden.
- PortableApps [PortableApps] bietet eine Sammlung von Programmen für die direkte Ausführung auf einem USB-Stick. Damit man die einzelnen Programme erreichen kann, wird ein Programm-Menu mit einer Liste aller Tools gestartet. Dazu wird eine angepasste autostart.inf verwendet, wie sie für den Start einer CD-ROM gebraucht wird.

In der Benutzerdokumentation erklären wir, wie man beide Varianten verwenden kann. Der Nachteil des C't USB Agenten ist, dass man ein Programm installieren muss, dafür startet es AtaraxiS automatisch. Mit der autostart.inf wird AtaraxiS nicht direkt gestartet, es erscheint aber ein Dialog, bei dem man mit einem klick AtaraxiS starten kann:

Mit all diesen einzelnen Schritten können wir dem Benutzer einen einfachen Umgang mit AtaraxiS ermöglichen. Wenn auch nicht alles voll automatisch abläuft, so muss der Benutzer doch kaum manuell in den Programmablauf eingreifen.

13 Tests

Da wir die ganze Verschlüsselung bereits in der Semesterarbeit implementiert und mit JUnit Tests geprüft haben, müssen nun nur noch zwei neue Komponenten getestet werden. Das GUI selbst wird nicht mit JUnit Tests geprüft, da das Schreiben eines aussagekräftiger Test für das GUI in keiner Relation zum Aufwand stehen würde. Es wurden nicht nur Positiv-Tests, sondern auch Negativ-Tests zur Überprüfung des Exception Handlings sowie zum Abstecken der Grenzen der Applikation erstellt.
Die Tests können mit dem Ant-Task `test` ausgeführt werden

13.1 Schredder

Der JUnit Test ShredderTest prüft sämtliche public Methoden der Klasse AtaraxisShredder. Mit der Klasse SecureRandomPerformance haben wir die Performance von Random und SecureRandom geprüft und stellten einen deutlichen Unterschied fest. Doch noch wichtiger ist die Tatsache, dass SecureRandom nur benutzt werden sollte, wenn es auch wirklich nötig ist. Da jedoch eine echte Zufälligkeit beim Schreiben von Zufallsdaten nicht entscheidend ist, hingegen aus diesem Vorgang kritische Informationen gewonnen und zum Angriff auf verschlüsselte Daten benutzt werden könnten, haben wir uns entschlossen, beim AtaraxisShredder standardmässig mit den von Random generierten Zufallsdaten zu arbeiten.

13.2 Passwort Verwaltung

Die Passwort Verwaltung besteht aus einem Teil mit der Behandlung der XML-Daten und einem GUI-Teil. Um einen umfassenden Test aller Funktionen zu machen, haben wir auf JUnit verzichtet und stattdessen einen manuellen Testablauf notiert.

Wir haben dazu eine Excel-Tabelle verwendet und die Aktionen sowie das erwartete Ergebnis notiert. Als nächsten Schritt haben wir unter Windows den Test durchgeführt und die erwarteten mit den effektiven Ergebnissen verglichen. Auf Grund der Resultate mussten wir das Verhalten der PasswordManager-Klasse anpassen und den Test erneut durchführen.
Als wir unter Windows bei allen Vorgängen die erwarteten Ergebnisse erzielt hatten, führten wir den Test unter Linux durch.

Diesen Test wiederholten wir jeweils als Teil des Integrationstests und am Projektende als Abschlusstest. Als Anhang 2 finden Sie das Testprotokoll für den Abschlusstest unter Windows.

Die Klasse so zu testen war zeitaufwendig, doch konnten dadurch einige Fehler gefunden werden, die sonst erst beim Endbenutzer aufgefallen wären.

14 Ausbaumöglichkeiten

AtaraxiS ist nach der Diplomarbeit noch nicht „fertig“. Die Kann-Ziele wurden in der Diplomarbeit nicht angegangen und wären die nächsten Schritte für den Ausbau von AtaraxiS. Wir haben diese Ziele sowie in der Diplomarbeit erhaltenen Anregungen hier ein wenig genauer umschrieben.

Erweiterung des Drop-Mechanismus zur Unterstützung mehrerer Dateien
Besonders sinnvoll würden wir eine Erweiterung des Drop-Mechanismus erachten, mit welchem nicht nur einzelne Dateien, sondern mehrere gleichzeitig eingefügt werden können. Diese Erweiterung würde auch keinen grossen Aufwand bedeuten, doch gibt es dabei ein grosses Problem. Was passiert, wenn eine Datei nicht bearbeitet (z.B. gelöscht) werden kann? Dann bricht das Programm ab und einige Dateien wurden bereits bearbeitet, andere jedoch nicht. Um dieses Problem umgehen zu können wäre ein grösserer Programmieraufwand erforderlich. Aus diesem Grund haben wir diesen Punkt nicht mehr in der Diplomarbeit realisiert. Ein weiterer Nachteil wäre, dass der Benutzer die Namen der neu zu erstellenden Dateien nicht mehr selber wählen könnte, sie würden von der Applikation vorgegeben.

Public-/Private-Key Verschlüsselung für Texte / E-Mails
Um Daten zwischen 2 Benutzern auszutauschen, könnte die Verwendung einer Public-/Private-Key Verschlüsselung sehr hilfreich sein. So könnte man, ohne sich um den Austausch eines geheimen Schlüssels kümmern zu müssen, Daten sicher austauschen. Bouncy Castle bietet eine API für OpenPGP, auf die man für diese Zwecke zurückgreifen könnte.

Import- / Exportmöglichkeit für Public-Key Zertifikate
Damit die Public-/Private-Key Verschlüsselung funktionieren kann, muss der eigene Public-Key exportiert und die Public-Key-Zertifikate anderer Personen importiert werden können. Der Keystore in AtaraxiS bietet von Beginn weg die Möglichkeit zum abspeichern von Zertifikaten. Um die Zertifikate zu Verwalten und zu im- und exportieren müsste eine weitere Stack-Komponente im GUI erstellt werden.

Hashbasierte Dateiverifikation
Zur Verifikation der Dateiintegrität werden häufig Checksummen oder Hashwerte verwendet. Um die erhaltene Datei zu kontrollieren, benötigt man aber oft Programme, die auf dem PC nicht installiert sind. Es wäre daher ideal, wenn man AtaraxiS ebenfalls zur Dateikontrolle nutzen könnte. Die Kontrollsummen können mit Java und Bouncy Castle einfach erstellt werden, eine Erweiterung um eine solche Funktion währe daher relativ einfach in AtaraxiS zu integrieren.

Detailliertere Kontrolle des Ver- und Entschlüsselungsvorgangs
Mit Hilfe der hashbasierten Dateiverifikation könne man auch den Ver- und Entschlüsselungsvorgang besser überwachen. Es wäre möglich, den Hashwert der Originaldatei zu berechnen und diesen nach der Entschlüsselung mit dem Hashwert der neuen (entschlüsselten) Datei zu vergleichen. Stimmen diese überein, hätte man die Gewissheit, dass die beiden Dateien den gleichen Inhalt haben.
Den Hashwert der Originaldatei müsste man dazu aber irgendwo zwischenspeichern, denn die Originaldatei liegt beim Entschlüsseln in der Regel nicht vor. Als Speicherort für den Hashwert könnte entweder eine zusätzliche Meta-Datei dienen oder aber man schreibt diesen Wert an das Ende oder den Anfang der verschlüsselten Datei.
Eine solche Erweiterung von AtaraxiS ist aber mit einigem Aufwand verbunden, könnte sich aber dennoch lohnen.

15 Verwendete Software

Eclipse	Entwicklungsumgebung	3.2.1
Checkstyle	Plugin für Code-Kontrolle	4.1.1
SmartSVN	Zugriff auf SVN-Repository	2.0.7
Subclipse	SVN-Zugriff aus Eclipse	1.0.3
Log4e	Log4j für Eclipse	1.2.11
Java-SDK	Standard Edition	1.5.0_09
Bouncy Castle	Security Provider für Java	jdk15-134
SWT	GUI-Toolkit	3.2.1
WindowBuilder	Designer für SWT	5.1.0
Ant	Build-Tool	1.6.5
JUnit	Test-Framework	4.1.0
Microsoft Office	Office Programme	2003

15.1 Bemerkungen zu einzelnen Programmen

15.1.1 Eclipse

Um die Zeit für das Einrichten und Konfigurieren von Eclipse zu verkürzen, haben wir in der Diplomarbeit einen anderen Weg gewählt als in der Semesterarbeit. In der Semesterarbeit haben wir auf jeder Station Eclipse neu konfiguriert. In der Diplomarbeit haben wir Eclipse nur auf einem PC komplett eingerichtet und konfiguriert und danach den Programmordner und den Ordner mit den Workspaces verteilt.

Mit der stabilen Version 3.2.1 funktionierte auch Subclipse wieder. Dadurch konnten wir nun direkt von Eclipse aus voll auf das SVN-Repository zugreiffen. Dies hat die Arbeit deutlich erleichtert, da nicht mehr manuell die Dateien mit dem lokalen SVN-Verzeichnis abgeglichen werden mussten.

15.1.2 WindowBuilder

Die Vollversion des WindowBuilder [WindowBuilder] der Firma Instantiations ist kostenpflichtig, da uns jedoch der Funktionsumfang des frei erhältlichen Teils SWT Designer genügte, haben wir uns für diesen Entschieden. Durch die Nutzung eines GUI Designers kommt man zu Beginn deutlich schneller vorwärts. Schwierigkeiten gab es dann jedoch Beispielsweise bei StackLayouts, das Kopieren von ganzen Komponenten funktioniert zwar problemlos, doch dann hat man im Code lauter mehrfach Nummerierte Variabelnamen, die das Verständnis beim Lesen des Codes stark erschweren. Deshalb muss man jede einzelne Variable des kopierten Komponenten nachträglich umbenennen. Ein grosser Nachteil ist ausserdem, dass bei jeder Änderung im StackLayout, die oberste Komponente (welche beim starten angezeigt wird) im Code geändert und in unserem Fall die ganze Logik im Switch-Statement wieder zu Nichte gemacht wurde (löschte Befehle zum wechseln der obersten, sichtbaren Komponente).
Grössere Schwierigkeiten haben wir zudem beim ändern von GridLayouts gehabt (andere Spaltenanzahl), dies hat jeweils die Teile, welche mehrere Spalten gross sind nicht korrekt angepasst und zudem ein durcheinander in der Reihenfolge der Felder hervorgerufen.

Schlussendlich können wir sagen, dass ein Tool zum Erzeugen des GUI sehr hilfreich, bei Änderungen am GUI jedoch grösste Vorsicht geboten ist. Wichtig ist, dass bereits die erste Version des GUI gut überdacht ist und nachträglich keine grösseren Änderungen und Erweiterungen mehr vorgenommen werden müssen.

16 Aktivitäten der Gruppenmitglieder

16.1 Aktivitäten von J. Graber

- Erstellen des Pflichtenhefts
- Erstellen des Projekthandbuchs
- Erstellen des Ablaufplans
- Erstellen des Endberichts
- Erstellen des Benutzerhandbuchs
- Implementieren des PasswortManagements (Klasse & GUI)
- Implementieren des Passwortgenerators (Klasse & GUI)
- Implementieren der komprimierten Datei- und Verzeichnisverschlüsselung
- Implementierung TextInputDialog
- Erstellen der Tasks distGeneric, runLinux im Ant-Buildfile
- Implementierung der fürs deployment notwendigen Tools (Backup, PolicyCopy)
- Beraten und Unterstützen bei GUI-Entwicklung

16.2 Aktivitäten von A. Müdespacher

- Mitarbeit am Pflichtenheft
- Mitarbeit am Projekthandbuch
- Erstellen von Teilen des Endberichts
- Mitarbeit am Benutzerhandbuch
- Implementieren des MainGUI – Gerüsts
- Implementieren des Starter & LoginGUI
- Implementieren des Shredders (AtaraxisShredder & ShredderTest)
- Implementieren der MainGUI – Teile: Verschlüsselung, Entschlüsselung, Schredder, Konfiguration, Information und des Tray Menus
- Erstellen der Tasks jar, test sowie Modifikation diverser Tasks im Ant-Buildfile
- Beraten und Unterstützen bei der Erweiterung des AtaraxisCrypter

17 Glossar

AES	American Encryption Standard
AWT	Abstract Widget Toolkit
BSI	Bundesamt für Sicherheit in der Informationstechnik (Deutschland)
CLI	Command Line Interface
DoD	Department of Defense, Verteidigungsministerium der USA
EPMLR	Extended Partial Response/Maximum Likelyhood
FM	Frequency modulation, Frequenzmodulation
GUI	Graphical User Interface
JAR	Java-Archiv
MFM	Modified Frequency Modulation, modifizierte Frequenzmodulation
PMLR	Partial Response/Maximum Likelyhood
RSA	Verschlüsselungsalgorithmus von Ron **R**ivest, Adi **S**hamir und Len **A**dleman
SWT	Standard Widget Toolkit
VSITR	Richtlinien zum Geheimschutz von Verschlusssachen beim Einsatz von IT

18 Quellen

18.1 Dokumente

[evalAlgo] SVN:…/documents/Evaluation_Algorithmen.pdf
[entsGUI] SVN:…/documents/Entscheidung_GUI.pdf

18.2 Bücher

Beginning Cryptography with Java
von David Hook, 2005, Wiley and Sons Ltd (ISBN: 0-7645-9633-0)

18.3 Links

Sämtliche aufgeführte Links beziehen sich auf die Version vom 15. Dezember 2006.

[5220.22-M] http://www.usaid.gov/policy/ads/500/d522022m.pdf
[C't USB Agent] ftp://ftp.heise.de/pub/ct/listings/0226-206.zip
[Cipherbox] http://www.cipherbox.de/sicherheit-wipe.html
[cliJakarta] http://jakarta.apache.org/commons/cli/
[compress] http://jakarta.apache.org/commons/sandbox/compress/
[Curie] http://de.wikipedia.org/wiki/Curie-Temperatur
[DataRemanence] http://en.wikipedia.org/wiki/Data_remanence
[DOM/JDOM] http://www.fh-wedel.de/~si/seminare/ss01/Ausarbeitung/4.domjdom/dom4.htm
[DSX] http://www.rcmp-grc.gc.ca/tsb/pubs/it_sec/g2-003_e.pdf
[HD-Encodings] http://www.storagereview.com/guide2000/ref/hdd/geom/data.html
[HSQLDB] http://www.hsqldb.org
[JDOM] http://jdom.org/
[PeterGutmann] http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html
[PortableApps] http://portableapps.com/
[SWT] http://www.eclipse.org/swt/
[w3c] http://www.w3.org/XML/
[WindowBuilder] http://www.instantiations.com/windowbuilderpro/

SWT-Präsentation von Ben Galbraith http://www.javalobby.org/eps/swt_intro/
Getting Started with Eclipse and the SWT http://www.cs.umanitoba.ca/~eclipse/

19 Anhang 1: Vergleich DOM/JDOM

Als Vergleich soll mit DOM und mit JDOM aus dem untenstehenden XML-Dokument der Mannschaftsname, die Punkte und die Tordifferenz in geeigneten Variablen gespeichert werden.

```
<league>
  <team>
    <name>Real Madrid</name>
    <points>23</points>
    <goals own="36" against="11"/>
  </team>
</league>
```

19.1 DOM

```
import org.w3c.dom.*;
...
try {
  Element t = (Element) root.getElementsByTagName("team").item(0);

  // Name der Mannschaft als String speichern
  Node n = t.getElementsByTagName("name").item(0);
  String name = n.getFirstChild().getNodeValue();

  // Anzahl der Punkte als int speichern
  Node p = t.getElementsByTagName("points").item(0);
  int pts = Integer.parseInt(p.getFirstChild().getNodeValue());

  // Tordifferenz als int speichern
  Element g = (Element) t.getElementsByTagName("goals").item(0);
  Attr own = g.getAttributeNode("own");
  Attr ags = g.getAttributeNode("against");
  int diff = Integer.parseInt(own.getValue()) -
             Integer.parseInt(ags.getValue());
}
catch (Exception e) {  // Ausnahmebehandlung hier nur rudimentär
  ...
}
...
```

19.2 JDOM

```
import org.jdom.*;
...
try {
  Element t = root.getChild("team");
  // Name der Mannschaft als String speichern
  String name = t.getChild("name").getText();
  // Anzahl der Punkte als int speichern
  int pts = Integer.parseInt(t.getChild("points").getText());
  // Tordifferenz als int speichern
  int diff = t.getChild("goals").getAttribute("own").getIntValue() -
             t.getChild("goals").getAttribute("against").getIntValue();
}
catch (Exception e) {  // Ausnahmebehandlung hier nur rudimentär
  ...
}
...
```

20 Anhang 2: PW-Manager Test (Windows)

Test OK

Testdatum: 14.12.2006

Ausgangslage	Aktion	Ergebnis (erwartet)	Bestanden	Ergebnis (wenn ungleich)
Neuer Benutzer erstellt		Benutzer erstellt und eingeloggt	Ja	
		E-Mail existiert	Ja	
	Gruppe Web hinzufügen	Web hinzugefügt	Ja	
	Eintrag BFH hinzufügen	BFH hinzugefügt	Ja	
	Web/ti hinzufügen	Web/ti hinzugefügt	Ja	
	E-Mail/B hinzufügen	E-Mail/B hinzugefügt	Ja	
Logout				
Login				
		2 Gruppen, 3 Einträge bestehen	Ja	
	Web/ti anzeigen und auf editieren klicken	Editiermodus bei Web/ti erscheint	Ja	
	Web/ti: PW generieren und speichern	pw erstellt und gespeichert	Ja	
	BFH: URL zu hermes hinzufügen	hermes-URL hinzugefügt	Ja	
	BFH: Passwort ilop + Benutzernamen JG add & save	JG hinzugefügt und gespeichert	Ja	
	BFH: URL-Button klicken	Browser öffnet sich	Ja	
	BFH: Benutzernamen kopieren	Benutzername in Zwischenablage	Ja	
	BFH: Passwort kopieren	Passwort in Zwischenablage	Ja	
	E-Mail/B: leeres Passwort kopieren	Keine Änderung der Zwischenablage	Ja	
	E-Mail/B: Eintrag editieren, Wechsel auf BFH	Meldung Änderung zuerst beenden	Ja	
	E-Mail/B: (Eintrag wird editiert) Wechsel auf Web	Meldung Änderung zuerst beenden	Ja	
	Editieren abbrechen	Anzeigemodus erscheint	Ja	
Logout				
Login				
		2 Gruppen, 3 Einträge bestehen	Ja	
	Web ausklappen	Baum Web ausgeklappt, ti sichtbar	Ja	
	BFH: mit 'Eintrag löschen' löschen	Meldung wirklich löschen	Ja	
	Löschen bestätigen	BFH weg, Web bleibt ausgeklappt	Ja	
	Web/ti auswählen	Web/ti wird angezeigt	Ja	
	Web/ti mit 'Gruppe löschen' löschen	Fehlermeldung nur leere Gruppen	Ja	
	E-Mail mit 'Gruppe löschen' löschen	Fehlermeldung nur leere Gruppen	Ja	
	Web mit 'Eintrag löschen' löschen	Fehlermeldung nur Einträge löschen	Ja	
	Web/ti mit 'Eintrag löschen' löschen	Wirklich löschen?	Ja	
	Löschen mit Nein beenden	Eintrag wird nicht gelöscht	Ja	

	Web/ti mit 'Eintrag löschen' löschen	Wirklich löschen?	Ja
	Löschen mit JA bestätigen	Eintrag wird gelöscht	Ja
	Web mit 'Gruppe löschen' löschen	Wirklich löschen?	Ja
	Löschen mit Nein beenden	Gruppe wird nicht gelöscht	Ja
	Löschen mit JA bestätigen	Gruppe wird gelöscht	Ja
	E-Mail/b und E-Mail wie vorgesehen löschen	Einträge gelöscht	Ja
		PW-Verwaltung leer	Ja
Logout			
Login			
		PW-Verwaltung leer	Ja
	Eintrag meinPC erstellen	Eintrag meinPC erstellt	Ja
	Gruppe meineAccounts erstellen	Gruppe meineAccounts erstellt	Ja
	Gruppe meinPC erstellen	Meldung Gruppe existiere bereits	Ja
	Eintrag meinPC erstellen	Meldung Titel existiert bereits	Ja
	Eintragstitel löschen, mit leerem Titel speichern	Meldung Titel darf nicht leer sein	Ja
	1 Leerzeichen als Titel	Meldung Titel darf nicht leer sein	Ja
	meinPC(Leerzeichen) als Titel	Meldung Titel existiert bereits	Ja
	MeinPc als Titel	Eintrag wird erstellt	Ja
	Eintrag meinPC auswählen	Eintrag wird angezeigt	Ja
	Eintrag meinPC löschen	Eintrag gelöscht, Felder leer	Ja
Logout			
	Passwort-Datei löschen	Passwort-Datei gelöscht	Ja
Login			
	Meldung Datei fehlt, Nein antworten	PWM Anzeige leer, Buttons inaktiv	Ja
Logout			
Login	Meldung Datei fehlt, Ja antworten	PWM Buttons aktiv, Eintrag E-Mail	Ja
	Gruppe Web hinzufügen	Web hinzugefügt	Ja
	Eintrag BFH hinzufügen	BFH hinzugefügt	Ja
	BFH: Gruppe wechseln auf Web	BFH unter Web eingeordnet	Ja
Logout			