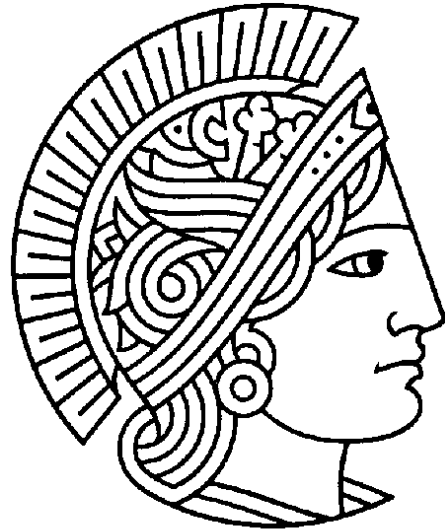


Technische Universität Darmstadt



Diplomarbeit

Erweiterung der Flexi-PKI um einen UMP-Update-Service

eingereicht beim

Fachbereich Informatik

Fachgebiet Theoretische Informatik

Prof. Dr. Johannes Buchmann

von

Björn Seiffert

Betreuer: Michael Hartmann

Januar 2004

Inhaltsverzeichnis

Kapitel 1 Einführung.....	7
Kapitel 2 Das Fail-Safe-Konzept.....	9
2.1 Grundlagen der PKI.....	9
2.2 Das Konzept.....	9
2.3 Komponenten.....	10
2.3.1 Teil-PKIs.....	10
2.3.2 Update-Service.....	11
2.3.3 Outlook-Plugin.....	11
2.4 Schadensfälle.....	11
2.4.1 CH-Schlüssel wurde kompromittiert.....	11
2.4.2 CA-Schlüssel wurde kompromittiert.....	12
2.4.3 Signaturverfahren wurde kompromittiert.....	12
2.4.4 Hashfunktion ist unsicher geworden.....	12
2.4.5 Signaturalgorithmus wurde kompromittiert.....	13
2.4.6 Paddingverfahren wurde kompromittiert.....	13
2.4.7 Mathematisches Basisproblem wurde gelöst.....	13
2.5 Das Update Management Protocol (UMP).....	14
2.5.1 Komponenten des UMP.....	14
2.5.1.1 Komponenten, die den Benutzer näher spezifizieren.....	14
2.5.1.2 Registry.....	15
2.5.1.3 UpdateComponent (UC).....	16
2.5.1.4 Das UpdateComponentResponse (UCR).....	16
2.5.1.5 Sicherheitsanker (Trust-Anchor, TA).....	17
2.5.1.6 Provider.....	17
2.5.1.7 UpdateComponentCode (UCC).....	18
Kapitel 3 Update-Service.....	19
3.1 Ablauf im Schadensfall.....	19
3.2 Komponenten des Update-Service.....	21
3.3 Bedienung.....	23
3.3.1 Status-Fenster.....	23
3.3.2 Schadensaufnahme.....	24
3.3.2.1 CH-Schlüssel kompromittiert.....	25
3.3.2.2 CA-Schlüssel kompromittiert.....	26
3.3.2.3 Hashfunktion kompromittiert.....	26
3.3.2.4 Signaturalgorithmus kompromittiert.....	27
3.3.2.5 Signaturverfahren kompromittiert.....	27
3.3.2.6 Paddingverfahren kompromittiert.....	28
3.3.2.7 Mathematisches Basisproblem gelöst.....	28
3.3.2.8 Neue Registry installieren.....	29
3.3.3 Aktionen festlegen.....	29
3.3.3.1 Einzelnen CH-Schlüssel löschen.....	30
3.3.3.2 Gruppe von CH-Schlüsseln löschen.....	30
3.3.3.3 Neuen CH-Schlüssel generieren.....	31
3.3.3.4 Sicherheitsanker löschen.....	31

3.3.3.5 Sicherheitsanker austauschen.....	31
3.3.3.6 Neue Zertifikate für CHs installieren.....	32
3.3.3.7 Algorithmen löschen.....	32
3.3.3.8 Neuen Sicherheitsanker installieren.....	33
3.3.3.9 Mehrere neue CH-Schlüssel generieren.....	33
3.3.4 Neue Provider an Clients verteilen.....	34
3.3.4.1 UpdateComponentCode-ID bestimmen.....	34
3.3.4.2 Zuordnung Provider-Klasse + ClientIdentifier → Provider.....	34
3.3.4.3 JCA Provider.....	36
3.3.5 Erweiterte Eingaben.....	36
3.3.5.1 „Provider installieren“-Aktionen nachträglich bearbeiten.....	37
3.3.5.2 Installierte CAs.....	38
3.3.6 Infos anzeigen.....	39
3.3.6.1 UpdateComponent-Status.....	40
3.3.6.2 UpdateComponent, UpdateComponentResponse, UpdateComponentCode, Registry anzeigen.....	40
3.3.6.3 Datenbankeinträge löschen.....	41
Kapitel 4 Implementierungen.....	43
4.1 Verwendete Werkzeuge und Software.....	43
4.2 Verwendete Java-Packages.....	44
4.3 Update-Service.....	44
4.3.1 Schadensaufnahme.....	44
4.3.1.1 Schadensfälle.....	45
4.3.1.2 Erweiterte Eingaben.....	46
4.3.1.3 Infos anzeigen.....	46
4.3.1.4 Provider-Implementierungen.....	47
4.3.2 Update-Service-Servlet.....	47
4.3.3 Generierung von neuen Zertifikaten.....	48
4.4 Beispiel-CAs.....	48
4.5 Datenbank.....	50
Anhang A Installation.....	53
A.1 Verzeichnisstruktur der Update-Service CD.....	53
A.2 Benötigte Komponenten.....	54
A.2.1 Java.....	54
A.2.2 Ant.....	54
A.2.3 MySQL.....	54
A.2.4 Certification-Authorities (CA).....	54
A.2.5 LDAP.....	55
A.2.6 Bezugsquellen.....	55
A.3 Notwendige Anpassungen.....	55
A.3.1 Pfade.....	55
A.3.2 Java.....	56
A.3.3 MySQL.....	56
A.3.4 LDAP.....	57
A.4 Update-Service.....	58
A.4.1 Compilierung.....	58
A.4.2 Ausführung.....	58

Anhang B UpdateManagementProtocol V2.....	61
B.1 UpdateComponent.....	61
B.2 UC - Definierte Aktionen.....	62
B.3 Komponenten.....	63
B.4 UpdateComponentCode.....	64
B.5 UpdateComponentResponse.....	64
Anhang C Registry.....	67
C.1 Die Registry im XML-Format.....	67
Anhang D Verzeichnisse.....	75
D.1 Literaturverzeichnis.....	75
D.2 Abbildungsverzeichnis.....	76

Hiermit versichere ich, die vorliegende Diplomarbeit 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 worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Bensheim, den 1.1.2004

Björn Seiffert

Kapitel 1 Einführung

Durch die zunehmende Vernetzung von Computern nimmt die Bedeutung elektronischer Kommunikationsformen immer mehr zu. Als Beispiele seien hier das bequeme Einkaufen über „Online-Shops“ (E-Commerce), virtuelle Behördengänge (E-Government) oder das Online-Banking genannt.

Allen gemein ist, dass die Kommunikation zumeist über offene, das heißt ungesicherte Netze wie zum Beispiel das Internet erfolgt. Dadurch steht man vor einer Reihe von Problemen: Wie kann man sicherstellen, dass

- der Empfänger auch wirklich derjenige ist, der er vorgibt zu sein (Authentizität),
- die Daten nur vom Sender und Empfänger gelesen werden (Vertraulichkeit),
- die Daten während der Übertragung nicht verändert wurden (Integrität),
- die Daten auch wirklich vom angegebenen Absender stammen (Verbindlichkeit).

Eine Lösung hierfür bietet die *Public-Key-Kryptographie*. Sie verwendet im Gegensatz zu klassischen Kryptoverfahren ein Schlüsselpaar bestehend aus einem geheimen (private) und einem öffentlichen (public) Schlüssel. Während der öffentliche Schlüssel jedermann zugänglich gemacht wird, darf der geheime Schlüssel nur dem Schlüsselinhaber bekannt sein. Für die Public-Key-Kryptographie gibt es zwei große Einsatzgebiete:

1. *Verschlüsselung von Daten:*

Hierzu wird der öffentliche Schlüssel des Kommunikationspartners dazu benutzt, um mit einem geeigneten Verschlüsselungsverfahren aus den zu verschickenden Daten einen verschlüsselten Text (Chiffretext) zu erzeugen. Diesen Chiffretext kann nur derjenige wieder entschlüsseln, der den dazugehörigen geheimen Schlüssel besitzt.

2. *Signatur von Daten:*

Die Daten werden aus Effizienzgründen erst einer öffentlich bekannten mathematischen Funktion unterzogen (einer Hashfunktion), die eine Art „Prüfsumme“ (den sog. Hashwert) berechnet. Das mathematische Verfahren dieser Prüfsumme muss dabei bestimmte Eigenschaften erfüllen. Beispielsweise muss es praktisch unmöglich sein, zu einer Prüfsumme eines Dokuments ein anderes Dokument mit gleicher Prüfsumme zu finden.

Auf die so erzeugte Prüfsumme wird dann eine weitere kryptographische Funktion (der Signaturalgorithmus) mit dem geheimen Schlüssel des Absenders angewendet. Das Ergebnis ist eine digitale Signatur des Absenders [X.509]. Diese wird zusammen mit den ursprünglichen Daten an den Empfänger gesendet. Die digitale Signatur kann wiederum nur mit dem korrespondierenden öffentlichen Schlüssel verifiziert werden.

Eine *Public-Key-Infrastruktur* (PKI) stellt das Fundament für jegliche Anwendung von Public Key Kryptographie dar und bezeichnet die Gesamtheit der Komponenten, Prozesse und Konzepte, die von Public-Key-Verfahren für elektronische Signatur und Verschlüsselung von Daten verwendet werden (vergl. Kapitel 2.1, [PKIX], [BUCH]). Grundlage der PKIs sind

mathematische Basisprobleme, von denen man annimmt, dass diese nicht mit vertretbarem Aufwand und vertretbarer Zeit zu lösen sind. Beispiele solcher Basisprobleme sind die Faktorisierung von großen Zahlen und die Berechnung von diskreten Logarithmen in bestimmten mathematischen Körpern.

Leider ist bis heute nicht beweisbar, ob diese Annahme auch wirklich stimmt. Vorstellbar sind somit Szenarien, in denen einzelne Basisprobleme gelöst werden, wodurch möglicherweise PKIs ihre Funktionalität verlieren können und eine Wiederherstellung viel Zeit und Geld kosten würde. Aber nicht nur das Lösen von kryptographischen Basisproblemen gefährdet die Funktionsfähigkeit von PKIs. Es kann auch vorkommen, dass Algorithmen fehlerhaft implementiert wurden oder sogar besser überhaupt nicht benutzt werden sollten, weil sich herausgestellt hat, dass sie nicht mehr kryptographisch sicher sind.

Aus diesem Grund ist es nötig, für solche Fälle Mechanismen zu entwickeln, die die Funktionalität und Vertrauenswürdigkeit von PKIs gewährleisten können. Das *Fail-Safe-Konzept* von Michael Hartmann und Sönke Maseberg [MaHa] beschreibt Möglichkeiten, PKIs so zu erweitern, damit sie im Schadensfall auch weiterhin funktionsfähig sind (vergl. Kapitel 2.2). Dieses Konzept definiert eine neue Datenstruktur, das *Update Management Protocol (UMP)*, mit der Teilnehmer von PKIs über eingetretene Schäden informiert werden können und das außerdem Maßnahmen beschreiben kann, die die Teilnehmer durchzuführen haben, damit die PKIs ihre Integrität nicht verlieren.

Aufgabe dieser Diplomarbeit war es, für das UMP eine grafische Oberfläche zu entwickeln (den Update-Service) die es einem Administrator erlaubt, auftretende Schäden aufzunehmen und das UMP zu initiieren. Außerdem sollte ein Programm entwickelt werden, das den weiteren Ablauf des Protokolls durchführt, die generierten Nachrichten zur späteren Kontrolle speichert und die notwendige Kommunikation mit den vorhandenen Zertifizierungs-Instanzen realisiert.

Kapitel 2 Das Fail-Safe-Konzept

2.1 Grundlagen der PKI

Ein auftretender Schaden kann für eine herkömmliche PKI gravierende Folgen nach sich ziehen:

- Digitalen Signaturen können ihre Beweiskraft verlieren.
- Revokationsmechanismen sind unter Umständen nicht mehr genügend gesichert.
- Verschlüsselungen können ihre Vertraulichkeit verlieren.
- Verfügbarkeit der PKI kann nicht mehr garantiert werden.

Der Nachteil einer PKI, die nur ein kryptographisches Verfahren verwendet, das ggf. auch noch fest in die PKI integriert ist, wird dabei deutlich sichtbar. Es ist demnach vorteilhafter, wenn bei der Implementierung ein anderer Ansatz gewählt wird und die Basiskomponenten austauschbar sind.

Genau dieser Ansatz wurde vom *Flexi-PKI*-Projekt am Lehrstuhl von Prof. Dr. Johannes Buchmann an der TU-Darmstadt gewählt [BuRT]. Ziel dieses Projektes ist es, alternative kryptographische Verfahren zu entwickeln und in existierende Anwendungen zu integrieren. In mehreren Teilprojekten wurden zum Beispiel neben Implementierungen von klassischen Verfahren wie RSA und 3DES auch alternative Verfahren entwickelt wie Elliptische Kurven oder Zahlkörper-Kryptographie. Ein weiterer Teilbereich beschäftigt sich mit dem „*Fail-Safe-Konzept*“, das hier nun näher diskutiert werden soll.

2.2 Das Konzept

Das Fail-Safe-Konzept [MaHa] erweitert die Flexi-PKI und besteht aus mehreren Teilen:

- Die Komponenten einer PKI sollen möglichst flexibel gehalten werden, so dass es möglich ist, sie bei Bedarf im laufenden Betrieb gegen andere auszutauschen. Dadurch soll ermöglicht werden, dass auf verändernde Sicherheitskriterien möglichst schnell reagiert werden kann. Das Fail-Safe-Konzept definiert hierfür ein neues Protokoll, das *Update-Management-Protocol* (UMP, siehe Kapitel 2.5).
- Der mögliche Austausch von Komponenten alleine stellt aber nicht die Funktionalität der PKI im Schadensfall sicher. Wenn ein Basisproblem gelöst wurde, dann kann den Signaturen, die auf diesem Problem beruhen, nicht mehr vertraut werden, weil sie fälschbar

geworden sind. Aus diesem Grund können die auszutauschenden Komponenten nicht mehr sicher übertragen werden.

Das Fail-Safe-Konzept unterteilt deshalb eine PKI in mindestens zwei Teil-PKIs, die nebeneinander existieren und von denen jede auf einem anderen Kryptoverfahren beruht. Diese Verfahren müssen mathematisch unabhängig voneinander sein. Jeder Teilnehmer einer solchen Fail-Safe-PKI besitzt Schlüssel und Zertifikate passend zu jeder Teil-PKI.

Was bedeutet in diesem Zusammenhang „unabhängig“? Damit bei einem Schaden nicht die gesamte PKI kompromittiert wird, müssen die verwendeten Verfahren auf verschiedenen mathematischen Basisproblemen aufbauen. Zum Beispiel kann eine Teil-PKI mit RSA-Verfahren arbeiten, die auf dem Faktorisierungsproblem für große natürliche Zahlen aufbauen und eine weitere Teil-PKI arbeitet mit Elliptische Kurven (Elliptic Curves, EC), die den diskreten Logarithmus in bestimmten endlichen Gruppen benutzen. Grund für diese Einschränkung ist die Annahme, dass die Wahrscheinlichkeit sehr gering ist, dass mehrere mathematische Basisprobleme gleichzeitig gelöst werden. Deshalb ist im Schadensfall immer nur eine Teil-PKI betroffen und die anderen bleiben unbeeinträchtigt.

Daten werden in einer Fail-Safe-PKI mit jedem einzelnen Signaturverfahren der Teil-PKIs signiert. Dadurch wird sichergestellt, dass immer mindestens eine Signatur gültig ist und ihre Beweiskraft erhalten bleibt.

2.3 Komponenten

Das Fail-Safe-Konzept legt nicht von vornherein fest, wieviele Teil-PKIs in einer Fail-Safe-PKI vorhanden sind. Es existiert nur die Bedingung, dass es mehr als eine sein muss. Im Rahmen dieser Diplomarbeit wird die Anzahl auf zwei festgelegt.

2.3.1 Teil-PKIs

Jeder Teil-PKI (PKIⁱ) ist ein Signaturverfahren σ^i zugeordnet, der auf einem anderen mathematischen Basisproblem α^i beruht. Dazu gibt es jeweils eine Zertifizierungsinstanz (Certification Authority, CAⁱ), die mit ihrem Schlüsselpaar (prK_{CA}^i , puK_{CA}^i) neue Zertifikate ausstellen kann. Der öffentliche Schlüssel puK_{CA}^i ist in einem speziellen Zertifikat, dem sog. Sicherheitsanker (Trust-Anchor, TAⁱ) gespeichert. Abgelaufene oder ungültig gewordene Zertifikate werden in Sperrlisten (Certificate Revocation List, CRLⁱ) abgelegt.

Jeder Zertifikat-Inhaber (Certificate Holder, CH_i) besitzt zwei Schlüsselpaare, die zu den Signaturverfahren der jeweiligen Teil-PKIs passen.

Einer Teil-PKI ist genau ein Signaturverfahren zugeordnet. Nur mit diesem Signaturverfahren werden die Zertifikate dieser Teil-PKI signiert. Die Schlüssel, die in diesen Zertifikaten gespeichert sind, müssen passend zu dem verwendeten Signaturverfahren sein. Benutzt zum Beispiel eine Teil-PKI das Signaturverfahren *SHA1withRSA*, so werden auch alle Zertifikate nur mit *SHA1withRSA* signiert und in diesen Zertifikaten nur *RSA*-Schlüssel gespeichert. Abbildung 2.1 zeigt den Schematischen Aufbau der PKI.

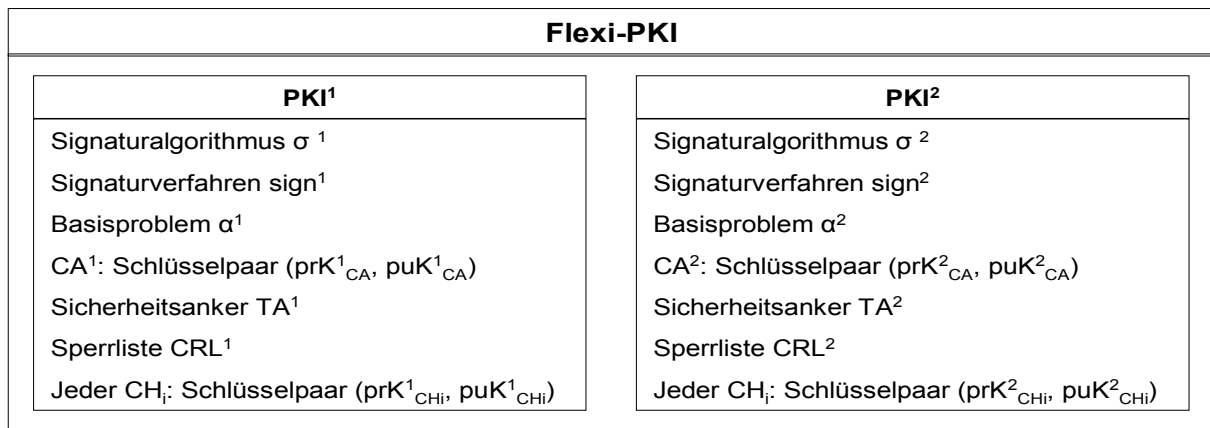


Abbildung 2.1: Fail-Safe-PKI

2.3.2 Update-Service

Wie oben bereits erwähnt wurde, soll es in einer Fail-Safe-PKI möglich sein, kryptographische Komponenten im laufenden Betrieb auszutauschen. Hierfür muss eine weitere Komponente geschaffen werden, der *Update-Service*. Er dient als zentrale Stelle, die mit den vorhandenen Teil-PKIs und den Teilnehmern der Fail-Safe-PKI kommuniziert und bei ggf. eintretenden Schadensfällen alle Betroffenen informiert sowie Gegenmaßnahmen einleitet und durchführt. Der Update-Service ist zentraler Bestandteil dieser Diplomarbeit und wird in Kapitel 3 diskutiert.

2.3.3 Outlook-Plugin

Für den sicheren Komponenten-Austausch ist es notwendig, dass auf der Seite der Teilnehmer der Fail-Safe-PKI spezielle Software zum Einsatz kommt, da die Kommunikation zwischen dem Update-Service und den Teilnehmern über ein neues Protokoll, dem *Update-Management-Protocol (UMP)* abgewickelt wird (s. Kapitel 2.5). Für das E-Mail-Programm Outlook wurde hierfür ein Plugin entwickelt [SCHRAMM], das um die UMP-Funktionalität erweitert wurde.

2.4 Schadensfälle

Ein Schadensfall ist ein Ereignis das eintreten kann und dadurch die Funktionsweise der Fail-Safe-PKI beeinträchtigen kann. In den Abschnitten 2.4.1 bis 2.4.7 werden die möglichen Schadensfälle mit den entsprechenden Maßnahmen zur Korrektur kurz vorgestellt.

2.4.1 CH-Schlüssel wurde kompromittiert

Ein einzelner geheimer CH-Schlüssel wurde kompromittiert und sollte nicht mehr verwendet werden.

- Das Zertifikat, in dem der zugehörige öffentliche Schlüssel gespeichert ist, muss revoziert werden.
- Der CH muss ein neues Schlüsselpaar generieren und dem Update-Service den neuen öffentlichen Schlüssel mitteilen.
- Der Update-Service leitet den neuen öffentlichen Schlüssel an die zugehörige RA weiter.
- Das neu erstellte Zertifikat wird veröffentlicht und dem CH zugesendet.

2.4.2 CA-Schlüssel wurde kompromittiert

Der geheime Schlüssel einer CA wurde kompromittiert und sollte nicht mehr verwendet werden.

- Der zugehörige Sicherheitsanker muss revoziert werden.
- Die CA generiert einen neuen Sicherheitsanker.
- Alle Zertifikate, die von der CA ausgestellt wurden, müssen revoziert werden.
- Da die in den Zertifikaten gespeicherten öffentlichen Schlüssel nicht von dem Schaden betroffen und damit noch gültig sind, brauchen die zugehörigen CHs keine neuen Schlüssel generieren. Es reicht aus, dass die CA die Schlüssel mit dem neuen Sicherheitsanker neu zertifiziert.
- Die neu erstellten Zertifikate werden veröffentlicht und an die jeweiligen CHs versendet.

2.4.3 Signaturverfahren wurde kompromittiert

Ein ganzes Signaturverfahren (SHA1withDSA, RidpMD160withRSA, etc.) wurde kompromittiert und sollte nicht mehr verwendet werden.

- Das Signaturverfahren muss bei allen CHs gelöscht werden.
- Alle CAs, die das Signaturverfahren benutzen, müssen deaktiviert und durch neue CAs ersetzt werden, die ein anderes Signaturverfahren benutzen.
- Alle durch deaktivierte CAs erstellten Zertifikate müssen revoziert werden.
- Die ersetzenden CAs stellen jeweils neue Zertifikate für die öffentlichen Schlüssel der revozierten Zertifikate aus.
- Die neu erstellten Zertifikate werden veröffentlicht und an die jeweiligen CHs versendet.

2.4.4 Hashfunktion ist unsicher geworden

Eine Hashfunktion (SHA1, MD5, etc.) ist unsicher geworden und sollte nicht mehr benutzt werden.

- Alle Signaturverfahren, die diese Hashfunktion verwenden, müssen bei den CHs gelöscht werden.

- Alle CAs, die ein so unsicher gewordenenes Signaturverfahren benutzen, müssen deaktiviert und durch neue CAs ersetzt werden.
- Alle Zertifikate der deaktivierten CAs müssen revoziert werden.
- Die ersetzenden CAs stellen neue Zertifikate für die öffentlichen Schlüssel der revozierten Zertifikate aus.
- Die neu erstellten Zertifikate werden veröffentlicht und an die jeweiligen CHs versendet.

2.4.5 Signaturalgorithmus wurde kompromittiert

Ein Signaturalgorithmus (RSA, DSA, etc.) wurde kompromittiert und sollte nicht mehr benutzt werden.

- Der Signaturalgorithmus muss bei den CHs gelöscht werden.
- Alle Signaturverfahren, die diesen Signaturalgorithmus verwenden, müssen bei den CHs gelöscht werden.
- Alle CAs, die ein so unsicher gewordenenes Signaturverfahren benutzen, müssen deaktiviert und durch neue CAs ersetzt werden.
- Alle durch deaktivierte CAs erstellten Zertifikate müssen revoziert werden.
- Die ersetzenden CAs stellen jeweils neue Zertifikate für die öffentlichen Schlüssel der revozierten Zertifikate aus.
- Die neu erstellten Zertifikate werden veröffentlicht und an die jeweiligen CHs versendet.

2.4.6 Paddingverfahren wurde kompromittiert

Ein Paddingverfahren wurde kompromittiert und sollte nicht mehr benutzt werden.

- Alle Signaturverfahren, die das Paddingverfahren verwenden, müssen bei den CHs gelöscht werden.
- Alle CAs, die ein so unsicher gewordenenes Signaturverfahren benutzen, müssen deaktiviert und durch neue CAs ersetzt werden.
- Alle durch deaktivierte CAs erstellten Zertifikate müssen revoziert werden.
- Die ersetzenden CAs stellen jeweils neue Zertifikate für die öffentlichen Schlüssel der revozierten Zertifikate aus.
- Die neu erstellten Zertifikate werden veröffentlicht und an die jeweiligen CHs versendet.

2.4.7 Mathematisches Basisproblem wurde gelöst

Ein Mathematisches Basisproblem (z.B. Faktorisierung großer natürlicher Zahlen) wurde gelöst.

- Alle kryptographischen Algorithmen, die auf diesem Basisproblem beruhen, müssen bei den Clients gelöscht werden.
- Alle CAs, die ein Signaturverfahren benutzen, das durch das gelöste Basisproblem unsicher geworden sind, müssen deaktiviert und durch neue CAs ersetzt werden.
- Alle durch deaktivierte CAs erstellten Zertifikate müssen revoziert werden.
- Die betroffenen CHs müssen neue Schlüssel erzeugen, die passend zu den Verfahren der ersetzenden CAs sind.
- Die ersetzenden CAs stellen jeweils neue Zertifikate für die neu erstellten öffentlichen Schlüssel aus.
- Die neu erstellten Zertifikate werden veröffentlicht und an die jeweiligen CHs versendet.

2.5 Das Update Management Protocol (UMP)

Tritt ein Schaden ein, der die Funktionalität einzelner Komponenten der Flexi-PKI beeinträchtigt, so wird das *Update Management Protocol (UMP)* abgewickelt. Das UMP ist als mehrstufiges Protokoll definiert, das je nach Schadensart in mehreren Iterationen durchgeführt wird. Es definiert eine Datenstruktur die es erlaubt, alle Teilnehmer einer Flexi-PKI über die Art des aufgetretenen Schadens zu informieren und eine Reihe von Maßnahmen zu beschreiben, die die Teilnehmer durchzuführen haben, um die volle Funktionalität der PKI wiederherzustellen.

Da das UMP nicht Gegenstand dieser Diplomarbeit ist, soll diese Thematik hier nicht in aller Vollständigkeit vorgestellt werden. Für eine ausführliche Beschreibung der UMP Datenstruktur und seiner Implementierung siehe [Hart], [KALE]. Die verwendete Definition des UMP steht im Anhang B.

2.5.1 Komponenten des UMP

An dieser Stelle kann nicht auf jede einzelne Komponente des UMPs eingegangen werden. Es werden hier nur ausgewählte Komponenten genannt und näher beschrieben.

2.5.1.1 Komponenten, die den Benutzer näher spezifizieren

- **Zertifikat-Inhaber (Certificate Holder, CH):**
Ein Zertifikat-Inhaber ist die Instanz, für die Zertifikate von einer CA ausgestellt werden. Meist sind das natürliche Personen.
- **Client:**
Ein Client ist die Oberbezeichnung für sowohl einen Autonomen wie einen Nicht-Autonomen Client (s.u.).
- **Autonomer Client (Autonomous Client):**
Ausprägung einer persönlichen Sicherheitsumgebung von Clients, die eine

Internetverbindung aufbauen und direkt mit dem Zertifikat-Inhaber kommunizieren kann (z.B. ein Computer)

- **Nicht-autonomer Client (Non-Autonomous Client):**
Ausprägung einer persönlichen Sicherheitsumgebung von Clients, die nicht selbstständig Kommandos entgegennehmen und auf diese reagieren kann. Kann nicht selbsttätig eine Internetverbindung aufbauen und mit dem Zertifikat-Inhaber kommunizieren (z.B. Chipkarte)
- **ClientIdentifier:**
Clients können zu Gruppen zusammengefasst werden. Jede Gruppe wird durch einen ClientIdentifier beschrieben (siehe Abbildung 2.3). In diesem Datenkonstrukt stehen Informationen der Gruppe von Clients wie z.B., ob ein Client autonom oder nicht-autonom ist, oder die Bezeichnung. Dadurch kann der Update-Service dem Client einen Provider (siehe Kapitel 2.5.1.6) zuordnen. Aus diesem Grund wird einem Client ein spezielles UpdateComponent zugesendet, in dem sein ClientIdentifier erfragt wird (siehe Kapitel 2.5.1.3).

ClientIdentifier	
Client-Classification	Autonomer- oder Nicht-Autonomer Client
Client-ID	Name der Gruppe von Clients
Client-Version	Version der Gruppe von Clients
Client-Serialnummer	Optionale Seriennummer

Abbildung 2.2: Schematischer Aufbau eines ClientIdentifiers

2.5.1.2 Registry

Der Update-Service muss wissen, welche kryptographischen Algorithmen und mathematischen Basisprobleme von der Flexi-PKI unterstützt werden. Alle diese Komponenten werden in der *Registry* (siehe Abbildung 2.3) gespeichert und können bei Bedarf abgerufen werden. Des weiteren sind in der Registry Abhängigkeiten zwischen den einzelnen Einträgen gespeichert. So kann über die Registry abgefragt werden, ob zwei kryptographische Algorithmen voneinander abhängen und welche Algorithmen auf einem bestimmten mathematischen Basisproblem aufbauen.

Die Registry ist ein ASN.1 Datenkonstrukt, die aus einer speziellen XML-Datei aufgebaut werden kann.

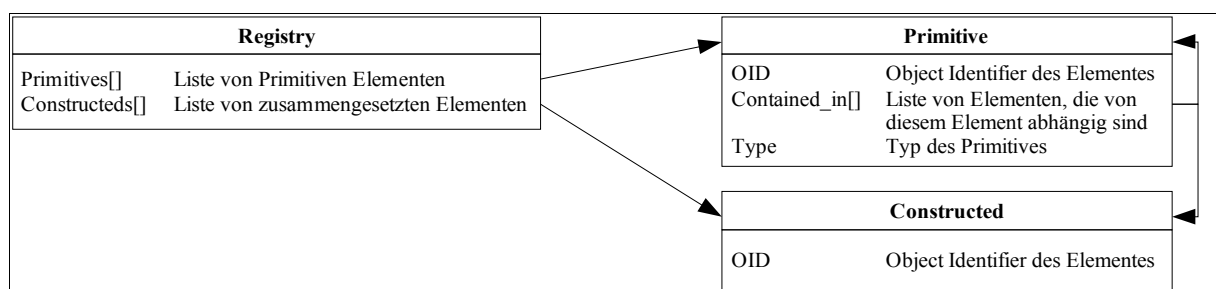


Abbildung 2.3: Schematischer Aufbau der Registry

2.5.1.3 UpdateComponent (UC)

Ein UpdateComponent ist ein ASN.1 Datenkonstrukt, welches vom Update-Service generiert und an die Clients der Zertifikat-Inhaber verschickt wird.

Auf oberster Ebene des UCs werden hauptsächlich Verwaltungsinformationen gespeichert: Z.B. die Version des verwendeten UMPs, die URL, an die die Clients ihre Antworten auf die UCs schicken und eine eindeutige Nummer für Verwaltungszwecke.

In ein UpdateComponent werden alle Anweisungen, die ein Client durchzuführen hat als einzelne Aktionen gekapselt. Abbildung 2.4 zeigt den schematischen Aufbau eines UCs. Für Autonome Clients wird ein *AutonomousClientRelevant*-, für nicht-autonome Clients wird ein *NonAutonomousClientRelevant*-Objekt erzeugt und in das UpdateComponent eingehängt. In diesen Objekten stehen neben den Adressaten die von den Clients durchzuführenden Aktionen.

Soll von einem Client der ClientIdentifier erfragt werden, so wird ein *GetClientIdentifier*-Objekt in das UpdateComponent eingefügt. Der Client wird darin über sein Zertifikat identifiziert.

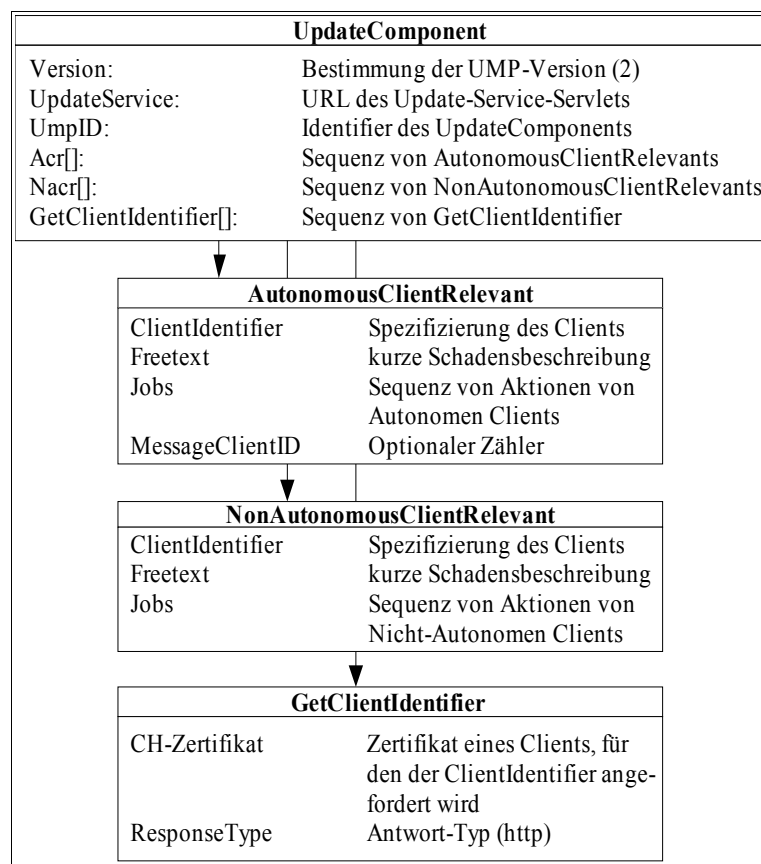


Abbildung 2.4: Schematischer Aufbau eines UpdateComponents

2.5.1.4 Das UpdateComponentResponse (UCR)

Ein UpdateComponentResponse ist ein ASN.1 Datenkonstrukt, welches von den Clients generiert und als Antwort auf ein UpdateComponent an den Update-Service zurück geschickt wird. Jede in einem UpdateComponent gespeicherte Aktion wird vom Client beantwortet, um

dem Update-Service zu berichten, welche Aktion erfolgreich ausgeführt wurde und welche nicht. Ein UpdateComponentResponse wird auch von den Clients dazu verwendet, dem Update-Service auf Anfrage seinen ClientIdentifier mitzuteilen.

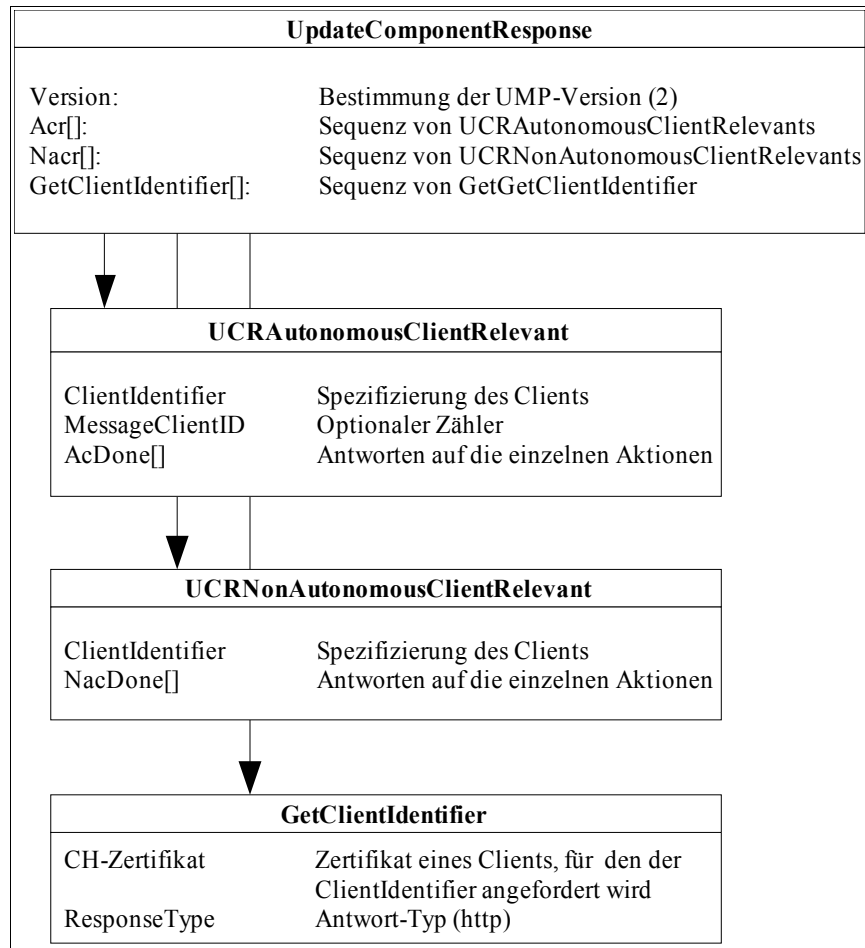


Abbildung 2.5: Schematischer Aufbau eines UpdateComponentResponses

2.5.1.5 Sicherheitsanker (Trust-Anchor, TA)

Jede CA besitzt ein Zertifikat von einem Schlüsselpaar, mit dem sie alle von ihr ausgestellten Zertifikate signieren kann. Dieses Zertifikat wird als Sicherheitsanker (TA) bezeichnet. Mit diesem TA kann außerdem die Authentizität eines UpdateComponent oder UpdateComponentCode überprüft werden.

2.5.1.6 Provider

Neu zu installierende Algorithmen werden in Providern zusammengefasst. Ein Provider umfasst eine Menge von kryptographischen Algorithmen, die auf die Clients eines speziellen ClientIdentifiers zugeschnitten sind. Beispiele hierfür sind der *Elliptic-Curves-* und *Number-Field-Provider*, die am Lehrstuhl von Prof. Dr. Johannes Buchmann an der TU-Darmstadt entwickelt wurden [FLEXI].

2.5.1.7 UpdateComponentCode (UCC)

Ein UCC ist ein Datenkonstrukt, welches die Implementierung eines Providers für einen speziellen ClientIdentifier kapselt.

Da die Clients völlig voneinander verschiedene Implementierungen besitzen können, muss nicht jeder Provider auf jedem Client lauffähig sein. Deshalb wird durch ein UCC-Objekt eine Verbindung zwischen Providern und ClientIdentifiern hergestellt. Außerdem besitzt ein UCC eine UCC-ID, die die Art des Providers beschreibt. Abbildung 2.6 zeigt beispielhaft den Zusammenhang zwischen UCC, ClientIdentifier und Provider.

Soll ein Client einen bestimmten Provider installieren, so wird ihm in einem Update-Component nur die UCC-ID des Providers mitgeteilt und er lädt sich den Code zu einem späteren Zeitpunkt herunter. Das hat den Vorteil, dass zum einen der Umfang des Update-Components verringert wird und zum anderen die Netzlast zeitlich gestreckt wird, wenn mehrere Clients den Code laden.

<i>Name</i>	<i>UCC-ID</i>	<i>ClientIdentifier</i>	<i>Provider</i>
Elliptic Curves	ID ₁	Cl-ID ₁	P ₁₁
		⋮	⋮
		Cl-ID _n	P _{1n}
Number Field	ID ₂	Cl-ID ₁	P ₂₁
		⋮	⋮
		Cl-ID _n	P _{2n}

Abbildung 2.6: Zusammenhang zwischen UpdateComponentCode und Provider

Kapitel 3 Update-Service

Ziel dieser Diplomarbeit ist es, einen Update-Service zu entwickeln, der einem Administrator die Möglichkeit bietet, eingetretene Schäden aufzunehmen und das Update-Management-Protokoll einzuleiten und durchzuführen. Der Update-Service kommuniziert dabei je nach Notwendigkeit mit den CAs der Teil-PKIs sowie den Clients und informiert diese über durchzuführende Schritte.

3.1 Ablauf im Schadensfall

Wenn ein Schaden eintritt, dann müssen je nach Art des Schadens mehrere Phasen durchlaufen werden. Diese Phasen werden im Folgenden vorgestellt und diskutiert. Der schematische Ablauf im Schadensfall ist in Abbildung 3.1 dargestellt.

- **Phase 0: Schadensaufnahme.**

Der Administrator des Update-Services erhält die Information, dass ein Schaden eingetreten ist. Quelle für diese Information können zum Beispiel das Bundesamt für Sicherheit in der Informationstechnik [BSI], wissenschaftliche Abhandlungen, Teilnehmer oder sonstige Quellen sein. Möglich sind die in Kapitel 2.4 bereits vorgestellten Schadensfälle:

- i. Schaden S1: CH-Schlüssel wurde kompromittiert.
- ii. Schaden S2: CA-Schlüssel wurde kompromittiert.
- iii. Schaden S3: Signaturverfahren wurde kompromittiert.
- iv. Schaden S4: Hashfunktion ist unsicher geworden.
- v. Schaden S5: Signaturalgorithmus wurde kompromittiert.
- vi. Schaden S6: Paddingverfahren wurde kompromittiert.
- vii. Schaden S7: Mathematisches Basisproblem wurde gelöst.

- **Phase 1: Revokation.**

Je nach Schaden kann es vorkommen, dass Zertifikate ungültig werden. Dies kann einzelnen CH-Zertifikaten (Schaden S1) oder gleich mehreren CH-Zertifikaten passieren, wenn ein CA-Sicherheitsanker (z.B. Schaden S2) gelöscht wird. Auch kann es vorkommen, dass in dieser Phase überhaupt kein Zertifikat revoziert werden muss (z.B. Schaden S3, wenn das betroffene Signaturverfahren von keiner CA verwendet wird).

- **Phase 2: Einholung benutzerspezifischer Daten.**
Ein Client kann ein Autonomer oder ein Nicht-Autonomer Client sein (s. Kapitel 2.5.1.1). Diese Information ist wichtig, weil je nach Beschaffenheit des Clients andere Arten von UpdateComponents generiert werden müssen. Da diese Information nicht schon bei der Registrierung durch die RA eingeholt wird (sie ist nur für den Update-Service von Nutzen), muss sie einmalig vor dem Erzeugen des allerersten UpdateComponents vom Client erfragt werden.
- **Phase 3: Update-Service leitet schadensabhängige Maßnahmen ein.**
Abhängig vom Schaden werden verschiedene UpdateComponents zusammengebaut und an die betroffenen Clients per E-Mail verschickt. Je nach Schadensart werden unterschiedliche Aktionen in ein UpdateComponent integriert und zum Schluss mit zwei Signaturen gesichert. In Kapitel 3.2 wird gezeigt, wie solche Schäden aufgenommen werden.
- **Phase 4: Update-Service verschickt UpdateComponent.**
Nachdem alle für den Schadensfall nötigen UpdateComponents generiert wurden, wird das erste UpdateComponent an die betroffenen Clients verschickt. Dazu verwendet der Update-Service E-Mails mit dem jeweiligen UpdateComponent als Anhang.
- **Phase 5: Client empfängt und führt UpdateComponent aus.**
Der Client empfängt das UpdateComponent per E-Mail und überprüft die Signaturen des UpdateComponents und führt dann die für ihn in dem UpdateComponent gespeicherten Aktionen aus. Zu jeder durchgeführten Aktion wird eine Antwort generiert, die alle zusammen in einem UpdateComponentResponse zurück an den Update-Service geschickt werden.
- **Phase 6: Update-Service empfängt und führt UpdateComponentResponse aus.**
Der Update-Service empfängt die UpdateComponentResponses der Clients und wertet sie aus. Die in den UpdateComponentResponses gespeicherten Ergebnisse werden zur späteren Kontrolle gespeichert. Je nach Schadensfall kann es nun vorkommen, dass weitere UpdateComponents verschickt werden müssen.
- **Phase 7: Austausch von Komponenten.**
Wenn die Clients kryptographische Komponenten ersetzen müssen, dann wird der Code dieser Komponenten nicht direkt in einem UpdateComponent versendet, sondern es wird den Clients nur mitgeteilt, wo er herunterzuladen ist. Das hat drei Vorteile:
 1. Der Umfang der UpdateComponents wird verringert. Dies ist insbesondere vorteilhaft, wenn durch ein UpdateComponent mehrere unterschiedliche Clients benachrichtigt werden, die unterschiedliche Codes benötigen.
 2. Die Netzlast wird zeitlich gestreckt, weil nicht alle Clients den Code gleichzeitig herunterladen.
 3. Jeder Client lädt sich nur denjenigen Code herunter, der genau für ihn passend ist.

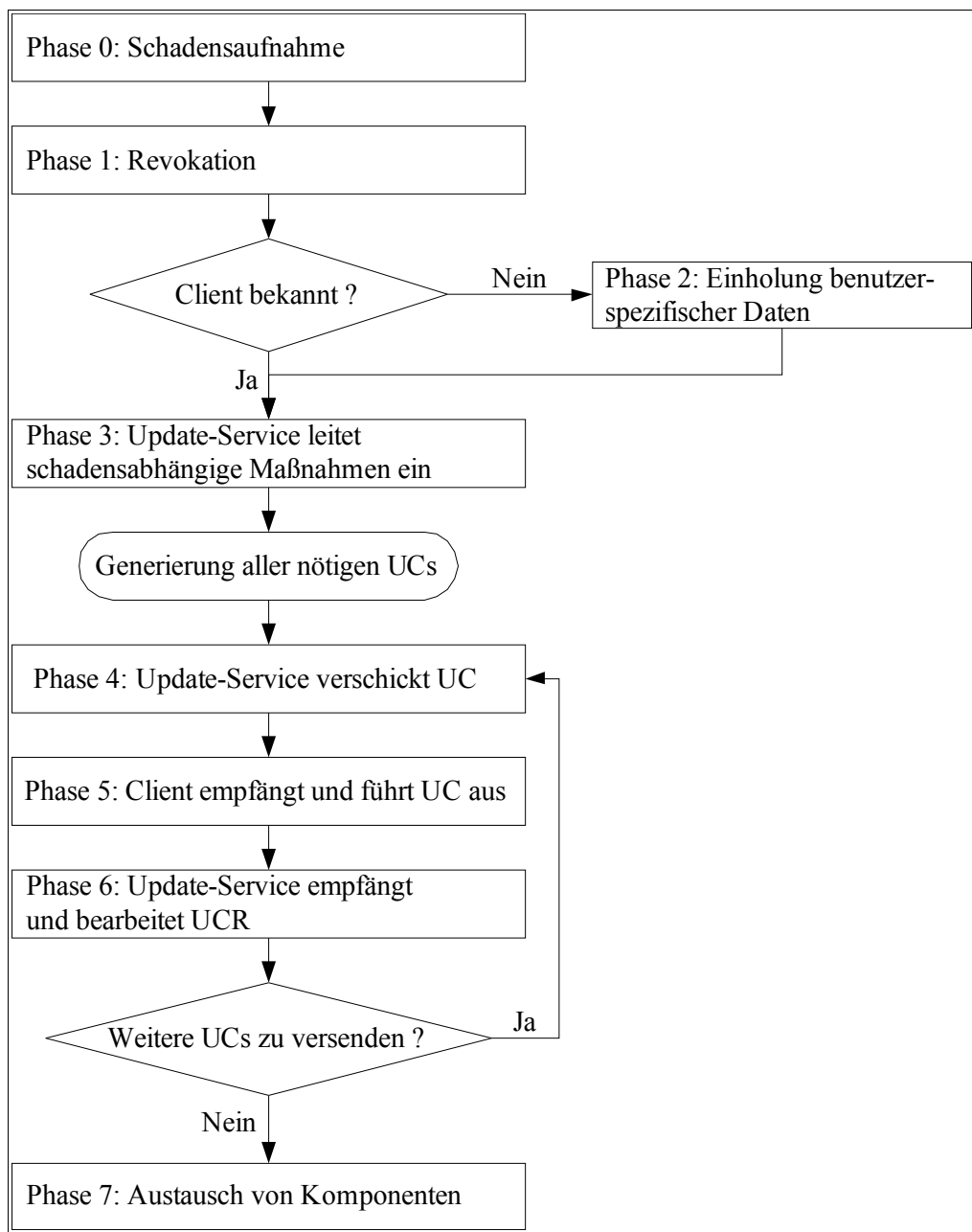


Abbildung 3.1: Schematischer Ablauf eines Schadensfalles

3.2 Komponenten des Update-Service

Der Update-Service besteht aus mehreren Komponenten, auf deren Eigenschaften in diesem Kapitel näher eingegangen wird. Abbildung 3.2 zeigt den Zusammenhang der einzelnen Komponenten, die entweder zum Update-Service gehören, oder die von ihm verwendet werden. Komponenten sind in der Abbildung durch Rechtecke mit durchgezogenen Linien und Daten durch Rechtecke mit gestrichelten Linien dargestellt. Die Komponenten innerhalb

des besonders markierten Bereichs bilden den Update-Service. Durchgezogene Verbindungspfeile symbolisieren den möglichen Datenaustausch bzw. die Kommunikation zwischen verschiedenen Komponenten, gestrichelte Pfeile deuten den Weg an, den bestimmte Daten nehmen können.

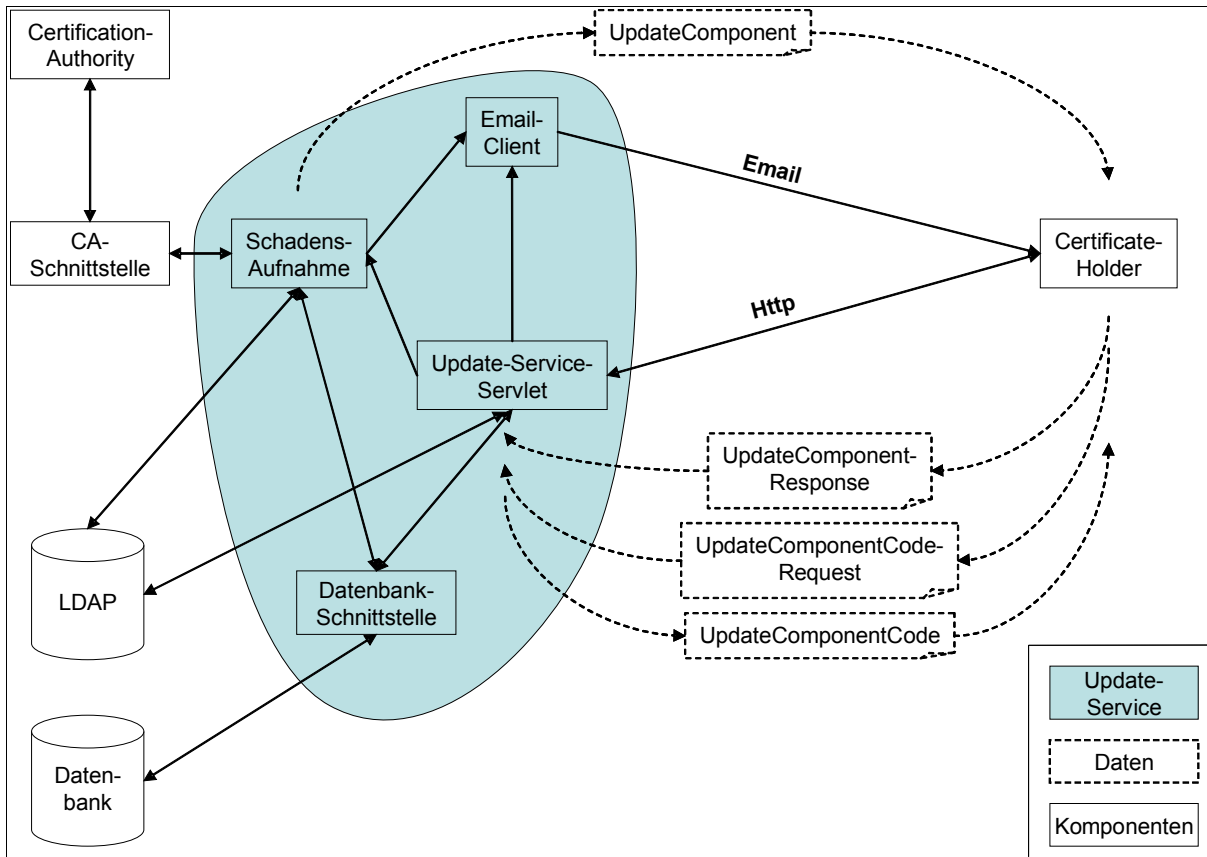


Abbildung 3.2: Zusammenhang zwischen den Komponenten des Update-Service

- Update-Service-Servlet**
 Alle Antworten der Clients werden von einem speziellen Servlet empfangen und beantwortet. Hierfür wird ein Http-Server benötigt, der auf eingehende Http-Verbindungen reagiert und bei einem Verbindungswunsch zu dem Update-Service-Servlet das Servlet startet und die Verbindung weiterleitet.
- E-Mail-Client**
 Ein einfacher E-Mail-Client wird verwendet, um UpdateComponents an die Empfänger zu versenden oder um Nachrichten dem Administrator des Update-Service zukommen zu lassen.
- Datenbank-Schnittstelle**
 Alle relevanten Daten, die der Update-Service benötigt, werden in einer Datenbank gespeichert. (siehe Kapitel 4.5) Hierfür wird eine MySQL-Datenbank verwendet und eine Datenbank-Schnittstelle bietet dem Update-Service Zugriff darauf.
- Schadensaufnahme**
 Für die Schadensaufnahme benutzt der Update-Service ein graphisches Frontend, mit dem die Schadensfälle bestimmt und festgelegt werden. Diese Komponente baut danach auch

die notwendigen UpdateComponents zusammen und übergibt sie dem E-Mail-Client zum Verschicken.

3.3 Bedienung

In diesem Kapitel wird die Bedienung des Update-Service erläutert. Die wichtigsten Schritte werden durch Screenshots dokumentiert. Es werden hierbei nur die für den aktuellen Zusammenhang nötigen Teile abgebildet.

3.3.1 Status-Fenster

Nach dem Programmstart erscheint als erstes das Status-Fenster (siehe Abbildung 3.3).

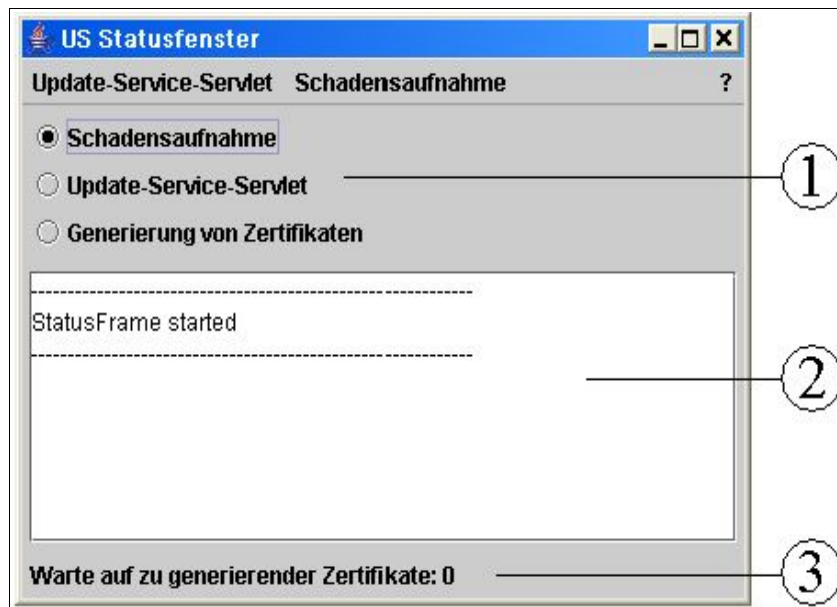


Abbildung 3.3: Status-Fenster

Hier gibt es einen Textbereich (2), der alle Status-Meldungen anzeigt, die die einzelnen Komponenten ausgeben. Der Übersichtlichkeit halber werden die Meldungen getrennt nach Komponenten angezeigt. Mit Hilfe der Radio-Buttons (1) können, die jeweiligen Meldungen ausgewählt werden:

- **Schadensaufnahme**
Die Meldungen der Schadensaufnahme werden angezeigt. Das sind vor allem Informationen, die die Generierung der Update-Components betreffen. Eventuelle Fehlermeldungen werden hier auch entsprechend angezeigt.
- **Update-Service-Servlet**
Jeder Verbindungswunsch der Clients mit dem Update-Service-Servlet wird protokolliert. In diesem Protokoll stehen dann Uhrzeit, IP-Adresse des Senders und Fortschritt der Bearbeitung.

- **Generierung von Zertifikaten**

Es kann vorkommen, dass neue Benutzer-Zertifikate erstellt werden müssen. In diesem Fall werden der gewünschten CA die nötigen Informationen übergeben und die CA generiert das Zertifikat. Da das Generieren von Benutzer-Zertifikaten unabhängig vom Update-Service ist, ist der Zeitbedarf dieses Vorgangs unbestimmt. Er könnte sogar je nach Implementierung der CA mehrere Tage dauern. Deshalb läuft im Hintergrund ein weiterer Prozess des Update-Service, der in regelmäßigen Abständen bei der jeweiligen CA nachfragt, ob die gewünschten Zertifikate schon fertig sind. Alle Ereignisse, die hiermit zu tun haben, werden in dieses Textfeld geschrieben.

In der Status-Zeile (3) wird die Anzahl der noch von den CAs zu generierenden Zertifikate angezeigt. Über die Menüzeile können das Update-Service-Servlet und die Schadensaufnahme gestartet werden.

3.3.2 Schadensaufnahme

Ist ein Schaden eingetreten über den die Clients informiert werden müssen, so wird die Schadensaufnahme gestartet (siehe Abbildung 3.4). Hier hat der Administrator die Möglichkeit, über verschiedene Reiter (1) die Schadensaufnahme einzuleiten, erweiterte Eingaben zu machen (siehe Kapitel 3.3.5) oder Informationen zu den Schadensverläufen abzurufen (siehe Kapitel 3.3.6). Bei der Auswahl eines Schadensfalles (2), erscheint eine kurze Beschreibung mit den durchzuführenden Aktionen in dem Textfenster (3).

Die Schadensaufnahme führt durch diverse Fenster, in denen der Schaden und die durchzuführenden Aktionen immer genauer vom Administrator zu spezifizieren sind. Zu jeder Zeit befinden sich am unteren Fensterrand drei Schaltflächen, mit deren Hilfe zwischen diesen Fenstern navigiert werden kann (4). Folgende Optionen stehen zur Verfügung:

- **Zurück** Schaltet zurück zum vorherigen Fenster.
- **Weiter** Schaltet weiter zum nächsten Fenster.
- **Abbrechen** Bricht die Schadensaufnahme ab und schaltet zurück zum Startfenster.

Die Schadensaufnahme läuft für jeden Schaden nach dem gleichen Muster ab:

1. Auswahl des Schadensfalls.
2. Evtl. Auswahl von betroffenen Zertifikaten bzw. Algorithmen.
3. Spezifizieren und Ausfüllen der durchzuführenden Aktionen.
4. Auswahl der zu erzeugenden Signaturen.
5. Auswahl der Clients, die benachrichtigt werden sollen.
Dieser Schritt dient hauptsächlich zu Testzwecken, wenn nur ganz bestimmte Clients angesprochen werden sollen. Im Normalfall wird dieser Schritt einfach durch Klick auf „Weiter“ übersprungen.
6. Evtl. weitere Spezifikation von UpdateComponentCodes.
7. Generierung und Verschickung der notwendigen UpdateComponents.

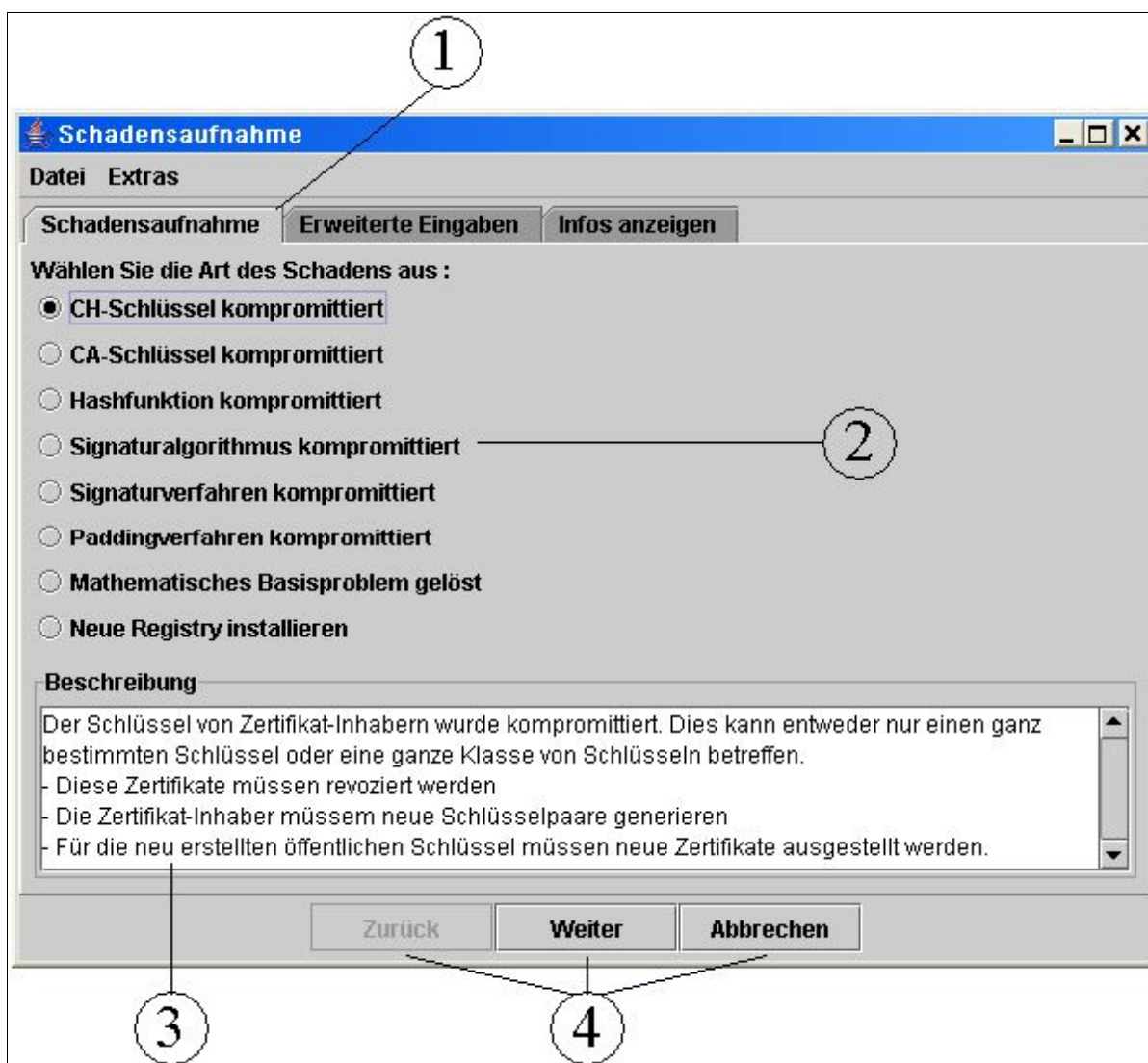


Abbildung 3.4: Auswahl des eingetretenen Schadens

3.3.2.1 CH-Schlüssel kompromittiert

Bei diesem Schadensfall gibt es zwei Möglichkeiten.

1. Ein ganz bestimmter CH-Schlüssel wurde kompromittiert, ist nicht mehr sicher und muss ausgetauscht werden.
 Hier wird das betroffene Zertifikat aus einer Liste von allen im LDAP-Verzeichnis [LDAP] gespeicherten Zertifikaten ausgewählt (siehe Abbildung 3.5). Über die Schaltfläche (1) in der Abbildung kann der Administrator sich das ausgewählte Zertifikat anzeigen lassen. Dieses Zertifikat wird revoziert und der zugehörige Benutzer muss es bei sich ebenfalls löschen. Dafür generiert der Benutzer ein neues Schlüsselpaar, für das eine neue Schlüssellänge anzugeben ist.
2. Eine Gruppe von CH-Schlüsseln darf nicht mehr verwendet werden und muss durch neue ersetzt werden.
 Dieser Schadensfall tritt z.B. immer dann auf, wenn vom BSI neue Empfehlungen herausgegeben werden, welche Schlüssellängen für bestimmte Signatur-Algorithmen verwendet

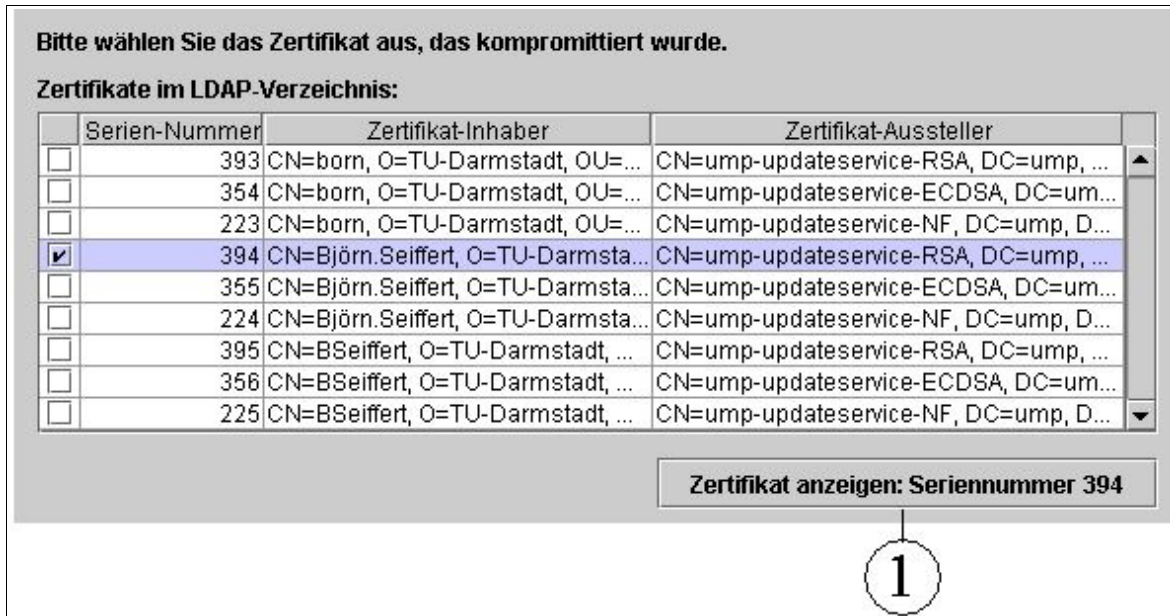


Abbildung 3.5: Auswahl eines CH-Zertifikats

werden sollen.

Um die Schadensbeschreibung auszufüllen, wird der Algorithmus ausgewählt, der betroffen ist. Dazu kann optional eine „KeyUsage“ und eine Schlüssellänge angegeben werden, die die Schlüssel näher beschreiben, die gelöscht werden sollen. Danach muss die neue Schlüssellänge der zu generierenden Schlüssel eingegeben werden. Optional ist auch hier die Angabe eine Key-Usage möglich.

3.3.2.2 CA-Schlüssel kompromittiert

Hierfür wird zuerst eine Auswahlliste von allen bekannten CAs angezeigt, aus der der betroffene CA-Schlüssel ausgewählt wird (siehe Abbildung 3.6). Der Sicherheitsanker der ausgewählten CA wird dann verwendet, um die durchzuführenden Aktionen korrekt auszufüllen.

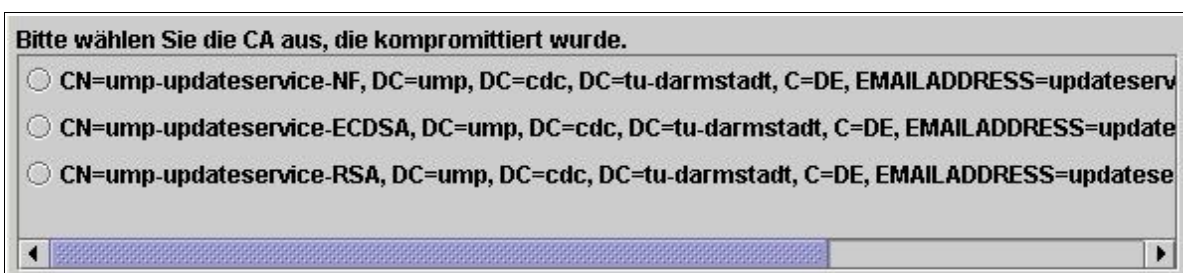


Abbildung 3.6: CA-Zertifikat auswählen

3.3.2.3 Hashfunktion kompromittiert

Bei diesem Schadensfall ist zuerst die betroffene Hashfunktion auszuwählen. Dem Administrator wird eine Auswahlliste von allen in der Registry aufgeführten Hashfunktionen angeboten (siehe Abbildung 3.7).

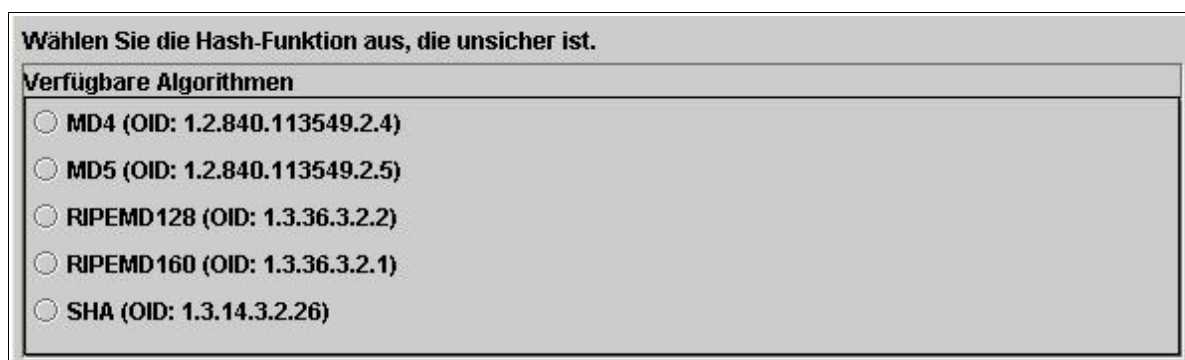


Abbildung 3.7: Hashfunktion auswählen

Es werden dann aus der Registry alle Algorithmen heraus gesucht, die die kompromittierte Hashfunktion verwenden. Diese Algorithmen sowie die Hashfunktion müssen gelöscht werden. Verwendet eine CA eine dieser zu löschenden Signaturverfahren, so ist auch sie betroffen, muss deaktiviert und durch eine neue CA ersetzt werden. Dadurch sind Benutzer-Zertifikate betroffen, die auch entsprechend zu löschen sind. Diese Benutzer müssen neue Schlüssel, passend zu der neuen CA generieren.

3.3.2.4 Signaturalgorithmus kompromittiert

Als Erstes ist der betroffene Signaturalgorithmus auszuwählen. Hierzu wird eine Auswahlliste mit allen in der Registry gespeicherten Algorithmen angezeigt (siehe Abbildung 3.8).

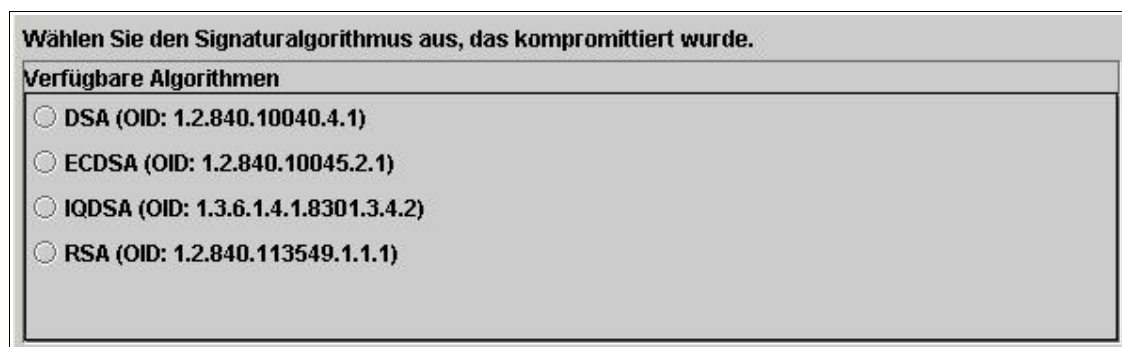


Abbildung 3.8: Signaturalgorithmus auswählen

Wurde ein Signaturalgorithmus gewählt, so wird dieser zusammen mit allen weiteren Algorithmen, die ihn verwenden, als zu löschen gekennzeichnet. Benutzt eine CA einen der zu löschenden Algorithmen, so wird auch die CA deaktiviert und durch eine andere ersetzt. Dadurch können wieder Benutzer-Zertifikate betroffen sein und müssen dementsprechend gelöscht werden. Die Benutzer müssen neue Schlüssel für das durch die neue CA unterstützte Verfahren generieren.

3.3.2.5 Signaturverfahren kompromittiert

Eine Liste von allen in der Registry vorhandenen Signaturverfahren wird als Erstes bei diesem Schadensfall angezeigt, aus dem das betroffene Verfahren ausgewählt wird (siehe Abbildung 3.9).

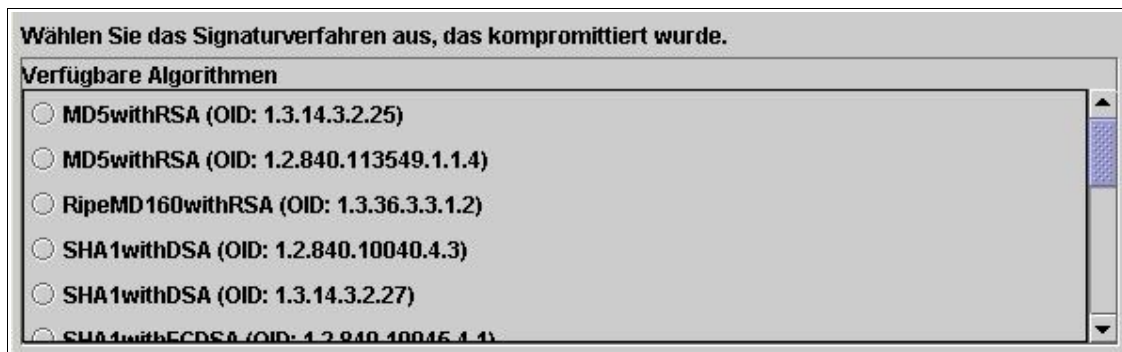


Abbildung 3.9: Signaturverfahren auswählen

Verwendet eine CA dieses Signaturverfahren, so wird sie deaktiviert und durch eine andere CA ersetzt. Außerdem sind die Benutzer-Zertifikate, die von der zu löschende CA ausgestellt wurden, zu löschen. Die betroffenen Benutzer müssen neue Schlüssel, passend zum von der neuen CA unterstützten Verfahren generieren.

3.3.2.6 Paddingverfahren kompromittiert

Da bisher noch keine OID für einzelne Paddingverfahren existieren, kann keine nur mit Paddingverfahren gefüllte Auswahlliste angeboten werden. Statt dessen werden in einer Liste alle in der Registry vorhandenen Algorithmen angezeigt, aus dem der Administrator den Algorithmus auswählt, der das kompromittierte Paddingverfahren benutzt. Dieser Algorithmus wird als zu löschen gekennzeichnet.

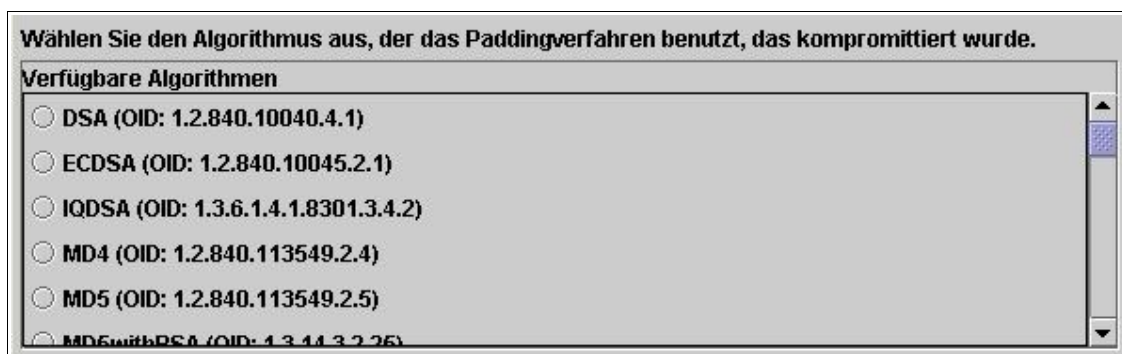
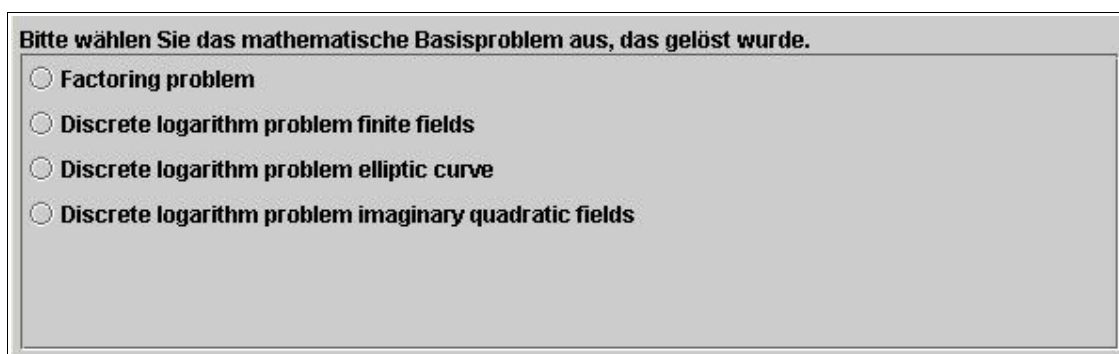


Abbildung 3.10: Algorithmus mit dem kompromittierten Paddingverfahren auswählen

Wie auch schon bei den anderen Schadensfällen kann dadurch ein Signaturalgorithmus betroffen sein, der von einer CA benutzt wird. Ist das der Fall, so muss auch diese CA mitsamt den von ihr ausgestellten Benutzer-Zertifikaten deaktiviert werden. Für die CA wird eine andere CA eingesetzt, wofür die betroffenen Benutzer neue Schlüssel generieren müssen.

3.3.2.7 Mathematisches Basisproblem gelöst

Bei diesem Schadensfall wird zuerst das Basisproblem ausgewählt, das nun nicht mehr als ausreichend „schwierig“ zu lösen angesehen wird (siehe Abbildung 3.11).



Bitte wählen Sie das mathematische Basisproblem aus, das gelöst wurde.

- Factoring problem
- Discrete logarithm problem finite fields
- Discrete logarithm problem elliptic curve
- Discrete logarithm problem imaginary quadratic fields

Abbildung 3.11: Mathematisches Basisproblem auswählen

Dadurch müssen sämtliche Algorithmen, die auf diesem mathematischen Basisproblem beruhen, gelöscht werden. Ist unter den zu löschenden Algorithmen ein Signaturverfahren, das von einer CA benutzt wird, so ist diese zu deaktivieren und alle von ihr ausgestellten Zertifikate sind zu löschen. Alle betroffenen Benutzer müssen neue Schlüssel, passend zur neuen CA generieren.

3.3.2.8 Neue Registry installieren

Der Update-Service sieht vor, dass neue Provider in die PKI integriert werden können. Da Provider eine Ansammlung von kryptographischen Algorithmen sind, können durch neue Provider auch neue Algorithmen in die PKI eingebracht werden. Kennt der Update-Service von diesen Algorithmen die zugehörigen OIDs noch nicht, so müssen sie ihm bekannt gemacht werden. Der Administrator muss dafür die Registry ändern. Er benutzt dafür am besten einen XML-Editor.

Damit auch die Clients die neue Registry benutzen können, wurde in der Schadensaufnahme ein weiterer Punkt aufgenommen: „Neue Registry installieren“. Wählt der Administrator diesen Punkt aus, wird er aufgefordert, die XML-Datei der Registry auszuwählen (siehe Abbildung 3.12). Über die Schaltfläche (1) kann er dazu einen Datei-Browser benutzen.



Verwende Aktion

Wählen Sie die neue Registry-Datei aus.

Dateiname ... **1**

Abbildung 3.12: Registry auswählen

3.3.3 Aktionen festlegen

In diesem Kapitel werden die grafischen Komponenten, aus denen die von den Benutzern durchzuführenden Aktionen erstellt werden, beschrieben. Die Screenshots zeigen die für die Aktion relevanten Bereiche.

In allen diesen Komponenten gibt es eine Check-Box „Verwende Aktion“. Ist sie markiert, so signalisiert dies, dass diese Aktion von den Benutzern durchzuführen ist und sie dementsprechend in das UpdateComponent aufgenommen wird. Ist die Check-Box nicht markiert, so

wird die gesamte Komponente grau hinterlegt und die zugehörige Aktion wird nicht generiert. Einige Komponenten besitzen weitere Check-Boxen (z.B. „Gruppe von CH-Schlüsseln löschen“: KeyUsage). Dies soll anzeigen, dass die zugehörigen Elemente (hier also die KeyUsage) optional sind. Ist die Check-Box selektiert, so werden die Elemente in die zu generierende Aktion aufgenommen, andernfalls nicht.

3.3.3.1 Einzelnen CH-Schlüssel löschen

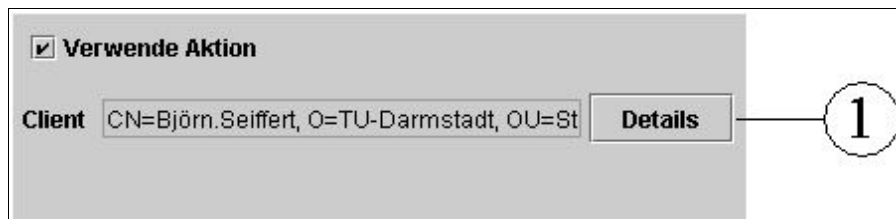


Abbildung 3.13: Einzelnen CH-Schlüssel löschen

Ein einzelner CH-Schlüssel, der gelöscht werden muss, wird durch das zugehörige Zertifikat bestimmt. Über die Schaltfläche (1) ist es möglich, sich das Zertifikat genauer anzuschauen.

3.3.3.2 Gruppe von CH-Schlüsseln löschen



Abbildung 3.14: Gruppe von CH-Schlüsseln löschen

Eine Gruppe von CH-Schlüsseln wird mit Hilfe des Algorithm-Identifiers der Schlüssel bestimmt. Für diese Aktion gibt es ein Textfeld, in das der Algorithmus eingetragen wird. Dies geschieht über die Schaltfläche (1). Wird diese Schaltfläche gedrückt, erscheint eine Auswahlliste von allen in der Registry gespeicherten Algorithmen, die für die Schlüsselgenerierung benutzt werden können.

Zusätzlich zu dem Schlüssel-Algorithmus kann die Gruppe durch Spezifizierung von einer Key-Usage und einer maximalen Schlüssellänge weiter eingeschränkt werden. Dazu wird die entsprechende Check-Box markiert und die einschränkenden Angaben spezifiziert.

3.3.3.3 Neuen CH-Schlüssel generieren

Verwende Aktion

Algorithm-Identifizier RSA (1.2.840.113549.1.1.1)

Key Usage

Bitte wählen Sie den Zweck aus:

DigitalSignature	NonRepudiation
KeyEncipherment	DataEncipherment
KeyAgreement	KeyCertSign
CriSign	EncipherOnly
DecipherOnly	

Schlüssel Länge 1024

Abbildung 3.15: Neuen CH-Schlüssel generieren

Der Algorithmus, für den die Benutzer neue Schlüssel generieren sollen, wird durch die zu löschenden Schlüssel bestimmt und ist hier nicht veränderbar. Zu einem zu installierenden Schlüssel gehört zwingend immer eine passende Schlüssellänge. Optional kann hier auch eine einschränkende Key-Usage mit angegeben werden.

3.3.3.4 Sicherheitsanker löschen

Verwende Aktion

Sicherheitsanker von CN=ump-updateservice-RSA, DC=ump, DC **1**

Abbildung 3.16: Sicherheitsanker löschen

Ein zu löschender Sicherheitsanker wird durch sein Zertifikat bestimmt. Über die Schaltfläche (1) kann man sich den Inhalt des Zertifikates genauer ansehen.

3.3.3.5 Sicherheitsanker austauschen

Verwende Aktion

Für den folgenden Sicherheitsanker muss ein neues Schlüsselpaar generiert werden. Daraus wird dann ein neuer Sicherheitsanker erzeugt, der installiert werden muss.

Sicherheitsanker von CN=ump-updateservice-RSA, DC=ump, DC **1**

Abbildung 3.17: Sicherheitsanker austauschen

Dieses Feld zeigt den Sicherheitsanker, für den ein neues Schlüsselpaar erzeugt werden muss. Über die Schaltfläche (1) kann man sich den Inhalt des Sicherheitsankers genauer ansehen.

3.3.3.6 Neue Zertifikate für CHs installieren

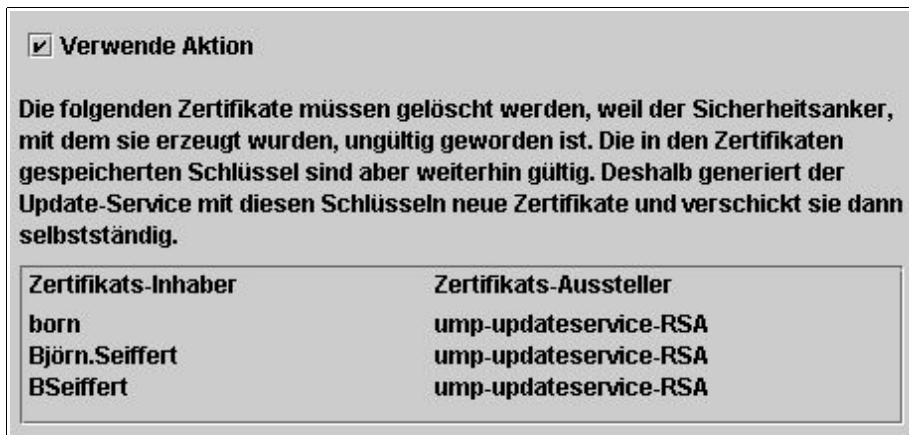


Abbildung 3.18: Neue Zertifikate für CHs installieren

Dieser Aktion ist vorausgegangen, dass der Sicherheitsanker einer CA ungültig geworden ist. Es mußte deshalb ein neuer Sicherheitsanker generiert werden, was alle Zertifikate, die mit dem alten Anker erzeugt wurden, ungültig macht. Die in den Zertifikaten gespeicherten CH-Schlüssel können allerdings weiter benutzt werden. Deshalb werden aus ihnen von der neuen CA neue Zertifikate erstellt und an die jeweiligen Benutzer versendet. Die Liste in Abbildung 3.18 zeigt alle betroffenen Zertifikate und ihre zugehörigen Zertifikats-Aussteller.

3.3.3.7 Algorithmen löschen

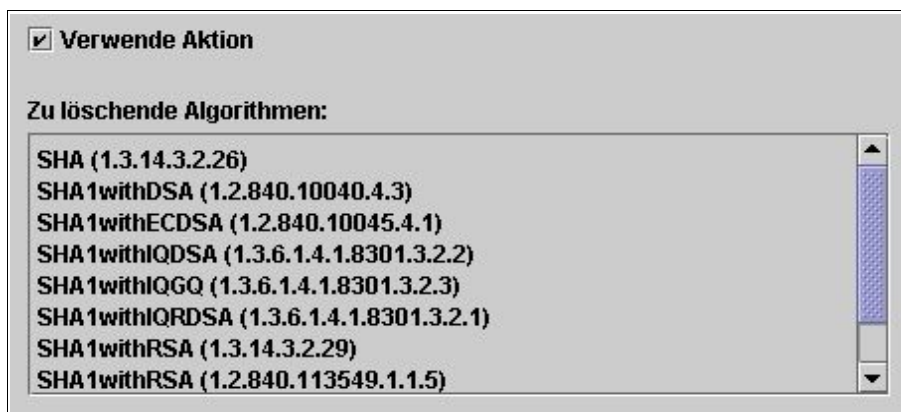


Abbildung 3.19: Algorithmen löschen

Hier können ein oder mehrere Algorithmen stehen, die gelöscht werden sollen. Meist sind diese Algorithmen von einander abhängig. In Abbildung 3.19 z.B. sind alle Algorithmen vom SHA-Algorithmus abhängig.

3.3.3.8 Neuen Sicherheitsanker installieren

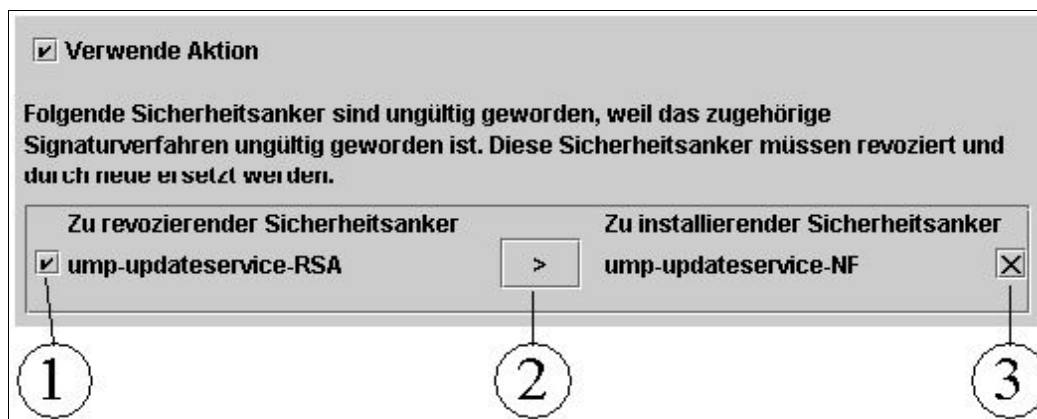


Abbildung 3.20: Neuen Sicherheitsanker installieren

Der Dialog aus Abbildung 3.20 beschreibt, welche Sicherheitsanker revoziert und durch welche neuen ersetzt werden sollen. Die zu revozierenden Anker werden durch die Algorithmen spezifiziert, die zusätzlich zu dieser Aktion gelöscht werden müssen. Für jeden zu revozierenden Sicherheitsanker muss ein anderer neuer Sicherheitsanker bestimmt werden. Dazu wird auf die Schaltfläche (2) gedrückt. Daraufhin erscheint eine Auswahl von allen bekannten CAs, von der die gewünschte ausgewählt wird. Die Schaltfläche (3) entfernt einen bereits ausgewählten zu installierenden Sicherheitsanker. Zu Testzwecken kann auch der Haken in der Check-Box vor einem zu revozierenden Sicherheitsanker entfernt werden (1). Dadurch wird für diese CA keine Aktion generiert.

3.3.3.9 Mehrere neue CH-Schlüssel generieren

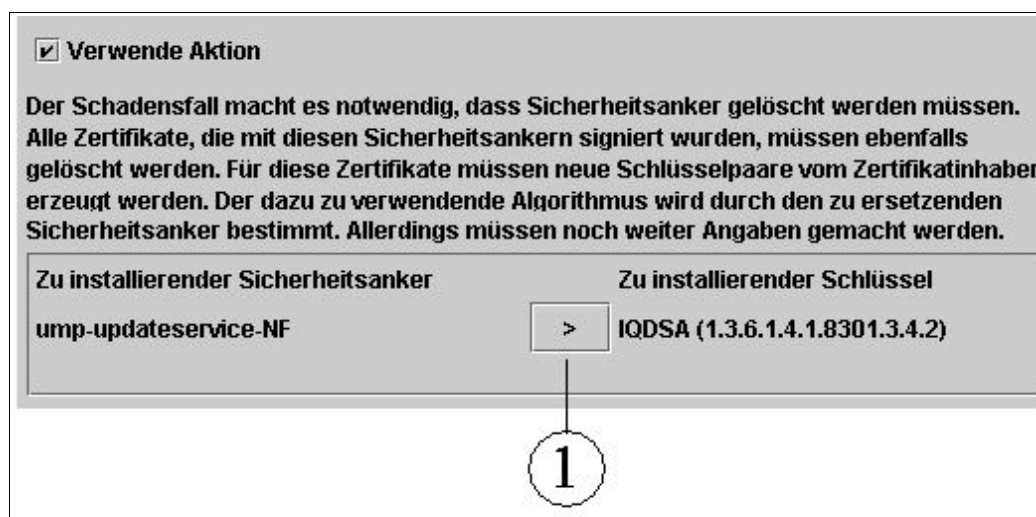


Abbildung 3.21: Mehrere neue CH-Schlüssel generieren

Wenn Sicherheitsanker gelöscht und durch neue ausgetauscht werden sollen, so werden die Zertifikate, die mit dem zu löschenden Sicherheitsanker ausgestellt wurden, ungültig und sind somit auch zu löschen. Deshalb müssen die zugehörigen Zertifikat-Inhaber für den neuen Sicherheitsanker neue, passende Schlüssel erzeugen. Der Algorithmus für diese Schlüssel wird durch die neue CA festgelegt. Allerdings muss zusätzlich noch mindestens die Schlüssel-

länge angegeben werden, optional ist auch eine Key-Usage möglich. Dazu klickt der Administrator auf die Schaltfläche (1). Dann erscheint ein neuer Dialog, in dem der zu installierende Schlüssel näher definiert wird. Dieser Dialog ist genauso aufgebaut, wie in Kapitel 3.3.3.3 beschrieben.

3.3.4 Neue Provider an Clients verteilen

Eine wichtige Funktionalität einer Fail-Safe-PKI ist der Austausch von kryptographischen Komponenten (siehe auch Kapitel 2.2). Im folgenden wird beschrieben, wie dies durch den Update-Service realisiert wird.

3.3.4.1 UpdateComponentCode-ID bestimmen

Als Teil der Schadensaufnahme ist die Aktion „Neuen Provider installieren“ vorgesehen. Hier wird die Provider-Klasse ausgewählt, die die Clients installieren sollen (siehe Abbildung 3.22).

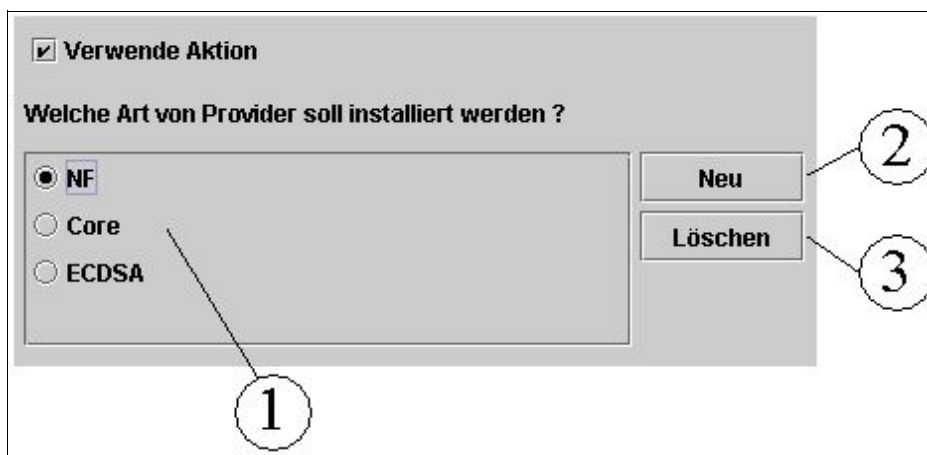


Abbildung 3.22: Neuen Provider auswählen

Bei Auswahl dieser Aktion erscheint eine Liste von den bekannten Klassen von Providern (1). Über die Schaltfläche (2) werden neue Klassen hinzugefügt und durch die Schaltfläche (3) kann die ausgewählte Klasse gelöscht werden.

Hinter jeder dieser Klassen von Providern steht eine eindeutige UpdateComponentCode-ID, die zusammen mit dem UpdateComponent an die Clients versendet wird. Der eigentliche Code des Providers wird allerdings auf diesem Weg noch nicht versendet.

3.3.4.2 Zuordnung Provider-Klasse + ClientIdentifier → Provider

Da die Clients der PKI nicht notwendig alle die gleiche Implementierung haben, muss für jede Klasse von Clients auch eine passende Provider-Implementierung existieren. Die Klasse eines Clients wird durch seinen ClientIdentifier bestimmt (siehe Kapitel 2.5.1.1).

Bevor der Update-Service nun ein UpdateComponent mit einer Install-Provider-Aktion erzeugt, wird überprüft, ob alle nötigen Implementierungen der Provider-Klasse existieren. Ist das nicht der Fall, so erscheint für jeden dieser ClientIdentifier ein weiterer Dialog, in dem die notwendigen Zuordnungen gemacht werden (siehe Abbildung 3.23). Es gibt zwei mögliche Stellen, an denen diese Abfrage gemacht wird:

1. Vom Update-Service direkt nach der Schadensaufnahme, wenn der ClientIdentifier eines Clients bekannt ist. Muss für den ClientIdentifier ein neuer Provider angegeben werden, so wird das direkt im Anschluss gemacht.
2. Wenn der ClientIdentifier eines Clients zum Zeitpunkt der Schadensaufnahme noch nicht dem Update-Service bekannt ist, wird der Client erst einmal aufgefordert, seinen Client-Identifier mitzuteilen. Das UpdateComponent mit den durchzuführenden Aktionen (also auch der Aktion, einen neuen Provider zu installieren) wird zwischengespeichert. Erhält das Update-Service-Servlet die Antwort mit dem ClientIdentifier, so kann es die Abfrage starten. Stellt sich nun heraus, dass für diesen ClientIdentifier und der Provider-Klasse noch keine existierende Provider-Implementierung bekannt ist, so wird der Administrator per E-Mail darüber informiert.

Für den folgenden Client sind weitere Eingaben erforderlich:

ClientIdentifier

ClientIdentifier - Klasse : Aktiv
Client ID : Nac-test
Client Version : 1.0
Client Serien Nummer :

Für diesen ClientIdentifier und dem UpdateComponentCode "NF" ist noch kein Provider angegeben worden.

Bitte wählen Sie einen Provider

FlexiNF-Provider
 FlexiCore-Provider
 FlexiEC-Provider

Neu
Löschen
Info

1, 2, 3, 4

Abbildung 3.23: Zuordnung Ucc-ID + ClientIdentifier → Provider

Entweder kann ein bereits existierender Provider ausgewählt werden (1) oder durch einen Klick auf die Schaltfläche (2) wird ein neuer angelegt (siehe Abbildung 3.24). Der Administrator kann dann wählen, von welcher Klasse ein neuer Provider sein soll. Momentan werden nur Java-Provider unterstützt, aber weitere Provider-Klassen können irgendwann nachgerüstet werden (siehe Kapitel 2.5.1.6). Bestehende Provider werden über die Schaltfläche (3) gelöscht.

Abbildung 3.24: Auswahl der Klasse eines neuen Providers

3.3.4.3 JCA Provider

Soll ein neuer JCA-Provider dem Update-Service bekannt gemacht werden, so werden dafür ein paar Angaben benötigt (siehe Abbildung 3.25):

- **Name des Providers (1).**
- **Jar-Datei, in der der JCA-Provider gespeichert ist (2).**
- **Klassenname des Providers (3).**
Nachdem die Jar-Datei des Providers ausgewählt wurde, werden alle darin enthaltenen Klassen in die Combo-Box eingetragen und der Administrator kann daraus auswählen.
- **Beschreibung des Providers (4).**
Diese Beschreibung ist optional. Sie wird dann angezeigt, wenn in Abbildung 3.23 die Schaltfläche (4) selektiert wird.

Abbildung 3.25: Angaben eines JCA-Provider

3.3.5 Erweiterte Eingaben

Wird zu Beginn der Schadensaufnahme der Reiter (1) selektiert, so gelangt man zu den erweiterten Eingaben (siehe Abbildung 3.26). In diesem Fenster werden alle weiteren Eingaben gemacht, die sich indirekt auf die Schadensaufnahme auswirken oder die

nachträglich gemacht werden müssen (2). In dem Textfeld (3) wird die Beschreibung der getroffenen Auswahl angezeigt.

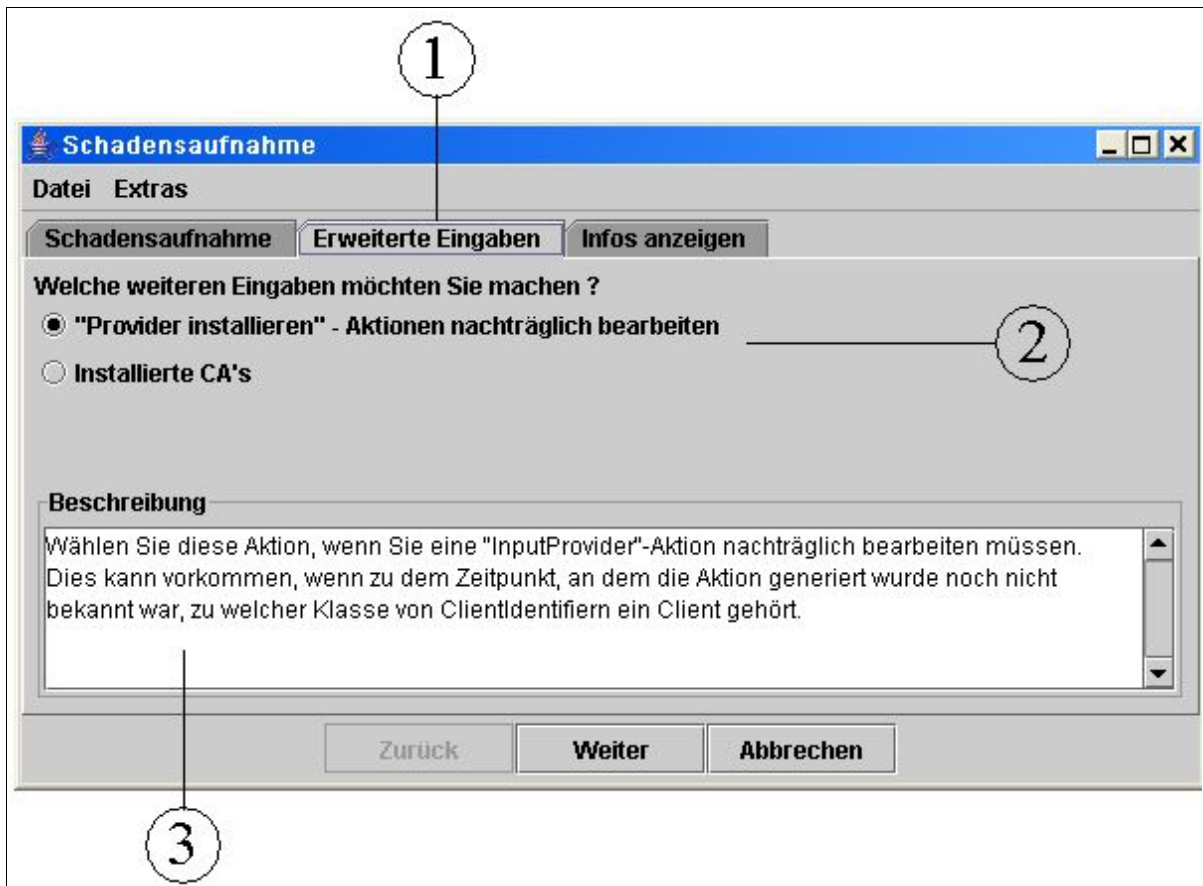


Abbildung 3.26: Erweiterte Eingaben

3.3.5.1 „Provider installieren“-Aktionen nachträglich bearbeiten

Bei der Schadensaufnahme definiert der Administrator, welche Aktionen die Clients auszuführen haben. Kennt der Update-Service den ClientIdentifier eines zu informierenden Clients, so werden diese Aktionen direkt in das UpdateComponent eingebaut. Ist der ClientIdentifier aber noch nicht bekannt, so wird er durch ein spezielles *GetClientIdentifier*-Objekt im UpdateComponent erfragt (vergl Kapitel 2.5.1.3). Die Aktionen werden dann zwischengespeichert und später, wenn der Client geantwortet hat, in einem neuen UpdateComponent verschickt.

Der ClientIdentifier ist immer dann für den Update-Service von Interesse, wenn der Client einen neuen Provider installieren soll. Durch den ClientIdentifier ist der Update-Service in der Lage, dem Client den korrekten Provider bereitzustellen. Wenn ein Client seinen ClientIdentifier zurückgeschickt hat, lädt der Update-Service die zwischengespeicherten Aktionen und überprüft, ob eine „Provider installieren“-Aktion darunter ist. Ist dies der Fall und ist für den erhaltenen ClientIdentifier und der in der Aktion gespeicherten UCC-ID (vergl Kapitel 2.5.1.7) bereits ein passender Provider vorhanden, so werden die Aktionen in ein UpdateComponent gepackt und nun versendet. Existiert hingegen noch kein geeigneter Provider in der Datenbank, so erhält der Administrator eine Nachricht per E-Mail, dass ein neuer Provider auszuwählen ist. Nach dem Erhalt dieser E-Mail ruft der Administrator im

Update-Service auf dem Reiter „Erweiterte Eingaben“ die Funktion „'Provider installieren'-Aktion nachträglich bearbeiten“ auf. Es erscheinen zwei Listen, in denen die Komponenten angegeben sind, die bearbeitet werden müssen (siehe Abbildung 3.27).

Hier ist eine Liste aller Ump-IDs, für die noch weitere Eingaben erforderlich sind.
Bitte wählen Sie erst eine Ump-ID und dann eine Client-ID.

Ump-ID	Client-ID
3	1

Abbildung 3.27: Angabe, welches UpdateComponent und welcher Client bearbeitet werden sollen

Auf der linken Seite des Fensters ist eine Liste aller UpdateComponents, die bearbeitet werden müssen (1). Nach der Auswahl einer Ump-ID erscheinen auf der rechten Fensterseite dann alle ClientIdentifier, die noch zu bearbeiten sind (2). Danach wird aus dem UpdateComponent mit der ausgewählten Ump-ID die Ucc-ID der „Neuen Provider installieren“-Aktion gelesen. Im nächsten Fenster kann nun bestimmt werden, welcher Provider für den ausgewählten ClientIdentifier und die Provider-Klasse benutzt werden soll. Die nötigen Eingaben sind bereits in Kapitel 3.3.4.2 beschrieben worden.

3.3.5.2 Installierte CAs

Durch auftretende Schäden kann es vorkommen, dass einzelne CAs ausfallen. Dann muss für jede dieser CAs eine neue bereitgestellt werden. Durch Auswahl vom Reiter „Erweiterte Eingaben“ und dann „Installierte CAs“ bekommt man einen Überblick über sämtliche dem Update-Service bekannte CAs (siehe Abbildung 3.28).

1

2

Installierte CAs

Deaktivieren	CN=ump-updateservice-NF, DC=ump, DC=cdc, DC=tu-darmstadt, C=DE, E...	Neu
Aktivieren	CN=ump-updateservice-ECDSA, DC=ump, DC=cdc, DC=tu-darmstadt, C=I...	
Deaktivieren	CN=ump-updateservice-RSA, DC=ump, DC=cdc, DC=tu-darmstadt, C=DE,	

Ende

Abbildung 3.28: Vorhandene CAs

Über die Schaltflächen (1) ist es möglich, einzelne CAs zu aktivieren bzw. zu deaktivieren. Soll eine neue CA installiert werden, so geschieht dies über die Schaltfläche (2). Daraufhin erscheint der Dialog, der in Abbildung 3.29 zu sehen ist. Nun wird die Jar-Datei angegeben, die die Implementierung der neuen CA beinhaltet und danach wird aus der Combo-Box die CA-Klasse angegeben.

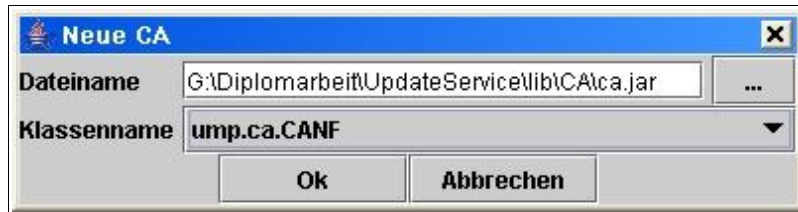


Abbildung 3.29: Neue CA installieren

Damit die neue CA vom Update-Service benutzt werden kann, ist es notwendig, die ausgewählte Jar-Datei in den Java-Classpath aufzunehmen. Anschließend muss der Update-Service neu gestartet werden.

3.3.6 Infos anzeigen

Der Update-Service generiert für die Benachrichtigung der Clients eine Reihe von Update-Components auf die er dann wiederum Antworten in Form von UpdateComponentResponses erhält. Je nach Schadensfall kann dieser Austausch von Objekten bis zu dreimal ablaufen. Damit der Administrator sich eine Übersicht über diese Objekte verschaffen kann, gibt es einen weiteren Reiter „Infos anzeigen“ (siehe Abbildung 3.30). Hier sind diverse Funktionen aufgelistet, die die unterschiedlichen Objekte bzw. die durchgeführten Abläufe anzeigen. Diese Funktionen werden in den folgenden Kapiteln näher beschrieben.



Abbildung 3.30: Infos anzeigen

3.3.6.1 UpdateComponent-Status

Hier hat man die Möglichkeit, den Verlauf der UpdateComponents zu verfolgen. In eine Tabelle werden die Ump-IDs der UpdateComponents, die einzelnen Clients, an die die UpdateComponents verschickt wurden und der aktuelle Status angezeigt (siehe Abbildung 3.31). Folgende Ergebnisse sind möglich:

- **Noch nicht versendet**
Das UpdateComponent wurde noch nicht an den Client gesendet. Dies kann z.B. dann auftreten, wenn zum Zeitpunkt, an dem es erstellt wurde, der zugehörige ClientIdentifier noch nicht bekannt war.
- **Noch nicht beantwortet**
Das UpdateComponent wurde bereits an den Client verschickt, aber der hat noch nicht darauf geantwortet.
- **Antwort-Status: Ok**
Eine Antwort des Clients ist eingegangen und alle Aktionen des UpdateComponents wurden vom Client erfolgreich ausgeführt.
- **Antwort-Status: Fehler**
Eine Antwort des Clients ist eingegangen und mindestens eine Aktion des UpdateComponents wurde vom Client nicht erfolgreich ausgeführt.

Bei der Selektion der Schaltfläche (1) in Abbildung 3.31, werden die erhaltenen Antworten eines UpdateComponents angezeigt.

Ergebnisse des Update-Service		
Ump-ID	Client	Ergebnis
1	CN=born, O=TU-Darmstadt, OU...	Noch nicht beantwortet
2	CN=born, O=TU-Darmstadt, OU...	Noch nicht versendet
1	CN=Björn.Seiffert, O=TU-Darms...	Antwort-Status: Ok
3	CN=Björn.Seiffert, O=TU-Darms...	Noch nicht versendet
1	CN=BSeiffert, O=TU-Darmstadt, ...	Noch nicht beantwortet
1	CN=BSeiffert, O=TU-Darmstadt, ...	Noch nicht versendet

Details

1

Abbildung 3.31: UpdateComponent-Status

3.3.6.2 UpdateComponent, UpdateComponentResponse, UpdateComponentCode, Registry anzeigen

Mit diesen Aktionen kann man sich die jeweiligen generierten Komponenten anzeigen lassen. Das ist besonders zu Testzwecken sehr nützlich um zu überprüfen, ob die Komponenten korrekt zusammgebaut wurden.

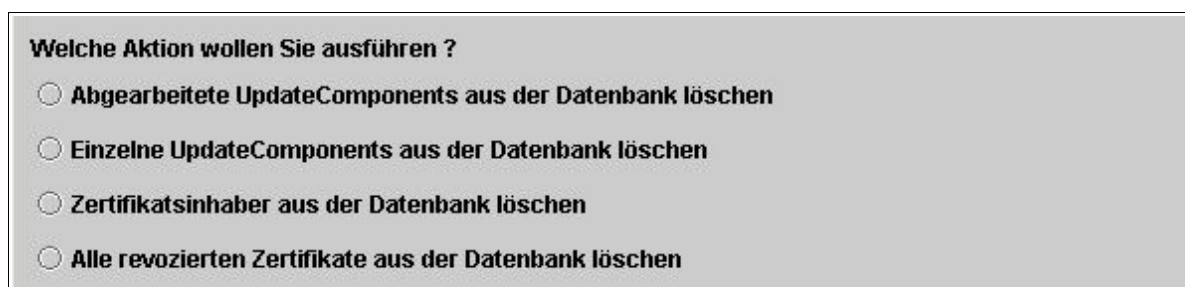
- **UpdateComponent und UpdateComponentResponse**
In einer Combo-Box werden alle Ump-IDs der UpdateComponents bzw. UpdateComponentResponses, die in der Datenbank gespeichert sind zusammen mit dem

Erstellungsdatum angezeigt. Nach der Auswahl einer Ump-ID wird das Textfeld mit dem Inhalt der Komponente gefüllt.

- ***UpdateComponentCode***
Hier muss zuerst die Ucc-ID und dann ein ClientIdentifier aus einer Combo-Box ausgewählt werden. Danach wird das Textfeld mit dem Inhalt des zugehörigen UpdateComponentCodes ausgefüllt.
- ***Registry***
Hiermit kann der Inhalt der aktuell verwendeten Registry angezeigt werden.

3.3.6.3 Datenbankeinträge löschen

Im Laufe der Zeit kann die Datenbank immer größere Ausmaße annehmen, weil jedes UpdateComponent, UpdateComponentResponse und Zertifikate in ihr gespeichert werden. Deshalb kann es nützlich sein, ab und zu Einträge, die nicht mehr benötigt werden, zu löschen. Abbildung 3.32 zeigt, welche Einträge gelöscht werden können.



Welche Aktion wollen Sie ausführen ?

- Abgearbeitete UpdateComponents aus der Datenbank löschen
- Einzelne UpdateComponents aus der Datenbank löschen
- Zertifikatsinhaber aus der Datenbank löschen
- Alle revozierten Zertifikate aus der Datenbank löschen

Abbildung 3.32: Datenbankeinträge löschen

- ***Abgearbeitete UpdateComponents aus der Datenbank löschen***
Es werden alle UpdateComponents, die von allen Clients beantwortet wurden angezeigt. Diejenigen Einträge, die nicht mehr benötigt werden, können markiert und anschließend gelöscht werden. Zusätzlich werden auch die zugehörigen UpdateComponentResponses gelöscht.
- ***Einzelne UpdateComponents aus der Datenbank löschen***
Sämtliche UpdateComponents, die in der Datenbank gespeichert sind, werden aufgelistet und können einzeln gelöscht werden.
- ***Zertifikatinhaber aus der Datenbank löschen***
Einzelne in der Datenbank gespeicherte Zertifikatinhaber können gelöscht werden.
- ***Alle revozierten Zertifikate aus der Datenbank löschen***
Sämtliche, durch eingetretene Schadensfälle ungültig gewordenen Zertifikate werden aus der Datenbank gelöscht.

Kapitel 4 Implementierungen

Dieses Kapitel beschreibt die Implementierung des Update-Service. Da das zu verwendende Update-Management-Protocol in Java geschrieben wurde, ist Java auch als Programmiersprache für den Update-Service ausgewählt worden. Java bietet außerdem noch weitere Vorteile gegenüber anderen Programmiersprachen:

- Java ist plattformunabhängig.
- In Java ist bereits die Internetanbindung integriert.
- Über *Java Database Connectivity* (JDBC) existiert eine standardisierte Anbindung an relationale Datenbanken.
- Java besitzt mit der *Java Cryptographic Architecture* (JCA) und der *Java Cryptography Extension* (JCE) Konzepte für den Einsatz von kryptographischen Algorithmen und X509-Zertifikaten. Damit ist auch die Verwendung des CDC-Providers möglich, der an der TU-Darmstadt entwickelt wird [FLEXI].

4.1 Verwendete Werkzeuge und Software

Im Rahmen dieser Diplomarbeit wurden eine Reihe von Programmen und Software benutzt:

- ***Java Development Kit 1.4.2_04***
Basisklassen und Applikationen zum Erstellen und Testen von Java-Programmen.
- ***JavaMail 1.3.1***
E-Mail-Erweiterung der Java-Programmiersprache von Sun.
- ***Update-Management-Protocol V2***
Java-Implementierung des UMPs.
- ***MySQL 4.0.18***
Frei erhältliche relationale Datenbank.
- ***OpenLDAP 2.1.29-1***
LDAP Implementierung für die Speicherung von Zertifikaten.
- ***FlexiProvider 1.1.5p5***
Implementierung von kryptographischen Standardalgorithmen sowie Elliptic-Curves- und Number-Field-Algorithmen.
- ***Eclipse 2.1***
Freie Entwicklungsumgebung für Java.
- ***OpenOffice 1.1.2***
Textverarbeitungssystem für die Erstellung dieser Ausarbeitung.

4.2 Verwendete Java-Packages

Der Update-Service ist in folgende Packages unterteilt:

- ***ump.update.service***
Basis-Package des Update-Service.
- ***ump.update.service.certificate_generation***
Klassen zum Generieren von Zertifikaten (Kapitel 4.3.3).
- ***ump.update.service.connection.deliver***
Klassen zum Versenden von UpdateComponents.
- ***ump.update.service.connection.receive***
Klassen zum Empfangen von UpdateComponentResponses (Update-Service-Servlet, Kapitel 4.3.2).
- ***ump.update.service.crypto***
Klassen zum Verwalten von Zertifikaten und CAs.
- ***ump.update.service.database***
Klassen für die Verwaltung der Datenbank (Kapitel 4.5).
- ***ump.update.service.gui***
Klassen für die Grafische Oberfläche (Kapitel 3.3.2).
- ***ump.update.service.helper***
Diverse Hilfsklassen.
- ***ump.update.service.icons***
Verwendete Icons.
- ***ump.update.service.translation***
Übersetzungsdateien der Texte des Update-Service.

4.3 Update-Service

Der Update-Service ist mit dem Ziel geschrieben worden, dass nachträgliche Erweiterungen leicht durchzuführen sind. Dazu wurden eine Reihe von Interfaces und abstrakten Klassen definiert, die benutzt werden müssen, damit Erweiterungen in das bestehende Programm integriert werden können.

4.3.1 Schadensaufnahme

In diesem Kapitel wird beschrieben, wie die Schadensaufnahme erweitert, bzw. angepasst werden kann.

4.3.1.1 Schadensfälle

In der Schadensaufnahme sind zwar schon eine Reihe von Schadensfällen implementiert worden, allerdings ist es denkbar, dass in Zukunft weitere Fälle berücksichtigt werden müssen.

Alle Schadensfälle müssen von der Klasse *ump.update.service.gui.panel.InputDamagePanel* abgeleitet werden. In ihr sind Methoden definiert, die den Schadensfall nach außen präsentieren und die überschrieben werden müssen. Außerdem werden in *InputDamagePanel* die UpdateComponents zusammengebaut. Folgende Methoden müssen überschrieben werden:

- ***getDamageName()***
Liefert den Namen des Schadensfalles. Dieser Name erscheint als Auswahlmöglichkeit im Reiter *Schadensaufnahme* (siehe Kapitel 3.3.2).
- ***getDamageDescription()***
Liefert die Beschreibung des Schadensfalles.
- ***checkRequirements()***
Diese Methode wird immer als Letztes aufgerufen, bevor die UpdateComponents zusammengebaut werden. Hier können noch letzte Abfragen des Schadensfalles gemacht werden, wenn das nötig sein sollte.

Neue Schadensfälle können bereits existierende Komponenten aus *InputDamagePanel* benutzen, um Eingaben vom Administrator anzufordern:

- ***UpdateComponentInputPanel***
JPanel mit Textfeldern für die Adresse des Update-Service-Servlets und der Kurzbeschreibung des Schadensfalles.
- ***DeleteAlgorithmActionPanel***
JPanel mit einem Textfeld, in dem Algorithmen mit Namen und OIDs angezeigt werden.
- ***DeleteMultipleAlgorithmActionPanel***
JPanel mit einer JScrollPane, das eine Liste von Algorithmen mit Namen und OIDs formatiert anzeigen kann.
- ***DeleteSingleChKeyActionPanel***
JPanel mit einem Textfeld, in dem der Name eines Zertifikat-Inhabers stehen kann. Über eine Schaltfläche kann das Zertifikat angezeigt werden.
- ***DeleteMultipleChKeyActionPanel***
JPanel, um eine Gruppe von Zertifikaten zu spezifizieren. Das erfolgt über die Eingabe von einem Algorithm-Identifizier und optional einer Key-Usage und einer Schlüssellänge.
- ***DeleteSingleTrustAnchorActionPanel***
JPanel mit einem Textfeld, in dem der Name eines einzelnen Sicherheitsankers steht. Über eine Schaltfläche kann der Sicherheitsanker angezeigt werden.
- ***ReplaceTrustAnchorKeyActionPanel***
JPanel das anzeigt, dass für einen Sicherheitsanker ein neues Schlüsselpaar generiert wird.
- ***ReplaceMultipleTrustAnchorActionPanel***
JPanel, in dem mehrere Sicherheitsanker durch andere ersetzt werden können.

- ***InstallMultipleChCertificateActionPanel***
JPanel, in dem alle Zertifikate aufgelistet werden, deren Inhaber neue Schlüsselpaare generieren müssen.
- ***InstallProviderActionPanel***
JPanel auf dem aufgelistet wird, welche UpdateComponentCodes bereits in der Datenbank gespeichert sind. Der Administrator kann daraus auswählen.
- ***InstallChKeyActionPanel***
JPanel, in dem näher definiert werden kann, was für ein Schlüsselpaar von einem Benutzer generiert werden soll.
- ***InstallRegistryActionPanel***
JPanel, in dem eine Registry-Datei ausgewählt werden kann.
- ***InstallMultipleChKeysForTrustAnchorsPanel***
JPanel, in dem angegeben werden kann, was für Schlüsselpaare die Benutzer für eine Reihe von Sicherheitsankern generieren sollen.

Damit eine Klasse, die einen neuen Schadensfall beschreibt, der Schadensaufnahme bekannt sein kann, muss sie noch in der Properties-Datei des Update-Service eingetragen werden. Die Datei befindet sich im *src*-Verzeichnis unter
ump/updateservice/updateservice.properties

In dieser Datei sind alle Schadensfälle eingetragen unter
MeasureClass.xx = <Vollständiger Klassenname>
 „xx“ steht hier für eine Zahl, die die Stelle angibt, an der der Schadensfall angezeigt wird.

4.3.1.2 Erweiterte Eingaben

Für Eingaben, die der Administrator machen kann, die nicht direkt einen Schadensfall betreffen, steht der Reiter „Erweiterte Eingaben“ im Hauptfenster des Update-Services (vergl. Kapitel 3.3.5). Wenn irgendwann weitere Eingaben nötig sein sollten, so erstellt man sich eine Klasse, die von der Klasse *NameAndDescriptionPanel* abgeleitet wird. Diese Klasse stellt das Grundgerüst für erweiterte Eingaben dar. Es müssen folgende Methoden überschrieben werden, um die neue Klasse in den Update-Service einzubinden:

- ***getDamageName()***
Liefert den Namen für die Eingabe.
- ***getDamageDescription()***
Liefert die Beschreibung für die Eingabe.

Wie auch bei neuen Schadensfällen müssen die Klassen für die erweiterte Eingabe in der Properties-Datei des Update-Service eingetragen werden. Die Einträge haben die folgende Form:

MoreInputClass.xx = <Vollständiger Klassenname>
 „xx“ steht für eine Zahl, die die Stelle angibt, an der der Name der Klasse angezeigt wird.

4.3.1.3 Infos anzeigen

Alle Klassen, die Informationen über Daten anzeigen sollen, die der Update-Service erzeugt hat, müssen von der Klasse *NameAndDescriptionPanel* abgeleitet werden. Auch hier müssen Methoden überschrieben werden:

- ***getDamageName()***
Liefert den Namen für die Eingabe.
- ***getDamageDescription()***
Liefert die Beschreibung für die Eingabe.

In den Properties stehen für alle Klassen, die Informationen anzeigen, Einträge der Form
DisplayInfoClass.xx = <Vollständiger Klassenname>
 „xx“ steht für eine Zahl, die die Stelle angibt, an der der Name der Klasse angezeigt wird.

4.3.1.4 Provider-Implementierungen

Zum Zeitpunkt, an dem diese Diplomarbeit geschrieben wurde, gibt es nur eine Klasse von Providern, die vom Update-Service benutzt werden: JCA-Provider. Falls irgendwann weitere Provider-Klassen benutzt werden sollen, so müssen diese erst dem Update-Service bekannt gemacht werden.

Jede existierende Klasse von Providern benötigt eine Java-Klasse, in der alle notwendigen Eingaben getätigt werden können. Diese Java-Klasse muss von der Klasse *ump.updateService.gui.interfaces.ProviderChoicesInterface* abgeleitet werden. Dieses Interface definiert folgende Methoden:

- ***getProviderChoiceName()***
Liefert den Namen des Providers.
- ***getProviderDescription()***
Liefert eine Beschreibung des Providers.
- ***getProvider()***
Liefert den kompletten, vom Administrator eingegebenen Provider in ASN.1-Notation.
- ***getInputPanel()***
Liefert das JPanel, in dem der Administrator den Provider definieren kann.

Alle Provider-Klassen werden in den Properties gespeichert unter
ProviderTypePanel.x = <Vollständiger Klassenname> .

4.3.2 Update-Service-Servlet

Alle Klassen, die zum Update-Service-Servlet gehören, sind im Package *ump.updateService.connection.receive* zusammengefasst:

- **Klasse *UpdateServiceHttpServer***
Implementiert einen rudimentären Http-Server, der auf dem Http-Server von der Firma Sun aufbaut. Das Update-Service-Servlet wird dem Http-Server über die Datei *UpdateServiceServlet.properties* bekannt gegeben.
- **Klasse *UpdateServiceServlet***
Basis-Klasse des Update-Service-Servlets. Sie arbeitet die Http-POST Methode ab, die vom Http-Server übergeben wurde und leitet die empfangenen UpdateComponent-Responses an die Klasse *ResponseService* zur Verarbeitung weiter.

- **Klasse *ResponseService***

Diese Klasse arbeitet empfangene *UpdateComponentResponses* ab, speichert die darin enthaltenen Antworten in die Datenbank und leitet alle weiteren nötigen Maßnahmen ein.

4.3.3 Generierung von neuen Zertifikaten

Neu zu erzeugende Zertifikate können nur von CAs ausgestellt werden. Dies kann je nach Implementierung der CAs längere Zeit in Anspruch nehmen. Deshalb wird das Beantragen neuer Zertifikate im Update-Service in einen separaten Thread ausgelagert, der so unabhängig die notwendigen Maßnahmen durchführt. Alle Klassen, die dafür benötigt werden, befinden sich in dem Package *ump.update.service.certificate_generation*.

4.4 Beispiel-CAs

Der Update-Service benötigt für seine Arbeit mindestens zwei CAs (vergl. Kapitel 2.2). Um möglichst unabhängig von etwaigen CA-Implementierungen zu sein, gibt es nur eine Klasse, die direkt mit den CAs kommuniziert (siehe Abbildung 4.1). Diese Klasse (*ump.ca.CaConnector*) implementiert das Interface *ump.ca.CaConnectorInterface*. Der Update-Service verwendet nur Methoden, die in diesem Interface definiert sind. Die Klasse kann daher jederzeit gegen eine andere ausgetauscht werden. Die neue Klasse muss lediglich ebenfalls das Interface *CaConnectorInterface* implementieren und in der Properties-Datei des Update-Service eingetragen werden unter

CA.Class = <Vollständiger Klassenname>

Folgende Methoden werden durch *CaConnectorInterface* definiert:

- ***disable(..)***
Deaktiviert eine CA.
- ***enable(..)***
Aktiviert eine CA.
- ***generateCertificate(..)***
Die angegebene CA soll ein neues Zertifikat für einen Benutzer generieren.
- ***generatedCertificateIsAvailable(..)***
Überprüft, ob ein zu erzeugendes Zertifikat bereits fertig erzeugt wurde.
- ***generateNewTrustAnchor(..)***
Die angegebene CA soll einen neuen Sicherheitsanker erzeugen.
- ***getAllIssuers()***
Alle Namen der vorhandenen CAs werden zurückgegeben.
- ***getGeneratedCertificate(..)***
Liefert ein neu erzeugtes Zertifikat.
- ***getIssuerAlgorithmIdentifier(..)***
Liefert den Algorithm-Identifizier, der von einer CA unterstützt wird.

- ***getOldTrustAnchor(..)***
Liefert den zuletzt gültigen Sicherheitsanker einer CA.
- ***getTrustAnchor(..)***
Liefert den aktuell gültigen Sicherheitsanker einer CA.
- ***isDisabled(..)***
Überprüft, ob eine CA deaktiviert ist.
- ***revokeCertificate(..)***
Revoziert ein Zertifikat.
- ***signMessage(..)***
Signiert eine Nachricht durch den aktuellen Sicherheitsanker einer CA.
- ***signMessageWithOldTrustAnchor(..)***
Signiert eine Nachricht durch den zuletzt gültigen Sicherheitsanker einer CA.

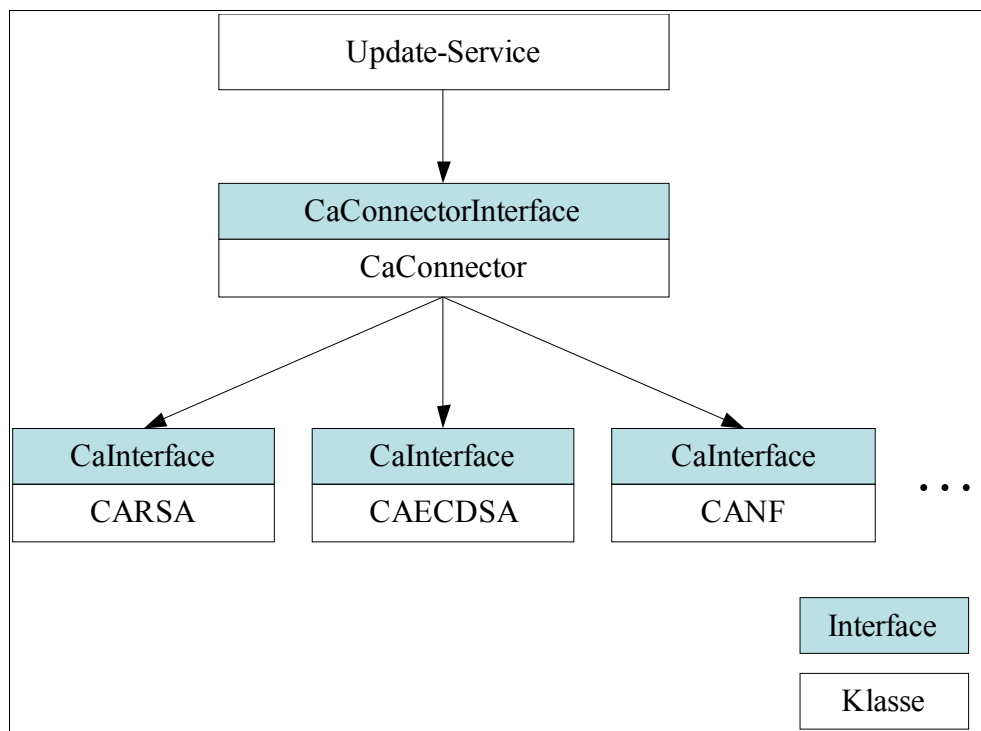


Abbildung 4.1: Zusammenhang zwischen Update-Service und CAs

CAs implementieren das Interface *ump.ca.CaInterface*. Es definiert alle benötigten Methoden, um auf die CAs zuzugreifen, z.B. Methoden zum Signieren von Daten oder zum Revozieren von Zertifikaten.

4.5 Datenbank

Der Update-Service benötigt für den Betrieb eine Datenbank, die alle nötigen Daten speichert. Abbildung 4.2 zeigt ihren tabellarischen Aufbau. Es gibt mehrere Bereiche, in denen Daten gespeichert werden:

- ***certificate***
Alle Zertifikate, die von einem Schadensfall betroffen sind, werden hier gespeichert. Um schnelleren Zugriff auf die in den Zertifikaten gespeicherten Daten zu ermöglichen, werden hier auch zusätzlich der Name des Zertifikat-Ausstellers und die Seriennummer gespeichert.
- ***certificate_holder***
Alle nötigen Daten über den Zertifikat-Inhaber werden hier abgelegt: Name des Inhabers, Verweis auf den Client-Identifizierer und Verweis auf die E-Mail-Adresse.
- ***client_identifier***
Alle bekannten Client-Identifizierer werden hier gespeichert.
- ***email***
Speicherplatz für die bekannten E-Mail-Adressen der Clients.
- ***update_component***
Unter ihrer Ump-ID wird das UpdateComponent zusammen mit dem Sende-Datum und dem Namen des zugehörigen Schadensfalles abgelegt.
- ***update_component_response***
Die UpdateComponentResponses werden zusammen mit ihrer Ump-ID und dem zugehörigen Absender gespeichert. Des Weiteren werden hier auch die vom Client generierten Antworten auf durchgeführte Aktionen, den Status, die IP-Adresse des Clients und das Datum, an dem das UpdateComponentResponse empfangen wurde, gespeichert.
- ***uc_client_status***
In dieser Tabelle werden Informationen abgelegt, die anzeigen, welche Teile eines UpdateComponents zu welchem Client gehören. Außerdem wird hier gespeichert, ob das UpdateComponent bereits gesendet oder beantwortet ist und ob bei Beantwortung ein weiteres UpdateComponent verschickt werden muss.
- ***update_components_wait_for_manual_input***
Infos über UpdateComponents, für die noch weitere Eingaben gemacht werden müssen, werden hier gespeichert (siehe auch Kapitel 3.3.5).
- ***update_component_code***
Der zu einer Ucc-ID und einem Client-ID gehörige UpdateComponentCode wird in dieser Tabelle gespeichert. Außerdem wird der Name und eine Beschreibung des Codes hier abgelegt.
- ***provider***
Hier werden alle nötigen Informationen über einen Provider gespeichert.
- ***updateservice_properties***
Diese Tabelle dient dazu, vom Administrator gemachte Einstellungen zu speichern.

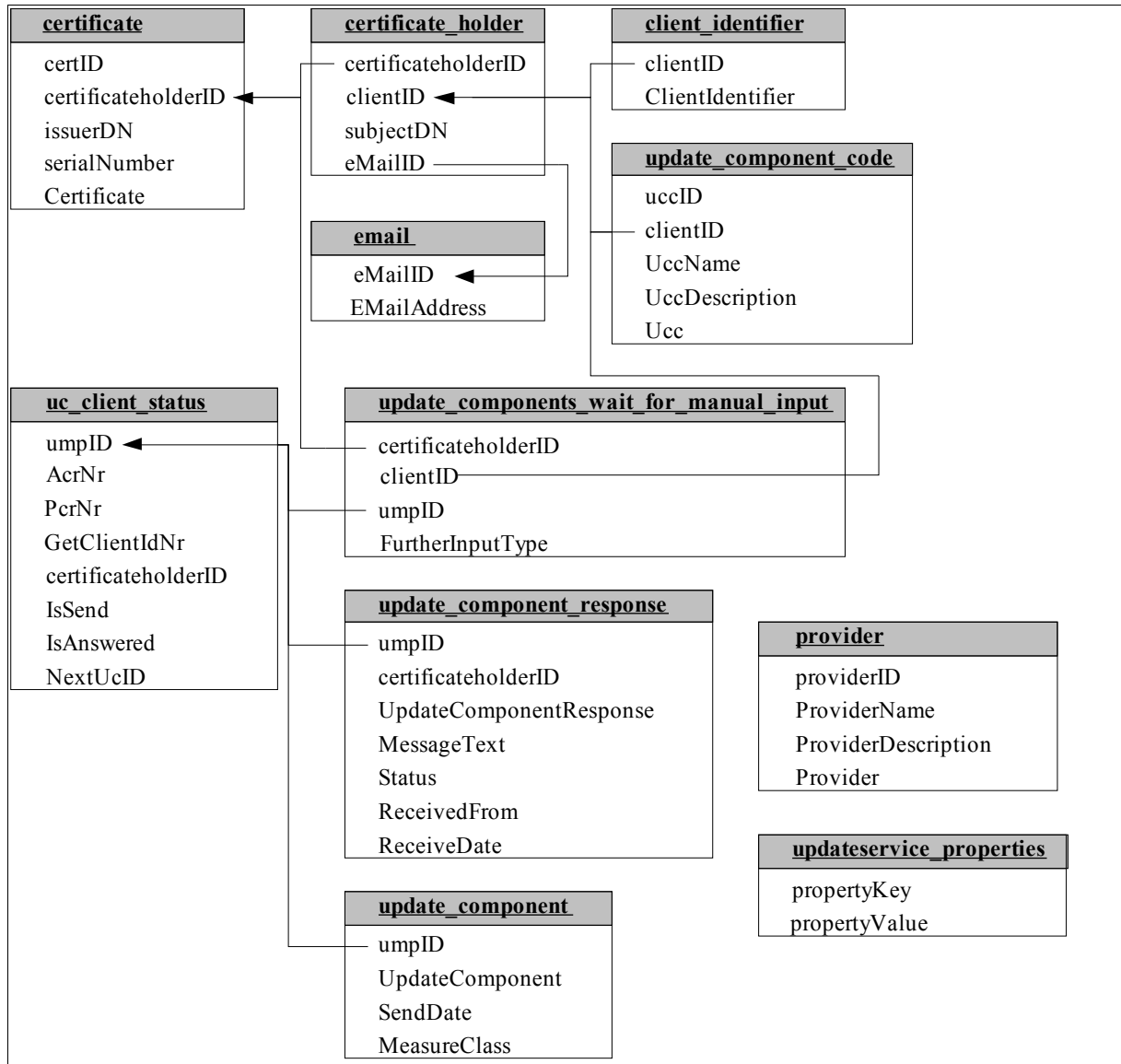


Abbildung 4.2: Tabellen der Datenbank

Anhang A Installation

In diesem Anhang wird die Installation des Update-Services beschrieben. Es werden die notwendigen Software-Komponenten vorgestellt. Jede benötigte Software ist auf der Update-Service CD enthalten.

A.1 Verzeichnisstruktur der Update-Service CD

	/Software	Hauptverzeichnis der benötigten Software-Komponenten
	/Software/Java	Hauptverzeichnis der Java-Komponenten
	/Software/Java/ANT	Apache ANT
	/Software/Java/Eclipse	IDE Eclipse für Windows
	/Software/Java/JDK	Java Development Kit Version 1.4.2_04
	/Software/LDAP	LDAP-Server für Windows und Linux LDAP-Browser zum Anzeigen von LDAP-Verzeichnissen
	/UpdateService	Hauptverzeichnis des Update-Services
	/UpdateService/bin/MySQL	Batch-Dateien zur Verwaltung der MySQL-Datenbank
	/UpdateService/config	Konfigurationsverzeichnis des Update-Service
	/UpdateService/config/CA	Installierte CAs
	/UpdateService/config/LDAP	Konfigurationsdateien zum Lesen des LDAP-Verzeichnisses
	/UpdateService/config/registry	Standard Registry
	/UpdateService/doc	Dokumentation
	/UpdateService/lib	Library-Verzeichnis
	/UpdateService/src	Quelltext-Verzeichnis

Abbildung 4.3: Verzeichnisstruktur der Update-Service CD

A.2 Benötigte Komponenten

Für eine erfolgreiche Installation des Update-Services werden eine Reihe von Programmen benötigt, die im Folgenden aufgelistet sind. Alle Programme sind als Freeware erhältlich. Auf der Update-Service CD (s. Kapitel A.1) befinden sich die momentan aktuellsten Versionen (Stand: Mai 2004). Bezugsquellen für neuere Versionen sind in Kapitel A.2.6 aufgelistet.

Auf die notwendigen Installationsschritte wird hier nicht eingegangen. Bei Problemen sollte man die Installationshinweise der jeweiligen Hersteller bzw. deren Internetseiten konsultieren.

A.2.1 Java

Für den Update-Service wird eine existierende Installation eines Java Development Kits (JDK) vorausgesetzt. Das Programm wurde mit dem JDK 1.4.2 geschrieben, also empfiehlt es sich, mindestens diese Version zu verwenden [JAVA]. Der Update-Service ist evtl. auch mit älteren Versionen des JDKs lauffähig, dies wurde aber nicht getestet. Für das JDK mit der Version 1.4.2_04 sind auf der Update-Service CD Installationsdateien enthalten. Für die Installation werden Administrator-Rechte notwendig.

Windows: /Software/Java/JDK/j2sdk-1_4_2_04-windows-i586-p.exe
Linux: /Software/Java/JDK/j2sdk-1_4_2_04-linux-i586.bin

A.2.2 Ant

Zum Übersetzen der Quelltexte wird das Tool „Apache ANT“ benötigt [ANT]. Diese Software bietet die Möglichkeit, Java-Quelltexte plattformübergreifend zu übersetzen. Eine Implementierung ist in

/Software/Java/ANT/apache-ant-1.6.1-bin.zip

zu finden.

ANT benötigt zum Compilieren von Java-Quelldateien eine spezielle XML-Datei (*build.xml*), die die notwendigen Installationsschritte definiert. Diese Datei ist im Update-Service-Hauptverzeichnis enthalten.

A.2.3 MySQL

Zum Ablegen programmspezifischer Daten wird eine Datenbank benötigt. Grundsätzlich lässt sich jede Datenbank verwenden, die über eine Java-JDBC-Schnittstelle verfügt. Im Rahmen dieser Diplomarbeit wird die frei erhältliche Datenbank MySQL benutzt [MYSQL]. Für die Installation werden Administrator-Rechte benötigt.

A.2.4 Certification-Authorities (CA)

Der Update-Service arbeitet mit mehreren CAs zusammen. Diese CAs verwalten die Benutzer-Zertifikate und ein LDAP-Verzeichnis. Diese CAs werden eigentlich vorausgesetzt.

Zu Testzwecken können aber auch drei CAs verwendet werden, die mit dem Update-Service mitgeliefert werden.

A.2.5 LDAP

Der Update-Service benötigt ein LDAP-Verzeichnis, in dem alle Benutzer-Zertifikate abgelegt sind. Normalerweise besitzt eine existierende CA ein solches Verzeichnis. Sollte aber keine CA zur Verfügung stehen, so kann zu Testzwecken die LDAP-Implementierung von der Update-Service CD verwendet werden [OPENLDAP].

Windows: /Software/LDAP/openldap-2_1_29-1-win32.exe
Linux: /Software/LDAP/openldap-stable-20040421.tgz

A.2.6 Bezugsquellen

- *JDK 1.4.2* <http://java.sun.com/j2se>
- *ANT* <http://ant.apache.org/index.html>
- *MySQL* <http://dev.mysql.com/>
- *LDAP* <http://www.openldap.org/>

A.3 Notwendige Anpassungen

Nach der erfolgreichen Grundinstallation der benötigten Software aus Kapitel A.2 müssen noch ein paar Anpassungen für den reibungsfreien Ablauf durchgeführt werden. Diese Anpassungen betreffen hauptsächlich Scripte, die die korrekten Pfade zu den installierten Software-Komponenten benötigen. Es müssen für manche Komponenten aber auch Initialisierungen durchgeführt werden.

Die im Folgenden beschriebenen Vorgehensweisen beziehen sich auf Windows XP Betriebssysteme. Sämtliche Batch-Dateien sind aber auch im Linux-Format vorhanden, so dass die hier beschriebenen Anpassungen auch leicht unter Linux zu verwirklichen sind.

A.3.1 Pfade

Im folgenden werden diese Abkürzungen für spezielle Verzeichnisse verwendet:

- *Update-Service Hauptverzeichnis* <update-service>
- *MySQL-Installationsverzeichnis* <mysql>
- *Java-Installationsverzeichnis* <java>
- *OpenLDAP-Installationsverzeichnis* <ldap>

Damit die Installation und Ausführung des Update-Services funktionieren kann, befinden sich auf der Update-Service CD Jar- und Batch-Dateien. In der Datei

`<update-service>/set_env.bat`

müssen folgende Pfade angepasst werden. Es sollten nur Pfad-Namen verwendet werden, die keine Leerzeichen enthalten.

- ***UMP_BASE*** Pfadangabe zum Update-Service Installations-Pfad
- ***JAVA_PATH*** Pfadangabe zur Java-Installation
- ***ANT_HOME*** Pfadangabe zur ANT-Installation
- ***MYSQL_HOME*** Pfadangabe zur MySQL-Installation
- ***LDAP_HOME*** Pfadangabe zur LDAP-Installation

A.3.2 Java

Zusätzlich zum JDK wird eine Erweiterung der im JDK integrierten Java Cryptographic Extension (JCE) benötigt, die „Unlimited Strength Java(TM) Cryptography Extension (JCE) Policy Files“. Diese Erweiterung befindet sich auf der CD in der Datei

`/Software/Java/JDK/jce_policy-1_4_2.zip`.

In der Archiv-Datei befindet sich die Installationsanleitung.

A.3.3 MySQL

Die notwendigen Batch-Dateien zur Manipulation der MySQL-Datenbank befinden sich im Verzeichnis `<update-service>/bin/MySQL`. Damit die Batch-Dateien korrekt funktionieren können ist es notwendig, dass die Umgebungsvariable `MYSQL_HOME` auf das MySQL-Installationsverzeichnis `<mysql>` zeigt.

Bevor die Datenbank angepasst werden kann, muss der Datenbank-Server gestartet werden. Unter Windows wird der Server als Dienst mit dem Befehl

`<mysql>/bin/mysqld-max-nt.exe -- install-manual`

installiert. Danach wird der Service mit der Batch-Datei

`<update-service>/bin/MySQL/startMySQL-NT.bat`

gestartet und mit

`<update-service>/bin/MySQL/stopmysql-NT.bat`

wieder angehalten.

Details zur MySQL-Datenbank und ihrer Benutzung kann man der mitgelieferten Bedienungsanleitung unter `<mysql>/Docs/manual.html` entnehmen.

Nachdem der Datenbank-Server gestartet ist, müssen die Benutzer und Zugriffsrechte sowie die Update-Service Tabellen erzeugt werden. Dafür gibt es zwei Batch-Dateien:

- ***setup_user.bat***
Dieses Script ändert das Root-Passwort, das direkt nach der Installation standardmäßig leer ist. Außerdem wird der Update-Service Benutzer angelegt und mit einem Passwort versehen.

- ***create_database.bat***
Damit der Update-Service seine Daten in die Datenbank eintragen kann, müssen zuvor die notwendigen Tabellen erstellt werden (vergl. Kapitel 4.5).

Es sind noch weitere Scripte in dem Verzeichnis vorhanden, die nun beschrieben werden.

- ***clear_database.bat***
Löscht sämtliche Tabellen-Einträge. Versetzt also die Datenbank in den Zustand direkt nach der Erzeugung.
- ***drop_database.bat***
Hiermit wird die Update-Service Datenbank komplett gelöscht (danach ist der Update-Service nicht mehr lauffähig).

A.3.4 LDAP

Nach der OpenLDAP-Installation müssen noch Grundeinstellungen durchgeführt werden. Dazu müssen alle Dateien samt Unterverzeichnisse von der Update-Service CD im Verzeichnis ab

`/Software/LDAP/config/`

in das Verzeichnis `<ldap>` kopiert und evtl. vorhandene Dateien überschrieben werden. Die so vorgenommenen Einstellungen initialisieren den LDAP mit den erwarteten Einstellungen. Der LDAP-Server sollte wie auch schon der MySQL-Server unter Windows als Service installiert werden. Dies geschieht mittels

`<ldap>/slapd.exe install`

Danach wird der LDAP-Server über

`<update-service>/bin/LDAP/startLDAP.bat`

gestartet und über

`<update-service>/bin/LDAP/stopLDAP.bat`

angehalten.

Ist der LDAP-Server gestartet, so muss nun noch der root-Knoten eingefügt werden. Das erledigt die Batch-Datei

`<ldap>/set_LDAP_root-Node.bat` .

Nun ist der LDAP-Server konfiguriert und kann mit Einträgen gefüllt werden. Der Update-Service erwartet, dass das LDAP-Verzeichnis nach dem X.500-Standard aufgebaut ist. Die Knoten, in denen die Zertifikate gespeichert sind, müssen die Objektklasse „*x509userCertificate*“ besitzen. Folgende Attribute müssen in den Knoten gespeichert sein:

- ***x509signatureAlgorithm***
- ***x509issuer***
- ***x509serialNumber***
- ***x509validityNotBefore***
- ***x509validityNotAfter***
- ***x509subjectPublicKeyInfoAlgorithm***
- ***x509userCert***

Diese Attribute dienen dazu, dass Informationen über die gespeicherte Zertifikate schnell abgefragt werden können, ohne dass zeitaufwändig das gesamte Zertifikat gelesen und ausgewertet werden muss. Abbildung 4.4 zeigt den schematischen Aufbau der LDAP-Einträge.

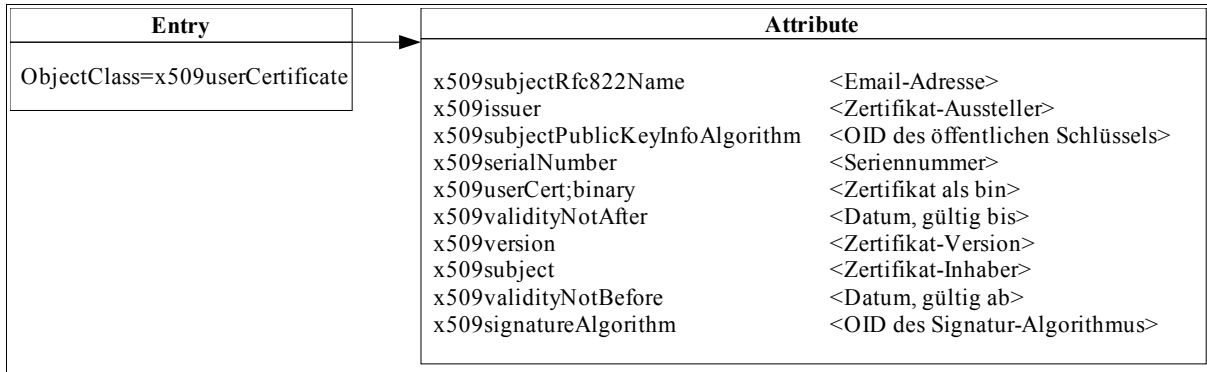


Abbildung 4.4: Schema der LDAP-Einträge

Werden zu Testzwecken neue LDAP-Einträge benötigt, so können diese generiert werden. Es gibt eine Test-Klasse, die aus den Daten der Datei

```
<update-service>/config/LDAP/UMP-Test-Certificates-Input.txt
```

Beispiel-Zertifikate erstellt und diese ins LDAP-Verzeichnis stellt. Die Test-Klasse kann mit der Batch-Datei

```
<update-service>/bin/generate_sample_certificates.bat
```

gestartet werden.

A.4 Update-Service

Im folgenden wird beschrieben, wie der Update-Service compiliert und gestartet wird und welche Anpassungen vorgenommen werden müssen.

A.4.1 Compilierung

Sollte es einmal nötig sein, die Quelldateien des Update-Services neu zu übersetzen, genügt es, die Batch-Datei

```
<update-service>/build.bat
```

auszuführen. Dadurch werden die notwendigen Scripte aufgerufen und ANT ausgeführt. Die so erzeugte Datei „UpdateService.jar“ wird im „lib“-Verzeichnis abgelegt.

A.4.2 Ausführung

Bevor der Update-Service gestartet werden kann, sollte sichergestellt werden, dass die MySQL- (Kapitel A.3.3) und LDAP-Server (Kapitel A.3.4) gestartet und initialisiert sind.

Danach wird zum Starten des Update-Services die Batch-Datei

`<update-service>/updateservice.bat`
ausgeführt. Sie startet zuerst die notwendigen Scripte, die die Umgebungsvariablen und den korrekten Classpath setzen und führt danach das Update-Service Java-Programm aus.


```

ClientIdentifier ::= SEQUENCE
{
  clientClassification      INTEGER,           -- 0: Autonomous
                           1: NonAutonomous
  clientId                  UTF8String,         -- Z.B. "TPS", "JavaCard",
                                                "Outlook Plugin", ...
  clientVersion             UTF8String,         -- Z.B. "1.0", "1.1", ...
  clientSerialNumber        UTF8String OPTIONAL, -- Z.B. "12345", ...
}
UmpApJob ::= SEQUENCE
{
  enumeration              INTEGER,
  action                   ApAction
}
UmpNapJob ::= SEQUENCE
{
  enumeration              INTEGER,
  action                   NapAction
}
ApAction ::= CHOICE
{
  -- wird weiter ausgebaut und beschreibt die Transaktionen

  installProvider          [0] InstallProvider,
  installRegistry          [1] InstallRegistry,
  installTrustAnchor       [2] InstallTrustAnchor,
  installChKey             [3] InstallChKey,
  deleteAlgorithm          [4] DeleteAlgorithm,
  deleteTrustAnchor        [5] DeleteTrustAnchor,
  deleteChKey              [6] DeleteChKey,
  installChCertificate      [7] InstallChCertificate
}
NapAction ::= CHOICE
{
  -- wird weiter ausgebaut und beschreibt die Transaktionen

  code                     [0] BIT STRING,
  installProvider          [1] InstallProvider
}

```

B.2 UC - Definierte Aktionen

```

InstallProvider ::= SEQUENCE
{
  url                      UTF8String -- URL, where the Provider can be
                                   downloaded from as
                                   UpdateComponentCode
  uccId                    Integer     -- unique identifier for
                                   UpdateComponentCode
}
InstallRegistry ::= SEQUENCE
{
  registry                 Registry
}
InstallTrustAnchor ::= SEQUENCE
{
  trustAnchor              Certificate
}
InstallChKey ::= SEQUENCE
{
  chKey                    ChKeyToInstall
}

```

```

InstallChCertificate ::= SEQUENCE
{
  certificate          Certificate,
  keyId                UTF8String OPTIONAL
}
DeleteAlgorithm ::= SEQUENCE
{
  algorithm            AlgorithmIdentifier
}
DeleteTrustAnchor ::= SEQUENCE
{
  trustAnchor         KeyOrCertToDelete
}
DeleteChKey ::= SEQUENCE
{
  chKey KeyOrCertToDelete
}

```

B.3 Komponenten

```

Protection ::= SEQUENCE
{
  -- certificate is always expected, when used within
  -- UpdateComponentResponse, UCRActiveClientRelevant

  signatureAlg        AlgorithmIdentifier,
  signature            BIT STRING,
  certificate          Certificate OPTIONAL
}
AlgorithmIdentifier ::= SEQUENCE
{
  algorithm            OBJECT IDENTIFIER,
  parameters          ANY DEFINED BY algorithm OPTIONAL
}
ChKeyToInstall ::= SEQUENCE
{
  algorithm            AlgorithmIdentifier, -- generate key for this
                                     algorithm
  usage                KeyUsage OPTIONAL, -- for this usage
  keyLength            INTEGER           -- with this key length in
                                     PSE
}
KeyOrCertToDelete ::= SEQUENCE
{
  cert                [0] Certificate OPTIONAL,
  algorithm            [1] AlgorithmIdentifier OPTIONAL,
                                     -- Lösche Schlüssel/
                                     Zertifikate zu diesem
                                     Verfahren und
  usage                [2] KeyUsage OPTIONAL -- diesem Usage
  keyLength            [3] INTEGER OPTIONAL -- maximale kritische
                                     Schlüssellänge
  publicKey            [4] SubjectPublicKeyInfo OPTIONAL
}
KeyUsage ::= BIT STRING
{
  digitalSignature(0), -- security mechanism other than non Repudiation
  nonRepudiation(1),  -- digital signature of a document
  keyEncipherment(2), -- public key is used for key transport
  dataEncipherment(3), -- for enciphering user data
  keyAgreement(4),    -- Diffie-Hellman key management
  keyCertSign(5),     -- for certificate signing
  cRLSign(6),         -- for revocation information signing
  encipherOnly(7),    -- belongs to keyEncipherment: for enciphering
  decipherOnly(8)     -- belongs to keyEncipherment: for deciphering
}

```

```

SubjectPublicKeyInfo ::= SEQUENCE
{
    algorithm                AlgorithmIdentifier,
    subjectPublicKey          BIT STRING
}

```

B.4 UpdateComponentCode

```

UpdateComponentCode ::= SEQUENCE
{
    uccId                    Integer,
    clientIdentifier         [0] ClientIdentifier OPTIONAL,
    provider                 SEQUENCE OF Provider,
    protection               SEQUENCE OF Protection
}
Provider ::= CHOICE
{
    jcaProvider              [0] JCAProvider,
    code                     [1] BIT STRING
}
JCAProvider ::= SEQUENCE
{
    providerName             UTF8, -- e.g. FlexiNFProvider
    providerClassName        UTF8, -- e.g. de.flexiprovider.nf.FlexiNFProvider
    providerFileName         UTF8, -- e.g. flexinfprovider.jar
    code                     BIT STRING
}

```

B.5 UpdateComponentResponse

```

UpdateComponentResponse ::= SEQUENCE
{
    version                  INTEGER DEFAULT 2,
    apRelevants              [1] SEQUENCE OF
        UCRAutonomousPseRelevant OPTIONAL,
    napRelevants             [2] SEQUENCE OF
        UCRNonAutonomousPseRelevant OPTIONAL,
    getClientIdentifierResponse [3] SEQUENCE OF
        GetClientIdentifierResponse OPTIONAL
    protections              SEQUENCE OF Protection OPTIONAL
}
UCRAutonomousClientRelevant ::= SEQUENCE
{
    umpId                    INTEGER,
    clientIdentifier         ClientIdentifier OPTIONAL,
    messageClientId          INTEGER OPTIONAL,
    apdone                   SEQUENCE OF ApDone,
    protections              SEQUENCE OF Protection
}
UCRNonAutonomousClientRelevant ::= SEQUENCE
{
    umpId                    INTEGER,
    clientIdentifier         ClientIdentifier OPTIONAL,
    napdones                  SEQUENCE OF NapDone
}
ApDone ::= SEQUENCE
{
    enumeration              INTEGER,
    response                  ApResponse
}

```

```

NapDone ::= SEQUENCE
{
  enumeration          INTEGER,
  response              BIT STRING
}
ApResponse ::= CHOICE
{
  genericResponse      [0] GenericResponse      EXPLICIT
  installChKeyResponse [1] InstallChKeyResponse EXPLICIT
}
GenericResponse ::= SEQUENCE
{
  ok                    Integer                -- = 0: OK!
                                                              -- != 0: Error
  messagetext           UTF8String OPTIONAL -- Beschreibung im Klartext
}
InstallChKeyResponse ::= SEQUENCE
{
  ok                    Integer, -- = 0: OK!
                                                              -- != 0: Error
  messagetext           [0] UTF8String OPTIONAL, -- Beschreibung im
                                                              Klartext
  subjectPKInfo         [1] SEQUENCE OF SubjectPublicKeyInfo
                                                              {{ PKInfoAlgorithmus }} OPTIONAL,
                                                              -- neuer Schlüssel entsprechend
                                                              PKCS#10 kodiert
  keyId                 [2] UTF8String OPTIONAL -- vom Client zu
                                                              vergebender Identifier
}
GetClientIdentifierResponse ::= SEQUENCE
{
  umpId                 INTEGER,
  clientIdentifier      ClientIdentifier,
  chCertificate         Certificate,
  protections           SEQUENCE OF Protection OPTIONAL
}

```


Anhang C Registry

Die Registry ist ein wichtiges Instrument, das dem Update-Service angibt, welche Algorithmen unterstützt werden und wie sie voneinander abhängen.

Die eigentliche Registry ist ein ASN.1 Datenkonstrukt. Da aber diese Darstellung nicht für Menschen lesbar ist, wurde zum Erzeugen der Registry das XML-Format gewählt.

C.1 Die Registry im XML-Format

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE UMP:umpregistry SYSTEM "umpregistry.dtd">
<UMP:umpregistry xmlns:UMP="UMP">
  <UMP:primitivelist>
    <UMP:primitive name="Factoring problem" type="BaseProblem">
      <UMP:id>1001</UMP:id>
      <UMP:oid>1.3.6.1.4.1.8301.3.3.2.7.1</UMP:oid>
      <UMP:contained_in>
        <UMP:id>1211</UMP:id>
        <UMP:id>1212</UMP:id>
        <UMP:id>1241</UMP:id>
        <UMP:id>2101</UMP:id>
        <UMP:id>2102</UMP:id>
        <UMP:id>2103</UMP:id>
        <UMP:id>2104</UMP:id>
        <UMP:id>2105</UMP:id>
        <UMP:id>2106</UMP:id>
        <UMP:id>2107</UMP:id>
        <UMP:id>2109</UMP:id>
        <UMP:id>2110</UMP:id>
        <UMP:id>2201</UMP:id>
      </UMP:contained_in>
      <UMP:securityconditions>
        <UMP:orcondition>
          <UMP:one>1</UMP:one>
          <UMP:two>1</UMP:two>
        </UMP:orcondition>
        <UMP:orcondition>
          <UMP:one>1</UMP:one>
          <UMP:two>1</UMP:two>
        </UMP:orcondition>
      </UMP:securityconditions>
    </UMP:primitive>
    <UMP:primitive name="Discrete logarithm problem finite fields"
      type="BaseProblem">
      <UMP:id>1002</UMP:id>
      <UMP:oid>1.3.6.1.4.1.8301.3.3.2.7.2</UMP:oid>
      <UMP:contained_in>
        <UMP:id>0</UMP:id>
      </UMP:contained_in>
      <UMP:securityconditions>
        <UMP:orcondition>
          <UMP:one>1</UMP:one>
          <UMP:two>1</UMP:two>
        </UMP:orcondition>
      </UMP:securityconditions>
    </UMP:primitive>
  </UMP:primitivelist>
</UMP:umpregistry>
```

```

    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="Discrete logarithm problem elliptic curve"
  type="BaseProblem">
  <UMP:id>1003</UMP:id>
  <UMP:oid>1.3.6.1.4.1.8301.3.3.2.7.3</UMP:oid>
  <UMP:contained_in>
    <UMP:id>2301</UMP:id>
    <UMP:id>2401</UMP:id>
    <UMP:id>1221</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="Discrete logarithm problem imaginary quadratic fields"
  type="BaseProblem">
  <UMP:id>1004</UMP:id>
  <UMP:oid>1.3.6.1.4.1.8301.3.3.2.7.4</UMP:oid>
  <UMP:contained_in>
    <UMP:id>2501</UMP:id>
    <UMP:id>2502</UMP:id>
    <UMP:id>2504</UMP:id>
    <UMP:id>1231</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="SHA1" type="MessageDigest">
  <UMP:id>1101</UMP:id>
  <UMP:oid>1.3.14.3.2.26</UMP:oid>
  <UMP:contained_in>
    <UMP:id>2103</UMP:id>
    <UMP:id>2104</UMP:id>
    <UMP:id>2201</UMP:id>
    <UMP:id>2301</UMP:id>
    <UMP:id>2401</UMP:id>
    <UMP:id>2501</UMP:id>
    <UMP:id>2502</UMP:id>
    <UMP:id>2504</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="MD5" type="MessageDigest">

```

```
<UMP:id>1102</UMP:id>
<UMP:oid>1.2.840.113549.2.5</UMP:oid>
<UMP:contained_in>
  <UMP:id>2101</UMP:id>
  <UMP:id>2102</UMP:id>
</UMP:contained_in>
<UMP:securityconditions>
  <UMP:orcondition>
    <UMP:one>1</UMP:one>
    <UMP:two>1</UMP:two>
  </UMP:orcondition>
  <UMP:orcondition>
    <UMP:one>1</UMP:one>
    <UMP:two>1</UMP:two>
  </UMP:orcondition>
</UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="MD4" type="MessageDigest">
  <UMP:id>1103</UMP:id>
  <UMP:oid>1.2.840.113549.2.4</UMP:oid>
  <UMP:contained_in>
    <UMP:id>0</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="RIPEMD160" type="MessageDigest">
  <UMP:id>1104</UMP:id>
  <UMP:oid>1.3.36.3.2.1</UMP:oid>
  <UMP:contained_in>
    <!-- <UMP:id>2109</UMP:id> -->
    <UMP:id>2110</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="RIPEMD128" type="MessageDigest">
  <UMP:id>1105</UMP:id>
  <UMP:oid>1.3.36.3.2.2</UMP:oid>
  <UMP:contained_in>
    <UMP:id>0</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="RSAPKCS115" type="SignatureAlgorithm">
  <UMP:id>1211</UMP:id>
  <UMP:oid>1.2.840.113549.1.1.1</UMP:oid>
```

```

<UMP:contained_in>
  <UMP:id>1212</UMP:id>
  <UMP:id>1241</UMP:id>
  <UMP:id>2101</UMP:id>
  <UMP:id>2102</UMP:id>
  <UMP:id>2103</UMP:id>
  <UMP:id>2104</UMP:id>
  <UMP:id>2105</UMP:id>
  <UMP:id>2106</UMP:id>
  <UMP:id>2107</UMP:id>
  <UMP:id>2109</UMP:id>
  <UMP:id>2110</UMP:id>
</UMP:contained_in>
<UMP:securityconditions>
  <UMP:orcondition>
    <UMP:one>1</UMP:one>
    <UMP:two>1</UMP:two>
  </UMP:orcondition>
  <UMP:orcondition>
    <UMP:one>1</UMP:one>
    <UMP:two>1</UMP:two>
  </UMP:orcondition>
</UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="RSAPKCS121" type="">
  <UMP:id>1212</UMP:id>
  <UMP:oid>1.2.840.113549.1.1.7</UMP:oid>
  <UMP:contained_in>
    <UMP:id>0</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="ECDSA" type="SignatureAlgorithm">
  <UMP:id>1221</UMP:id>
  <UMP:oid>1.2.840.10045.2.1</UMP:oid>
  <UMP:contained_in>
    <UMP:id>2301</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="IQDSA" type="SignatureAlgorithm">
  <UMP:id>1231</UMP:id>
  <UMP:oid>1.3.6.1.4.1.8301.3.4.2</UMP:oid>
  <UMP:contained_in>
    <UMP:id>2501</UMP:id>
    <UMP:id>2502</UMP:id>
    <UMP:id>2504</UMP:id>
  </UMP:contained_in>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>

```

```

        <UMP:one>1</UMP:one>
        <UMP:two>1</UMP:two>
    </UMP:orcondition>
</UMP:securityconditions>
</UMP:primitive>
<UMP:primitive name="DSA" type="SignatureAlgorithm">
    <UMP:id>1241</UMP:id>
    <UMP:oid>1.2.3</UMP:oid>
    <UMP:contained_in>
        <UMP:id>2201</UMP:id>
    </UMP:contained_in>
    <UMP:securityconditions>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
    </UMP:securityconditions>
</UMP:primitive>
</UMP:primitivelist>
<UMP:constructedlist>
    <UMP:constructed name="SHA1WITHIQDSA" type="Signature">
        <UMP:id>2501</UMP:id>
        <UMP:oid>1.3.6.1.4.1.8301.3.2.2</UMP:oid>
        <UMP:securityconditions>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>
                <UMP:two>1</UMP:two>
            </UMP:orcondition>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>
                <UMP:two>1</UMP:two>
            </UMP:orcondition>
        </UMP:securityconditions>
    </UMP:constructed>
    <UMP:constructed name="SHA1withIQRDSA" type="Signature">
        <UMP:id>2502</UMP:id>
        <UMP:oid>1.3.6.1.4.1.8301.3.2.1</UMP:oid>
        <UMP:securityconditions>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>
                <UMP:two>1</UMP:two>
            </UMP:orcondition>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>
                <UMP:two>1</UMP:two>
            </UMP:orcondition>
        </UMP:securityconditions>
    </UMP:constructed>
    <UMP:constructed name="SHA1withIQGQ" type="Signature">
        <UMP:id>2504</UMP:id>
        <UMP:oid>1.3.6.1.4.1.8301.3.2.3</UMP:oid>
        <UMP:securityconditions>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>
                <UMP:two>1</UMP:two>
            </UMP:orcondition>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>
                <UMP:two>1</UMP:two>
            </UMP:orcondition>
        </UMP:securityconditions>
    </UMP:constructed>
    <UMP:constructed name="MD5WITHRSAPKCS115" type="Signature">
        <UMP:id>2101</UMP:id>
        <UMP:oid>1.3.14.3.2.25</UMP:oid>
        <UMP:securityconditions>
            <UMP:orcondition>
                <UMP:one>1</UMP:one>

```

```

        <UMP:two>1</UMP:two>
    </UMP:orcondition>
</UMP:orcondition>
    <UMP:one>1</UMP:one>
    <UMP:two>1</UMP:two>
</UMP:orcondition>
</UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="MD5WITHRSAPKCS115" type="Signature">
    <UMP:id>2102</UMP:id>
    <UMP:oid>1.2.840.113549.1.1.4</UMP:oid>
    <UMP:securityconditions>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
    </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA1WITHRSAPKCS115" type="Signature">
    <UMP:id>2103</UMP:id>
    <UMP:oid>1.3.14.3.2.29</UMP:oid>
    <UMP:securityconditions>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
    </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA1WITHRSAPKCS115" type="Signature">
    <UMP:id>2104</UMP:id>
    <UMP:oid>1.2.840.113549.1.1.5</UMP:oid>
    <UMP:securityconditions>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
    </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA256WITHRSAPKCS115" type="Signature">
    <UMP:id>2105</UMP:id>
    <UMP:oid>2.16.840.1.101.3.4.2.1</UMP:oid>
    <UMP:securityconditions>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>
    </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA384WITHRSAPKCS115" type="Signature">
    <UMP:id>2106</UMP:id>
    <UMP:oid>2.16.840.1.101.3.4.2.2</UMP:oid>
    <UMP:securityconditions>
        <UMP:orcondition>
            <UMP:one>1</UMP:one>
            <UMP:two>1</UMP:two>
        </UMP:orcondition>

```

```

    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA512WITHRSAPKCS115" type="Signature">
  <UMP:id>2107</UMP:id>
  <UMP:oid>2.16.840.1.101.3.4.2.3</UMP:oid>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA1WITHDSA" type="Signature">
  <UMP:id>2201</UMP:id>
  <UMP:oid>1.2.840.10040.4.3</UMP:oid>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA1WITHDSA" type="Signature">
  <UMP:id>2109</UMP:id>
  <UMP:oid>1.3.14.3.2.27</UMP:oid>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="RIPEMD160WITHRSA" type="Signature">
  <UMP:id>2110</UMP:id>
  <UMP:oid>1.3.36.3.3.1.2</UMP:oid>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:constructed>
<UMP:constructed name="SHA1WITHECDSA" type="Signature">
  <UMP:id>2301</UMP:id>
  <UMP:oid>1.2.840.10045.4.1</UMP:oid>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>

```

```
        <UMP:two>1</UMP:two>
      </UMP:orcondition>
    </UMP:securityconditions>
  </UMP:constructed>
<UMP:constructed name="SHA1WITHECNR" type="Signature">
  <UMP:id>2401</UMP:id>
  <UMP:oid>1.2.3</UMP:oid>
  <UMP:securityconditions>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
    <UMP:orcondition>
      <UMP:one>1</UMP:one>
      <UMP:two>1</UMP:two>
    </UMP:orcondition>
  </UMP:securityconditions>
</UMP:constructed>
</UMP:constructedlist>
</UMP:umpregistry>
```

Anhang D Verzeichnisse

D.1 Literaturverzeichnis

- ANT: The Apache Software Foundation, The Apache Ant Project, 2004, <http://ant.apache.org/index.html>
- BSI: Bundesamt für Sicherheit in der Informationstechnik, , 2004, <http://www.bsi.de/index.htm>
- BUCH: Johannes Buchmann, Einführung in die Kryptographie, 1999, Springer, ISBN:3-540-66059-3
- BuRT: Johannes Buchmann, Markus Ruppert, Markus Tak, FlexiPKI- Realisierung einer flexiblen Public-Key-Infrastruktur, 1999, www.informatik.tu-darmstadt.de/TI/Veröffentlichung
- FLEXI: TU-Darmstadt, FlexiProvider, 2004, <http://www.flexiprovider.de/>
- Hart: Michael Hartmann, Ges. Austausch von kryptogr. Basiskomponenten in Personal Security Env., 2004
- JAVA: Sun, Java TechnologyJava 2 Platform, Standard Edition (J2SE), 2004, <http://java.sun.com/j2se/>
- KALE: Igor Kalenderian, Impl. des Austausches kryptogr. Kompon. mittels UMP, November 2001, <http://www.informatik.tu-darmstadt.de/TI/>
- LDAP: Network Working Group, Internet X.509 Public Key Infrastructure-Operational Protocols-LDAPv2, April 1999
- MaHa: Sönke Maseberg, Michael Hartmann, Fail-Safe-Konzept für Public-Key-Infrastrukturen, 2002, <http://www.informatik.tu-darmstadt.de/TI/>
- MYSQL: MySQL AB, MySQL Database, 2004, <http://dev.mysql.com/>
- OPENLDAP: OpenLDAP Foundation, Community developed LDAP software, 2004, <http://www.openldap.org/>
- PKIX: IETF Working Group, Public-Key Infrastructure (X.509) (pkix), 2004, <http://www.ietf.org/html.charters/pkix-charter.htm>
- SCHRAMM: Dirk Schramm, Entwicklung einer flexiblen S/MIME-Erweiterung für Microsoft-Outlook, Januar 2001, <http://www.informatik.tu-darmstadt.de/TI/>
- X.509: International Telecommunications Union, ITU-T Recommendation X.509 (1997 E), Juni 1997

D.2 Abbildungsverzeichnis

Abbildung 2.1: Fail-Safe-PKI.....	11
Abbildung 2.2: Schematischer Aufbau eines ClientIdentifiers.....	15
Abbildung 2.3: Schematischer Aufbau der Registry.....	15
Abbildung 2.4: Schematischer Aufbau eines UpdateComponents.....	16
Abbildung 2.5: Schematischer Aufbau eines UpdateComponentResponses.....	17
Abbildung 2.6: Zusammenhang zwischen UpdateComponentCode und Provider.....	18
Abbildung 3.1: Schematischer Ablauf eines Schadensfalles.....	21
Abbildung 3.2: Zusammenhang zwischen den Komponenten des Update-Service.....	22
Abbildung 3.3: Status-Fenster.....	23
Abbildung 3.4: Auswahl des eingetretenen Schadens.....	25
Abbildung 3.5: Auswahl eines CH-Zertifikats.....	26
Abbildung 3.6: CA-Zertifikat auswählen.....	26
Abbildung 3.7: Hashfunktion auswählen.....	27
Abbildung 3.8: Signaturalgorithmus auswählen.....	27
Abbildung 3.9: Signaturverfahren auswählen.....	28
Abbildung 3.10: Algorithmus mit dem kompromittierten Paddingverfahren auswählen.....	28
Abbildung 3.11: Mathematisches Basisproblem auswählen.....	29
Abbildung 3.12: Registry auswählen.....	29
Abbildung 3.13: Einzelnen CH-Schlüssel löschen.....	30
Abbildung 3.14: Gruppe von CH-Schlüsseln löschen.....	30
Abbildung 3.15: Neuen CH-Schlüssel generieren.....	31
Abbildung 3.16: Sicherheitsanker löschen.....	31
Abbildung 3.17: Sicherheitsanker austauschen.....	31
Abbildung 3.18: Neue Zertifikate für CHs installieren.....	32
Abbildung 3.19: Algorithmen löschen.....	32
Abbildung 3.20: Neuen Sicherheitsanker installieren.....	33
Abbildung 3.21: Mehrere neue CH-Schlüssel generieren.....	33
Abbildung 3.22: Neuen Provider auswählen.....	34
Abbildung 3.23: Zuordnung Ucc-ID + ClientIdentifier → Provider.....	35
Abbildung 3.24: Auswahl der Klasse eines neuen Providers.....	36
Abbildung 3.25: Angaben eines JCA-Provider.....	36
Abbildung 3.26: Erweiterte Eingaben.....	37
Abbildung 3.27: Angabe, welches UpdateComponent und welcher Client bearbeitet werden sollen.....	38
Abbildung 3.28: Vorhandene CAs.....	38
Abbildung 3.29: Neue CA installieren.....	39
Abbildung 3.30: Infos anzeigen.....	39
Abbildung 3.31: UpdateComponent-Status.....	40
Abbildung 3.32: Datenbankeinträge löschen.....	41
Abbildung 4.1: Zusammenhang zwischen Update-Service und CAs.....	49
Abbildung 4.2: Tabellen der Datenbank.....	51
Abbildung 4.3: Verzeichnisstruktur der Update-Service CD.....	53
Abbildung 4.4: Schema der LDAP-Einträge.....	58