

Entwurf und Implementierung einer
Infrastrukturkomponente für ein Java-basiertes
Trustcenter

Thilo Planz ¹

Januar 2002

Diplomarbeit
am Lehrstuhl für Theoretische Informatik
der Technischen Universität Darmstadt

Prof. Dr. J. Buchmann

Betreuer: Markus Ruppert

¹TU Darmstadt, Fachbereich Rechts- und Wirtschaftswissenschaften, Matrikelnr.
547479, <mailto:planz@epost.de>

Anmerkung zur vorliegenden Fassung

Dieser Text ist eine leicht überarbeitete Fassung meiner Diplomarbeit, die ich am 30. Januar 2002 der Prüfungsbehörde des Fachbereichs Rechts- und Wirtschaftswissenschaften vorgelegt habe. Die Änderungen erfolgten zur Korrektur von Rechtschreibfehlern und aufgrund von Anmerkungen seitens meines Betreuers Markus Ruppert.

Darmstadt, den 15. März 2002

Thilo Planz
planz@epost.de

Vorwort

Die vorliegende Arbeit ist Teil des umfangreichen FlexiTrust-Projekts, welches die Entwicklung einer leistungsfähigen Trustcenter-Software zum Ziel hat. Im Rahmen von FlexiTrust sind zahlreiche Studien- und Diplomarbeiten entstanden, die in teilweise engem Zusammenhang mit meiner Arbeit stehen.

Insbesondere seien hier die Diplomarbeiten von Alexander Wiesmaier [Wie01] und von Markus Tak [Tak99], in denen der Aufbau der eigentlichen Zertifizierungsinstanz (FlexiTrust CA) beschrieben ist, sowie die Diplomarbeiten von Jens Dambruch [Dam01] und Markus Schuster [Sch01], in denen die Registrierungsstelle (FlexiTrust RA) entworfen wird, genannt.

Prototypen der Trustcenter-Software, inklusive der in dieser Arbeit vorgestellten Infrastrukturkomponente (FlexiTrust IS), waren über die gesamte Dauer meiner Diplomarbeit bei externen Partnern im praktischen Einsatz. Auf diese Weise konnten die Bedürfnisse der späteren Anwender in die Entwicklung des Projekts einfließen.

Da die Software noch immer weiterentwickelt wird, ist meine Diplomarbeit als ein dokumentarischer Schnappschuß zu verstehen, vor allem was die Details der Implementierung anbetrifft. Für einen genauen Einblick in den aktuellen Stand der Entwicklung sei auf die Dokumentation und den Quelltext im CVS-Repository verwiesen.

Besonderer Dank gilt Markus Ruppert, der die Kommunikation mit den externen Partnern (und den zahlreichen an verschiedenen Stellen beteiligten Studenten) in bemerkenswert souveräner Weise koordinierte, Marcus Lippert als Ansprechpartner für FlexiTrust RA, Alexander Wiesmaier als Verantwortlicher für FlexiTrust CA und Professor Buchmann, der für seine Studenten auch bei ungewöhnlichen Anliegen stets ein offenes Ohr hat.

Inhaltsverzeichnis

1	Einleitung	5
1.1	Kryptographie	5
1.2	Public-Key-Infrastrukturen	9
2	Die Trustcenter-Software FlexiTrust	16
2.1	FlexiTrust RA	16
2.2	FlexiTrust CA	17
2.3	FlexiTrust IS	18
2.4	PKI-Projekte mit FlexiTrust	19
3	Entwurf der Infrastruktur-Komponente	21
3.1	Anforderungen an die Software	21
3.2	Vorbemerkung	25
3.3	Produkt-Datensätze	26
3.4	Produkt-Profile	27
3.5	Operatoren	28
3.6	Ablaufsteuerungssystem für Produkte	29
3.7	Multi-CA-Fähigkeit	32
3.8	Datenbank	32
3.9	IS-Dämon	33
4	Implementierung	35
4.1	Übersicht	35
4.2	Produkt-Datensätze	37
4.3	Produkt-Profile	38
4.4	XML	39
4.5	Operatoren	40
4.6	Datenbankanbindung	44
4.7	Template-Mechanismus	46
4.8	FlexiTrust Administrationsframework	47
4.9	Internationalisierung	49
4.10	Benutzte Bibliotheken	50
5	Betriebsanleitung	52
5.1	Installation	52
5.2	Konfiguration	53
5.3	Starten	56

5.4	Stoppen	56
5.5	die mitgelieferten Operatoren	57
5.6	Formatvorlagen	62
5.7	Datenbankinhalt	63
5.8	Fehlerbehandlung	64
6	Ausblick	66
A	Datenbankschema	68
B	Beispiel für eine CA-Konfigurationsdatei	70
C	Struktur der Konfigurationsdatei	73
D	Abkürzungsverzeichnis	75
	Index	76

Kapitel 1

Einleitung

1.1 Kryptographie

Durch die in den letzten Jahren rasant vorangeschrittene Digitalisierung von Informationen und deren Speicherung auf Rechnersystemen sowie durch die immer umfassendere Vernetzung dieser Rechnersysteme und die damit einhergehenden neuen Möglichkeiten zum Datenaustausch haben sich beträchtliche Probleme bezüglich der Datensicherheit ergeben.

Insbesondere besteht ein großer Widerspruch zwischen dem auf offenen Netzen basierenden Internet, das sich mittlerweile als absolut dominantes Kommunikationsmedium durchgesetzt hat und mit dessen Hilfe sich gewaltige Kostensenkungs- und Effizienzsteigerungen in fast allen rechnergestützten Abläufen erzielen lassen, und den Anforderungen zum Schutz von personenbezogenen oder unternehmenskritischen Daten.

In diesem Zusammenhang hat die mathematische Disziplin der Kryptographie, die sich mit dem Chiffrieren von Informationen beschäftigt, viel Aufmerksamkeit erhalten.

1.1.1 das Problem der sicheren Kommunikation

Kryptographie stellt Methoden bereit, mit denen zwei (oder mehrere) Kommunikationspartner, die nur über einen öffentliche Kanal – in dem Sinne, daß es Dritten möglich ist, die Nachricht abzufangen – miteinander sprechen können, vertrauliche Informationen durch verschlüsselte Nachrichten austauschen können.

In der Literatur werden vier Anforderungen an kryptographisch abgesicherte Kommunikation gestellt:

Vertraulichkeit Ein Außenstehender, der eine verschlüsselte Nachricht abfängt, kann keine Informationen aus ihr gewinnen.

Integrität Es ist dem Außenstehenden auch nicht möglich, den Inhalt der Nachricht zu verändern, ohne daß dies vom vorgesehen Empfänger bemerkt wird.

Authentifizierung Der Empfänger kann prüfen, ob der Verfasser der Nachricht tatsächlich mit dem angegebenen Absender übereinstimmt.

Verbindlichkeit Der Absender der Nachricht kann zu einem späteren Zeitpunkt nicht abstreiten, daß die Nachricht von ihm stammt.

1.1.2 symmetrische Kryptographie

Eine in verschiedenen Ausprägungsformen schon seit der Antike bekannte Anwendung der Kryptographie sind symmetrische Chiffren. Hierbei wird die geheime Botschaft mithilfe einer Verschlüsselungsfunktion in eine Nachricht umgewandelt, aus der ohne Kenntnis der dazugehörigen Entschlüsselungsfunktion die Botschaft nicht mehr hervorgeht.

Während es früher üblich war, die gesamte Verschlüsselungsfunktion geheim zu halten, ist man mittlerweile dazu übergegangen, öffentlich bekannte Funktionen einzusetzen, die lediglich über einen geheim zu haltenden Schlüssel (meist eine Bitfolge) parametrisiert werden. Dies hat neben der Erhöhung der Verfügbarkeit von Implementierungen und der Interoperabilität auch zur Folge, daß diese allgemein bekannten Funktionen der kritischen Kontrolle durch die wissenschaftliche Öffentlichkeit unterliegen, so daß eventuelle Schwächen entdeckt und publik gemacht werden können, bevor ein Schaden (durch kompromittierte Daten) eintritt.

Wenn eine Chiffrierfunktion sicher ist und der Schlüssel geheim gehalten werden kann, so kann die Nachricht nur durch Ausprobieren aller möglichen Schlüssel entziffert werden. Die heute als sicher geltenden Verfahren (zum Beispiel 3DES oder IDEA) verwenden Schlüssellängen von 128 Bit, so daß 2^{128} verschiedene Schlüssel zur Verfügung stehen (das lange Zeit verwendete DES-Verfahren mit 56-Bit-Schlüsseln kann mittlerweile mit wenigen Tagen Rechenzeit gebrochen werden).

Mit symmetrischen Kryptoverfahren ist es also möglich, eine Nachricht in vertrauenswahrer Weise zu speichern oder zu übertragen. Die eingesetzten Verfahren sind zudem sehr effizient zu implementieren, so daß beispielsweise bei der Übertragung verschlüsselter Daten über das Internet die Rechenzeit für die Ver- und Entschlüsselung gegenüber der Transportzeit über das Netzwerk nicht ins Gewicht fällt.

Ein Problem bei diesen Verfahren besteht in der Geheimhaltung der Schlüssel. Dies ist noch zu bewältigen, wenn die Daten nur vom Sender selbst entschlüsselt werden sollen (kryptographische Datenspeicher). Ansonsten muß der Sender allen Empfängern vorab und auf sicherem Wege den Schlüssel zukommen lassen. Wenn der Schlüssel mehrmals verwendet werden soll, müssen alle Beteiligten den Schlüssel fortan geheim halten, und den übrigen Teilnehmern zutrauen, daß diese den Schlüssel ebenfalls geheim halten.

Ein zweites Problem ist die Anzahl der benötigten Schlüssel. Wenn jeder Teilnehmer in der Lage sein soll, jedem anderen Teilnehmer eine Nachricht zu senden, deren Inhalt kein dritter Teilnehmer (und natürlich auch kein Außenstehender) erkennen kann, dann braucht jedes Paar von Teilnehmern einen eigenen Schlüssel. Die Zahl der Schlüssel wächst hierbei quadratisch mit der Anzahl der Teilnehmer. In großen Gemeinschaften (oder gar dem Internet als Ganzem) ist dies völlig unpraktikabel.

1.1.3 asymmetrische Kryptographie

Das Problem der gemeinsam geheimzuhaltenden Schlüssel wird bei der sogenannten asymmetrischen Kryptographie umgangen. In diesem von Whitfield Diffie und Martin Hellman 1976 vorgestellten Verfahren besitzt jeder Teilnehmer einen privaten und einen öffentlichen Schlüssel. Diese beiden Schlüssel bilden dadurch, daß sie in einer gewissen mathematischen Beziehung zueinander stehen, ein Schlüsselpaar. Mit dem öffentlichen Schlüssel eines Empfängers können Nachrichten verschlüsselt werden, die anschließend nur mit dessen privatem Schlüssel wieder entziffert werden können.

Ein Verfahren der asymmetrischen Kryptographie basiert auf einem schwierigen (das heißt für praktische Zwecke nicht lösbaeren) mathematischen Problem, daß es unmöglich macht, den privaten Schlüssel aus dem öffentlichen Schlüssel zu berechnen, obwohl die Beziehung der beiden zueinander allgemein bekannt ist (sonst wäre es nicht als Schlüsselpaar zu verwenden). So wird es möglich, den öffentlichen Schlüssel völlig frei zu verteilen, ohne daß dadurch eine Gefahr für den geheimen Schlüssel (der weiterhin verborgen bleiben muß) entsteht. Ohne hier in Details gehen zu wollen, sei angemerkt, daß man solche Verfahren unter anderem auf dem Problem der Faktorisierung großer Zahlen und dem diskreten Logarithmusproblem aufbauen kann. Eine gute Einführung in den mathematischen Hintergrund bieten Lehrbücher zum Thema [Buc01, Sch96].

Man kann sich die Verwendung asymmetrischer Kryptographie wie einen Hausbriefkasten vorstellen [CKLW00, S. 8]: Jedermann, der im Besitz des öffentlichen Schlüssels ist, kann einen Brief hineinwerfen. Aber nur der Empfänger als Besitzer des privaten Schlüssels kann den Briefkasten öffnen und den Brief wieder herausnehmen.

Die bekannteste konkrete Umsetzung von Diffies und Hellmans Idee ist das nach seinen Erfindern Ronald Rivest, Fiat Shamir und Leonard Adleman benannte RSA-Verfahren, das auf dem Faktorisierungsproblem beruht. Fast überall wo heute asymmetrische Kryptographie zum Einsatz kommt handelt es sich um RSA.

Neben dem vereinfachten Schlüsselmanagement bieten asymmetrische kryptographische Verfahren auch die Möglichkeit digitaler Signaturen (siehe Abschnitt 1.1.5).

Der große Nachteil asymmetrischer Verfahren ist der damit verbundene erheblich höhere Rechenaufwand. Auch sind die Schlüssellängen deutlich größer als bei symmetrischen Verfahren. Beides ist der Notwendigkeit geschuldet das zugrunde liegende mathematische Problem schwierig zu halten. Wenn beispielsweise das RSA-Verfahren verwendet wird, muß verhindert werden, daß ein Teil des öffentlichen Schlüssels (der sogenannte Modulus) faktorisiert werden kann, so daß mit Schlüssellängen von mindestens 1024 Bit (für hochsichere Anwendungen wird 2048 Bit empfohlen) gearbeitet werden muß. Dieser Aufwand macht asymmetrische Verfahren für längere Nachrichten unpraktikabel.

1.1.4 hybride Verfahren

Hybride Verfahren verbinden die Stärken der asymmetrischen (einfaches Schlüsselmanagement) mit den Stärken der symmetrischen Kryptographie (niedriger Rechenaufwand).

Die Idee dahinter ist recht einfach: Die Nachricht wird mit einem symmetrischen Verfahren verschlüsselt, und der dabei verwendete Schlüssel, der ja nur wenige Bytes groß ist, wird mit dem öffentlichen Empfängerschlüssel eines asymmetrischen Verfahrens verschlüsselt und der Nachricht beigelegt.

Der geheime Schlüssel für das symmetrische Verfahren wird dabei zufällig gewählt und nur für eine einzige Nachricht verwendet. Es besteht somit keine Notwendigkeit, ihn auf Absender- oder Empfängerseite zu speichern. Solche Schlüssel werden wegen ihres flüchtigen Charakters als Einmal-Schlüssel (*session key*) oder aufgrund ihrer Verwendung zum Verschlüsseln der eigentlichen Nachricht als *bulk encryption key* bezeichnet.

1.1.5 digitale Signaturen

Asymmetrische Verschlüsselungsverfahren bieten neben der Datenverschlüsselung noch eine weitere wichtige Möglichkeit an: Wenn man nämlich eine Nachricht mit seinem privaten Absenderschlüssel anstelle des öffentlichen Empfängerschlüssels „verschlüsselt“, so erhält man einen Code, der von jedem entschlüsselt werden kann, der den zugehörigen öffentlichen Schlüssel (des Absenders) kennt. Diese Operation kann nur dann gelingen, wenn das verwendete Schlüsselpaar wirklich zusammengehört, funktioniert also nur, wenn der Absender im Besitz des passenden privaten Schlüssels ist. Auf diese Weise lassen sich digitale Signaturen erzeugen.

Eine digitale Signatur für ein Dokument besteht aus der „Verschlüsselung“ des Dokuments mit dem privaten Schlüssel des Signierers (tatsächlich genügt es, einen Hashwert des Dokumentes zu signieren, siehe Abschnitt 1.1.6). Zur Überprüfung (Verifikation) der Signatur wird diese mit dem öffentlichen Schlüssel des Signierers „entschlüsselt“ und mit dem vorliegenden Dokument verglichen.

Beim Vergleich digitaler Signaturen mit handschriftlichen Unterschriften fällt auf, daß digitale Signaturen inhaltsabhängig sind: Da das signierte Dokument in die Berechnung der Signatur eingeht, ist diese untrennbar mit dem Dokument verbunden und kann nicht für andere Dokumente verwendet werden. Diese Eigenschaft ist angesichts der Möglichkeit, digitale Daten leicht zu vervielfältigen, unabdingbar, wäre aber auch sonst für jede Art von Unterschrift wünschenswert, verhindert sie doch die nachträgliche Änderung des Dokuments.

Mit digitalen Signaturen lassen sich die Forderungen nach Authentizität und Verbindlichkeit kryptographischer Nachrichten erfüllen, denn nur dem Absender ist es möglich, eine korrekte Signatur zu erzeugen. Signierte Nachrichten erzwingen automatisch auch die Integrität des Dokumentes. Durch dessen zusätzliche Verschlüsselung mit einem Empfängerschlüssel ist es auch möglich Vertraulichkeit sicherzustellen, so daß alle vier in Abschnitt 1.1.1 genannten Forderungen Beachtung finden.

1.1.6 kryptographische Hashfunktionen

Da, wie bereits mehrfach erwähnt, asymmetrische Verfahren sehr rechenaufwendig sind, verwendet man einen Trick, um im Rahmen digitaler Signaturen möglichst kurze Daten signieren zu müssen. Es genügt nämlich, anstatt des ganzen Dokumentes lediglich eine Prüfsumme zu signieren. Man spricht hier auch von dem Fingerabdruck (*fingerprint*) des Dokuments. Berechnet werden solche Prüfsummen von sogenannten Hashfunktionen (auch *message digest*), die ein beliebig langes Dokument auf einen Funktionswert fester (und kurzer) Länge abbilden.

Die Verifikation einer solchen Signatur verläuft dann derart, daß für das Dokument der Hashwert (auf die genaue Funktion hat man sich zuvor geeinigt) berechnet und mit dem aus der Signatur gewonnenen Hashwert verglichen wird. Wenn die beiden Werte übereinstimmen, kann davon ausgegangen werden, daß dem Signierer tatsächlich das gleiche Dokument vorlag.

Damit dieses Verfahren funktioniert und vor allem auch sicher ist, muß die Hashfunktion einige Eigenschaften erfüllen. Zunächst einmal muß die Hashfunktion effizient auszurechnen sein. Darüber hinaus muß es sich um eine Einwegfunktion handeln, das heißt es muß praktisch unmöglich sein, zu einem gegebenen Hashwert ein passendes Dokument zu finden. Wenn man dies könnte, wäre man in der Lage, zu einer vorab erzeugten Signatur (und dem sich daraus ergebenden Hashwert) ein Dokument zu konstruieren und für dieses eine Signatur zu fälschen. Eine Hashfunktion heißt (stark) kollisionsresistent, wenn es unmöglich ist, zwei Dokumente anzugeben, die zum gleichen Hashwert führen. Obwohl solche Kollisionen natürlich existieren (schließlich werden beliebig lange Dokumente, von denen es unendlich viele verschiedene gibt, auf den endlichen Raum der Hashwerte abgebildet), darf man mit endlichem Aufwand keine Kollision finden können. Es gibt verschiedene Konstruktionsprinzipien für Hashfunktionen, die scheinbar für Kollisionsresistenz sorgen können (auch wenn dies bisher nicht bewiesen werden konnte). Um sich gegen Brute-Force-Attacken (häufiges Berechnen von Hashwerten für verschiedene Eingabewerte) zur Kollisionsfindung schützen zu können (die sogenannte „Geburtstagsattacke“), müssen Hashwerte mindestens 160 Bit lang sein.

1.2 Public-Key-Infrastrukturen

Wie im vorangegangenen Abschnitt geschildert, lassen sich mit dem Methoden der Kryptographie die vier grundlegenden Anforderungen an sichere Kommunikation (Vertraulichkeit, Integrität, Authentifizierung und Verbindlichkeit) erfüllen, wenn jeder Teilnehmer seinen privaten Schlüssel geheimzuhalten vermag und von jedem gewünschten Kommunikationspartner den öffentlichen Schlüssel kennt.

Um den privaten Schlüssel vor unbefugtem Zugriff zu schützen existieren Sicherungsmechanismen wie kennwortgeschützte Dateien und Chipkarten. Die Verteilung der öffentlichen Schlüssel ist Aufgabe einer Public-Key-Infrastruktur.

1.2.1 das Problem des Schlüsselaustauschs

Wie bereits dargelegt ist das Problem des Schlüsselaustausches in symmetrischen Kryptoverfahren, wo der Schlüssel geheim gehalten werden muß, nicht ohne vorherige Absprachen der Teilnehmer möglich. Aber auch in asymmetrischen Verfahren ist dieser Vorgang nicht unproblematisch.

Obwohl der ausgetauschte (öffentliche) Schlüssel selbst keine geheimen Daten enthält, muß doch zumindest dessen Unversehrtheit und vor allem auch die Zuordnung zu den Kommunikationspartnern gewährleistet werden. Die Unversehrtheit des Schlüssels kann mit digitalen Signaturen gesichert werden. Aber wenn jeder Teilnehmer seinen Schlüssel selbst signiert (mit eben diesem Schlüssel), dann muß durch eine zusätzliche Information verhindert werden, daß ein Dritter anstelle des angeblichen Schlüsselinhabers seinen eigenen öffentlichen Schlüssel unter falschem Namen signiert und so eine andere Identität annimmt.

Das Angriffsmuster, jemandem einen falschen öffentlichen Schlüssel für den vermeintlichen Kommunikationspartner unterzuschieben, nennt man *man-in-the-middle*-Attacke. In der Metapher des Hausbriefkastens entspricht dies einem falsch beschrifteten Briefkasten. Um diesen Angriff abzuwehren muß auch bei Public-Key-Kryptoverfahren eine Vertrauensbasis für den Schlüsselaustausch bestehen.

1.2.2 Direct Trust

Die einfachste Möglichkeit die Zuordnung der öffentlichen Schlüssel zu den Teilnehmern sicherzustellen ist es, den öffentlichen Schlüssel direkt vom vermeintlichen Inhaber bestätigen zu lassen. Dies kann beispielsweise dadurch geschehen, daß der Schlüssel persönlich von seinem Inhaber überbracht wird (etwa auf Diskette) oder durch telefonischen Vergleich eines Hashwertes (*finger print*) des Schlüssels.

Offensichtlich funktioniert dieses Verfahren nur in kleinen Benutzergruppen und kann auch nicht verwendet werden um mit Unbekannten zu kommunizieren.

1.2.3 das System der Vertrauensketten

Eine Erweiterung des obigen Verfahrens sieht die Verbreitung der öffentlichen Schlüssel entlang von Vertrauensketten vor. Hierbei signieren die Teilnehmer gegenseitig Ihre öffentlichen Schlüssel und bürgen damit für die Identität des jeweils anderen. Idealerweise bildet sich dann zwischen zwei beliebigen Netzteilnehmern eine Kette von Bürgen und die beiden können entscheiden, ob ihnen diese Kette glaubwürdig genug erscheint.

Vertrauensketten werden bei der sehr populären PGP-Software von Philip Zimmermann eingesetzt. Neben dem Problem, daß sich nicht immer eine Kette zwischen zwei Teilnehmern finden lassen muß, ist es bei diesem Ansatz schwierig, einen kompromittierten Schlüssel zurückzuziehen (beispielsweise bei Diebstahl der Chipkarte mit dem privaten Schlüssel oder nach dem Ausscheiden eines zuvor zeichnungsberechtigten Mitarbeiters eines Unternehmens).

1.2.4 das System der vertrauenswürdigen Dritten

Eine andere Idee besteht in der Einrichtung von zentralen Zertifizierungsstellen, denen alle Teilnehmer vertrauen können und die den Teilnehmern den Besitz ihrer jeweiligen Schlüsselpaare bestätigen. Jeder Teilnehmer erhält hierbei ein sogenanntes Zertifikat über seinen öffentlichen Schlüssel. Dieses Zertifikat enthält neben dem eigentlichen Schlüssel und dem Namen des Inhabers auch Angaben über die zeitliche Gültigkeit und den Verwendungszweck des Schlüssels. Das Zertifikat wird von der Zertifizierungsstelle signiert.

Damit hat sich das Problem auf die ordnungsgemäße Verteilung der öffentlichen Schlüssel der Zertifizierungsinstanzen reduziert. Durch den zentralisierten Ansatz wird es auch ermöglicht, Zertifikate (und damit Schlüssel) für ungültig zu erklären. Zu diesem Zweck führt und veröffentlicht jede Zertifizierungsstelle eine Revokationsliste.

1.2.5 Aufgaben einer Zertifizierungsinstanz

Die prinzipielle Aufgabe einer Zertifizierungsinstanz (*certification authority*, CA), also die Zuordnung von öffentlichen Schlüsseln zu Teilnehmern der Public-Key-Infrastruktur, läßt sich in eine Reihe von Teilaufgaben gliedern:

Registrierung Hierbei werden die Teilnehmer, die ein Zertifikat benötigen, erfaßt. Die Zertifizierungsstelle muß die Identität des Teilnehmers überprüfen. Dies ist ein kritischer Vorgang, der den Sicherheitsanforderungen aller Teilnehmer genügen muß. Auf dieser Identitätsprüfung basiert das Vertrauen in die später ausgestellten Zertifikate. Je nach Einsatzgebiet kann hierzu ein persönliches Erscheinen des Teilnehmers bei der Zertifizierungsstelle und das Überprüfen von Ausweispapieren notwendig sein. Oft wird der Registrierungsvorgang von den übrigen Aufgaben abgetrennt und einer separaten Registrierungsstelle (*registration authority*, RA) zugewiesen. Nach erfolgter Identitätsprüfung weist die Zertifizierungsstelle dem Teilnehmer eine systemweit eindeutige Kennung zu.

Schlüsselerzeugung Das Schlüsselpaar des Teilnehmers muß in einer gesicherten Umgebung erzeugt werden. Dies muß nicht unbedingt durch die Zertifizierungsstelle erfolgen. Statt dessen kann der Teilnehmer sein Schlüsselpaar auch selbst generieren. Bei der Registrierung muß er dann lediglich seinen öffentlichen Schlüssel mitteilen und (zum Beispiel durch eine digitale Signatur) nachweisen, daß er den zugehörigen geheimen Schlüssel auch besitzt. Dies hat den Vorteil, daß der geheime Schlüssel nicht einmal während seiner Erzeugung jemand anderem als dem Besitzer bekannt ist.

Zertifizierung In diesem Schritt wird das eigentliche Zertifikat erzeugt und mit dem privaten Schlüssel der CA signiert. Dies ist ein sehr sicherheitskritischer Vorgang, da hierbei der geheime CA-Schlüssel benötigt wird. Wird dieser Schlüssel kompromittiert, verlieren alle bisher und in Zukunft ausgestellten Zertifikate ihre Glaubwürdigkeit.

Übermittlung des privaten Schlüssels Falls der Teilnehmer sein Schlüssel-paar selbst generiert hat, entfällt dieser Schritt. Ansonsten muß dafür gesorgt werden, daß der private Schlüssel seinem Besitzer auf sichere Weise mitgeteilt wird. Geeignete Wege hierzu sind Chipkarten oder kennwortgeschützte Dateien. Es ist empfehlenswert, den Empfang des Schlüssels durch seinen Besitzer zu überprüfen, bevor man das Zertifikat veröffentlicht (ohne Zertifikat ist der Schlüssel wertlos).

Veröffentlichung des Zertifikats Um das Schlüsselpaar nutzbar zu machen, muß das Zertifikat den Teilnehmern zur Verfügung gestellt werden. Üblicherweise geschieht dies durch einen Verzeichnisdienst.

Rezertifizierung Aufgrund der begrenzten Gültigkeit der Zertifikate benötigen die Teilnehmer in regelmäßigen Abständen neue Zertifikate. Oft ist es wünschenswert, wenn die CA in einem solchen Fall automatisch ein neues Zertifikat erzeugt und verbreitet. Andererseits kann es auch sinnvoll sein, ein neues Schlüsselpaar zu erzeugen und dieses zu zertifizieren.

Revokationen Eine der Stärken der zentralisierten PKI ist die Möglichkeit, Zertifikate (und damit das Schlüsselpaar) für ungültig zu erklären. Dies wird immer dann notwendig, wenn der geheime Schlüssel kompromittiert wurde. Eine CA muß hierfür eine Revokationsliste führen und veröffentlichen (eine Ergänzung oder Alternative zu Revokationslisten ist das Online Certificate Status Protocol OCSP, bei dem Clients die Gültigkeit von Zertifikaten direkt bei der CA erfragen können).

Neben diesen Kernaufgaben werden einer Zertifizierungsstelle gelegentlich noch weitere Dienste zugeordnet, beispielsweise Zeitstempeldienste, das oben erwähnte OCSP und die Möglichkeit der Verwahrung (*key escrow*) und Wiederherstellung verloren gegangener privater Schlüssel (*key recovery*). Eine um solche Aufgaben erweiterte CA bezeichnet man mitunter als Trustcenter. In dieser Arbeit werden die Begriffe Trustcenter, CA, Zertifizierungsinstanz oder -stelle weitgehend synonym benutzt.

Abschließend soll noch angemerkt werden, daß es in Deutschland durch die Einführung des Signaturgesetzes [SigG01, SigV01] möglich ist, digitale Signaturen zu erzeugen, die einer handschriftlichen Unterschrift in ihrer Rechtsgültigkeit praktisch gleichgestellt sind. Der Betrieb einer signaturgesetzkonformen Zertifizierungsstelle ist jedoch aufgrund der im Gesetz vorgeschriebenen Sicherheitsstandards mit erheblichen Kosten verbunden. Für einen unternehmensweiten PKI-Einsatz kann in den meisten Fällen auf Signaturgesetzkonformität verzichtet werden. Zu weiteren Ausführungen der rechtlichen und sonstigen Anforderungen an den Betrieb einer Zertifizierungsstelle sei hier auf die Fachliteratur verwiesen [CKLW00].

1.2.6 wichtige PKI-Standards

Von entscheidender Bedeutung für die Einsatzmöglichkeiten einer Public-Key-Infrastruktur ist die breite Unterstützung durch eine möglichst große Zahl von

Endanwender-Applikationen. Hierzu sind gemeinsame Standards notwendig, vor allem was den Austausch von PKI-Dokumenten angeht. Die in diesem Abschnitt vorgestellten Standards haben sich mittlerweile gegenüber proprietären Lösungen durchgesetzt und werden in einer großen Zahl von Protokollen, beispielsweise bei SSL (*secure socket layer*) zur geschützten Kommunikation mit Webservern oder bei S/MIME für verschlüsselte und signierte E-Mails, sowie in den gängigen Client-Anwendungen wie dem Microsoft Internet Explorer, dem Netscape Communicator oder Microsoft Outlook verwendet.

X.509

Als Teil der X.500-Standard-Serie, in der der gleichnamige Verzeichnisdienst beschrieben wird, entstand der ITU-Standard X.509 (identisch mit ISO-Standard ISO/IEC 9594-8), der ein Austauschformat für Public-Key-Zertifikate definiert, welches sich in der aktuellen Version 3 zum dominierenden Standard entwickelt hat.

Ein X.509-Zertifikat enthält

- den Namen des Zertifikatsinhabers (*subject distinguished name*). Für Personen ist hierfür das Namensschema gemäß den X.500-Konventionen vorgesehen. Darüber hinaus sind aber auch anders aufgebaute Namen möglich, was vor allem bei für Rechner oder Webseiten ausgestellten Zertifikaten nötig ist.
- den öffentlichen Schlüssel des Inhabers. Dieser enthält auch Informationen über den Algorithmus, der zum Einsatz kommen soll.
- den Namen des Ausstellers (*issuer distinguished name*)
- die Zertifikatsseriennummer
- den Gültigkeitszeitraum, gegeben durch ein Beginn- und ein Verfallsdatum
- die Signatur des Ausstellers. Neben der eigentlichen Signatur werden auch Angaben zum verwendeten Algorithmus gemacht.

Darüber hinaus sind generische Zertifikatserweiterungen (*extensions*) möglich, so daß bei Bedarf zusätzliche Informationen ins Zertifikat aufgenommen werden können. Diese Erweiterungen können als kritisch (*critical*) deklariert werden. Auf diese Weise wird von der Anwendung, die das Zertifikat auswertet, verlangt, das Zertifikat als ungültig zu werten, wenn sie die Erweiterung nicht unterstützt. Der Erweiterungsmechanismus macht den Standard sehr flexibel und hat entscheidend zu seiner Verbreitung beigetragen.

In X.509 selbst sind bereits einige Erweiterungen für häufig auftretende Anforderungen definiert. Beispielsweise kann hier die Schlüsselverwendung eingeschränkt werden (*X509v3 Key Usage*), so daß das Schlüsselpaar nur zum Verschlüsseln, aber nicht zum Signieren eingesetzt werden darf. Auch muß in CA-Zertifikaten eine Erweiterung (*X509v3 Basic Constraints*) entsprechend gesetzt werden.

Revokationslisten Neben den Zertifikaten selbst definiert X.509 auch das Format für Revokationslisten, mit den folgenden Inhalten:

- den Namen des Ausstellers der Liste (*issuer distinguished name*). Eine Revokationsliste gilt immer nur für eine einzige CA.
- das Erstellungsdatum der Liste (*this update*)
- das Erscheinungsdatum der nächsten Liste (*next update*). Revokationslisten werden regelmäßig neu herausgegeben, üblich sind Gültigkeitsdauern von einem Monat. Selbstverständlich wird bei Auftreten einer Revokation sofort außerplanmäßig eine neue Liste erstellt.
- die Liste der ungültigen Zertifikate. Pro Zertifikat ist hier der Zeitpunkt der Revokation und die Seriennummer angegeben (es sind ab Version 2 des Standards auch noch ergänzende Informationen möglich).
- die Signatur des Ausstellers

PKCS

Während sich X.509 als Standardformat für Zertifikate (und damit öffentliche Schlüssel) durchgesetzt hat, ist die Situation für das Transport- und Speicherformat von privaten Schlüsseln (als Datei, wenn keine Chipkarten verwendet werden) weniger klar. Ein weit verbreitetes Format ist das von den RSA Laboratories [PKCS] entwickelte PKCS#12. Dieses wurde als universelles Dateiformat (unabhängig vom Schlüsselalgorithmus und ausgestattet mit unterschiedlichen Sicherheitsstufen) für Schlüssel, Zertifikate und sonstige Geheimnisse entworfen. Solche Dateien werden auch als Softtoken bezeichnet.

PKCS#12 wird von den gängigen Internetanwendungen unterstützt. Üblicherweise wird dabei der geheime Schlüssel zusammen mit dem Zertifikat mit einem symmetrischen Kryptoverfahren verschlüsselt und dann abgespeichert. Der symmetrische Schlüssel wird hierbei aus einem Kennwort abgeleitet, ohne dessen Kenntnis die Daten folglich nicht mehr wiederhergestellt werden können.

Die PKCS-Familie umfaßt neben PKCS#12 (*Personal Information Exchange Syntax Standard*) noch weitere Standards für unterschiedliche Bereiche in PKI-Anwendungen. Hier sind vor allem PKCS#7 (*Cryptographic Message Syntax Standard*) für verschlüsselte und/oder signierte Nachrichten und PKCS#11 (*Cryptographic Token Interface Standard*) für den Zugriff auf Chipkarten zu nennen.

LDAP

Um PKI sinnvoll nutzen zu können, müssen die Zertifikate allen Teilnehmern zur Verfügung gestellt werden. Hierzu werden normalerweise Verzeichnisdienste eingesetzt. Als Alternative zu dem (in der Implementierung) aufwendigen X.500-Dienst hat sich dessen vereinfachte Variante LDAP (*light-weight directory access protocol*) durchgesetzt.

In einem solchen Verzeichnisdienst werden die Zertifikate (und beliebige sonstige Informationen, etwa Telefonnummern und Email-Adressen) der Teilnehmer in einer hierarchischen Struktur abgelegt. Auch Revokationslisten werden üblicherweise über LDAP bekannt gemacht (eine Alternative zur Veröffentlichung von Revokationslisten besteht im Bereitstellen der Möglichkeit, den Status eines Zertifikates online bei der CA abzufragen, wie es im OCSP-Standard vorgesehen ist).

Die Verzeichnishierarchie folgt dem in X.500 definierten hierarchischen Namensschema: Ein vollständiger Teilnehmernamen (*distinguished name*) besteht aus mehreren Namensteilen, meist mindestens dem „Realwelt“-Namen (*common name*), der Organisationszugehörigkeit und einer Länderkennung.

Gemäß X.500 ist der Autor als `cn=Thilo Planz, ou=CDC, o=TU Darmstadt, l=Darmstadt, c=DE` bekannt.

Kapitel 2

Die Trustcenter-Software FlexiTrust

FlexiTrust ist das im Rahmen des FlexiPKI-Projektes an dem Lehrstuhl von Professor Johannes Buchmann an der Technischen Universität Darmstadt entwickelte Softwarepaket zum Betrieb einer Zertifizierungsstelle.

Die Motivation zur Entwicklung einer CA-Software im universitären Rahmen lag vor allem in der Umsetzung der Idee von einer flexiblen Sicherheitsarchitektur, in der sich einzelne Komponenten schnell und mit minimaler Beeinträchtigung des laufenden Betriebs austauschen lassen. Insbesondere soll es ermöglicht werden, die zugrunde liegenden kryptographischen Basisalgorithmen zu wechseln. Im Gegensatz zu kommerziellen Lösungen, die sich meist ausschließlich auf das RSA-Verfahren stützen, könnte man eine derart flexible PKI mit anderen Basisalgorithmen (zum Beispiel der am Fachgebiet intensiv erforschten Elliptischen-Kurven-Kryptographie) auch weiterverwenden, wenn sich RSA als unsicher erweisen sollte.

Die Software besteht aus drei großen Teilen, einer Registrierungsschnittstelle (FlexiTrust RA), dem CA-Kern (FlexiTrust CA) und der in der vorliegenden Arbeit beschriebenen Infrastrukturkomponente (FlexiTrust IS). Alle Komponenten sind in Java implementiert, und verwenden intern zwei weitere vom Lehrstuhl betreute Module, ein ASN.1-Kodierungspaket (Codec) und eine Bibliothek mit kryptographischen Basisalgorithmen (CDC-Provider).

Die genannten Pakete sind auf Anfrage [CDC] bei der Technischen Universität Darmstadt erhältlich.

2.1 FlexiTrust RA

FlexiTrust RA basiert auf den Ergebnissen der Diplomarbeiten von Jens Dambruch [Dam01] und Markus Schuster [Sch01]. Die Software besteht aus einem sehr flexiblen Formularmanager, in dem Eingabemasken sowie Bearbeitungs- und Prüfschritte definiert werden, einem Servlet, mit dessen Hilfe über einen Webbrowser die Formulare ausgefüllt werden, und einem Modul, das abschließend aus den fertig bearbeiteten Formularen Produkte erzeugt, die dann an CA und IS weitergereicht werden.

Auf diese Weise kann die Interaktion mit dem Benutzer über das Bearbeiten von Formularen erfolgen. Das System ist so ausgelegt, daß es sowohl von Administratoren als auch von Endanwendern verwendet werden kann. In die Zuständigkeit von Administratoren fällt etwa die Überprüfung der Identität eines Zertifikatsantragsstellers, einem Endanwender hingegen wird es zum Beispiel ermöglicht, sein eigenes Zertifikat zu widerrufen. Für die verschiedenen Aufgaben sind jeweils eigene Formulare zu definieren, deren Beschreibungen wahlweise in einer Datenbank oder in XML-Dateien abgelegt werden. Die Authentifizierung der Benutzer (inklusive Administratoren) erfolgt über SSL, wobei wiederum von FlexiTrust erzeugte Schlüssel und X.509-Zertifikate zum Einsatz kommen.

FlexiTrust RA kommuniziert mit der CA (und indirekt auch mit IS) über Transferdateien. Diese Dateien ermöglichen den Betrieb der CA auf einem separaten Rechner in einer gesicherten Umgebung. Sie werden in diesem Fall manuell (per Diskette) der CA zugeführt. Ansonsten ist es aber auch möglich die Dateien über ein gemeinsames Verzeichnis zu übergeben. In den Transferdateien enthalten sind alle Informationen, die CA und IS zur Abarbeitung des jeweiligen Vorgangs benötigen, abgelegt als CA-Processables (siehe hierzu die Ausführungen zur CA). Außerdem werden die Transferdateien von der RA signiert und bei Bedarf auch verschlüsselt.

Die Kommunikation mit IS erfolgt in erster Linie indirekt über die an die CA versandten Processables. In ein Processable können beliebige zusätzliche (von der CA nicht ausgewertete) Informationen eingefügt werden. Die CA reicht diese Daten mit dem fertig abgearbeiteten Processable dann einfach an IS durch. Mit diesem Mechanismus kann die RA Steueranweisungen zur weiteren Verarbeitung seitens der IS einstellen.

Für den Fall, daß die Transferdateien auf dem Weg zur CA oder auf dem Rückweg zu IS verloren gehen oder aber ein Vorgang (Processable) die CA nicht mehr verläßt, ist es Aufgabe der IS, diesen Fehler zu bemerken und die Administratoren zu alarmieren. Hierzu existiert ein direkter Kommunikationsweg von RA zu IS, mit dessen Hilfe der IS mitgeteilt wird, daß ein Processable an die CA exportiert wurde und folglich demnächst bei IS eintreffen sollte. Diese Benachrichtigung erfolgt über die von RA und IS gemeinsam genutzte Datenbank.

2.2 FlexiTrust CA

FlexiTrust CA implementiert den Zertifizierungskern der Trustcenter-Software und damit den sicherheitskritischsten Teilbereich. Hier werden die CA-Schlüssel erzeugt und verwahrt und mit deren Hilfe Zertifikate ausgestellt. Die CA bearbeitet Anfragen seitens der RA und übergibt ihre Produkte (vor allem Zertifikate und Schlüssel) an die IS. Die Kommunikation mit der CA geschieht über signierte Transferdateien, die idealerweise per Diskette von und zu der auf einem abgeschotteten System installierten CA transportiert werden.

FlexiTrust CA ist in ihrer aktuellen Version die Weiterentwicklung der Diplomarbeit von Alexander Wiesmaier [Wie01]. In dieser Arbeit wird auch ein

Konzept zur dynamisch skalierbaren Verteilung von Applikationen auf mehrere Rechner (die *Advanced Processing Infrastructure*) vorgestellt, welches als Basis für die Implementierung von FlexiTrust CA dient.

In dieser Architektur werden Aufgaben oder Vorgänge durch sogenannte Processables abgebildet, die selbst um die Einzelheiten ihrer Bearbeitungsschritte wissen und auch für eine eventuelle Fehlerbehandlung verantwortlich sind. Die zur Bearbeitung der Processables notwendigen Ressourcen werden als Processors bezeichnet. Das System sorgt dafür, daß die Processables den jeweiligen Processors zugeführt werden. Skalierbarkeit wird dadurch erreicht, daß bei Bedarf weitere Processors eingerichtet werden können, unter Umständen auch auf anderen Rechnern (die Kommunikation erfolgt dabei mit den von Java angebotenen Techniken RMI und Jini).

Je nach Anwendung können ganz unterschiedliche Arten von Processables und Processors existieren. Im Rahmen von FlexiTrust kommen beispielsweise Processables zum Erzeugen von Zertifikaten oder zur Chipkartenpersonalisierung und Processors zum Einlesen der signierten Transferdateien oder zum Ansprechen des Chipkartenlesers zum Einsatz.

2.3 FlexiTrust IS

Gegenstand dieser Diplomarbeit war der Entwurf und die Implementierung von FlexiTrust IS, der Infrastrukturkomponente der Trustcenter-Software. Die Hauptaufgabe von FlexiTrust IS ist die Entgegennahme der CA-Produkte, ihre Weiterverarbeitung, Archivierung und Veröffentlichung sowie das Entdecken und die Behandlung eventueller Fehler. Dazu kommen einige Nachsorgetätigkeiten, wie das Anstoßen der RA zur automatischen Zertifikatserneuerung und die Unterstützung von externen Anwendungen beim Zugriff auf die archivierten CA-Produkte.

Im Zentrum der IS stehen sogenannte Produkt-Datensätze. Von der CA eingehende Processables werden (nach Prüfung der Signatur der Transferdatei) zunächst in solche Produkt-Datensätze transformiert. Im Ergebnis enthält dann der Datensatz alle Berechnungsergebnisse des Processables, also beispielsweise Zertifikate, Schlüsselpaare, PINs oder Revokationslisten. Die Produkt-Datensätze werden in der Datenbank abgelegt und von dort vor jedem Bearbeitungsschritt neu geladen (und nach einem erfolgreichen Bearbeitungsschritt zurückgespeichert).

Jedem Produkt wird ein Produktprofil zugeordnet, in dem durch eine Abfolge von Bearbeitungsschritten beschrieben ist, was mit dem Produkt zu geschehen hat. Diese Produktprofile sind in XML-Dateien abgelegt, wodurch eine flexible Anpassung an die spezifischen Anforderungen der jeweiligen Installation ermöglicht wird. Die Zuordnung von Produktprofilen zu Processables kann anhand des Processable-Typs oder (besser) durch Vorgaben seitens der RA geschehen.

Die in den Profilen genannten Bearbeitungsschritte werden von sogenannten Operatoren implementiert. IS enthält ein Framework für die Erstellung und Einbindung eigener Operatoren sowie eine Reihe von Standardoperatoren

für gängige Aufgaben. Operatoren, die gemäß diesen Vorgaben implementiert wurden, können von der IS automatisch geladen und auf anliegende Produkt-Datensätze angewandt werden.

Für den Benutzer stellt sich IS als Hintergrundprozeß dar, der ein Eingangsverzeichnis für CA-Transferdateien überwacht und die in der Datenbank abgelegten Produkte gemäß der angegebenen Profile bearbeitet. IS kann die CA-Produkte in den PKI-Standardformaten in Dateien, in der Datenbank, per Email oder über das LDAP veröffentlichen, so wie es durch die Produktprofile festgelegt ist. Eine Benutzerinteraktion ist nur im Fehlerfall notwendig. Bei Auftreten eines Fehlers alarmiert IS den zuständigen Administrator (zum Beispiel per Email), der dann mit entsprechenden Tools das Problem beheben kann.

2.4 PKI-Projekte mit FlexiTrust

Während der Entstehungszeit dieser Arbeit sind eine ganze Reihe von Interessenten an das Fachgebiet (und die mit der Vermarktung von FlexiTrust beauftragte Firma FlexSecure) herangetreten, die die im Entstehen begriffene Software in ihren Projekten einsetzen wollen. Dies wurde von Fachbereich aktiv unterstützt, um Einblick in die Bedürfnisse der tatsächlichen Anwender (*use cases*) zu gewinnen. Dazu kamen noch universitätsinterne Projekte.

Im folgenden werden die wichtigsten dieser Projekte kurz vorgestellt, wobei es durchaus noch weitere mehr oder weniger konkrete Kooperation gab. Zum Zeitpunkt der Abgabe dieser Arbeit waren die unten aufgeführten Projekte alle noch nicht abgeschlossen und teilweise noch in sehr frühen Stadien.

2.4.1 LiDIA-CA

Schon seit längerem unterhält das Fachgebiet zu Demonstrationszwecken eine eigene Zertifizierungsinstanz. Diese wurde bisher mit einer Vorgängerversion der FlexiTrust-Software betrieben, die durch die neue Version abgelöst werden soll. Es handelt sich hierbei um eine Online-CA, die ohne wirkliche RA und ohne Überprüfung von Personalien die Zertifizierungsanträge direkt von einer Webschnittstelle übernimmt und das Schlüsselpaar per Email versendet.

2.4.2 Badenia Bausparkasse

Der erste größere externe Projektpartner wurde die Badenia Bausparkasse, die eine Public-Key-Infrastruktur für ihre Mitarbeiter aufbauen will. Die Kernanwendung hier ist die Absicherung des Zugriffs auf das Firmennetz durch Außendienstmitarbeiter. Durch den Aufbau von authentisierten und verschlüsselten Verbindungen soll ein *virtual private network* entstehen. Hierbei kommt eine kommerzielle Firewall-Lösung zum Einsatz, die beim Verbindungsaufbau die Gültigkeit der Teilnehmerzertifikate (unter Verwendung des LDAP-Dienstes) sicherstellt. Bei diesem Projekt konnten Erfahrungen mit der Anpassung an vorgegebene LDAP-Schemata und bei der Erstellung der speziellen Firewall-Zertifikate gemacht werden.

2.4.3 Rechnerbetriebsgruppe des Fachbereichs

Als größeres universitätsinternes Projekt steht die Verbreitung von Zertifikaten unter den Studenten des Fachbereichs Informatik an. Jeder bei der Rechnerbetriebsgruppe registrierte Student soll automatisch ein Schlüsselpaar überreicht bekommen und dies z.B. für Emailverschlüsselung verwenden können. Der Hauptzweck besteht hierbei in der Bekanntmachung der Technologie und der Vermittlung eines Sicherheitsbewußtseins. Allein schon aus Kostengründen sollen Softtoken statt Chipkarten eingesetzt werden (allerdings wird eine sehr ansprechende Technologie zum Drucken sicherer PIN-Briefe herangezogen). In einer zweiten Ausbaustufe sollen dann auch Mitarbeiter mit Zertifikaten versehen werden.

2.4.4 Universität Gießen

Ein weitreichenderes Projekt findet in Kooperation mit der Universität Gießen (und weiteren dort angeschlossenen Universitäten) statt, nämlich die Einführung digitaler Studentenausweise für möglichst alle Studenten. Der herkömmliche Ausweis aus Papier soll durch eine Chipkarte ersetzt werden, die mit zahlreichen Funktionen (u.a. Bezahlvorgänge, Emailsignaturen, Zugangskontrollen) versehen ist. Die wichtigste Aufgabenstellung hierbei stellt sich in der Anpassung an die Abläufe des Rechenzentrums und durch die hohe Anzahl der Chipkarten. Da im universitären Umfeld zudem eine im Vergleich zur Situation in Unternehmen wesentlich heterogenere Landschaft von Endanwender-Systemen existiert, darf auch mit zahlreichen praktischen Erfahrungen bezüglich Interoperabilität von PKI-Anwendungen gerechnet werden.

2.4.5 TÜV Süddeutschland

Zum wichtigsten Projekt wurde die Kooperation mit dem TÜV Süddeutschland, der eine einbruchssichere Box zum Betrieb eines Trustcenters entwickelt hat und diese als Komplettlösung zusammen mit einer Trustcentersoftware vermarkten möchte (bisher hatte sich der TÜV vor allem auf die Entwicklung der Hardware konzentriert, wobei die Mitarbeiter auf Erfahrung bei der Konstruktion von Kernkraftwerken zurückgreifen konnten). Die wesentliche Herausforderung an FlexiTrust besteht in der geforderten Langzeitstabilität: die Software muß monatelang ohne Eingriffe von außen (die ein Öffnen der Box nötig machen würden) funktionieren. Gespannt sehen die Entwickler denn auch dem vorgeschriebenen 1000-Stunden-Test entgegen. Darüber hinaus ist die Hardware der TÜV-Box für einen hohen Durchsatz vorgesehen, so daß die eingesetzte Software hier nicht zum limitierenden Faktor werden sollte.

Kapitel 3

Entwurf der Infrastruktur-Komponente

3.1 Anforderungen an die Software

3.1.1 Ist-Zustand

Zu Beginn dieser Arbeit bestand die FlexiTrust-Software lediglich aus den beiden Komponenten CA und RA, im wesentlichen auf dem Entwicklungsstand, der in den jeweiligen Diplomarbeiten beschrieben wird [Wie01, Sch01, Dam01]. Zum Extrahieren der CA-Produkte aus den proprietären CA-Transferfiles existierte lediglich ein sehr rudimentäres Tool, das keinerlei Mechanismen zur automatischen (geschweige denn flexiblen) Weiterverarbeitung besaß. Insofern war die Software nicht direkt in einem Trustcenter einsetzbar.

Am Fachbereich im Wirkbetrieb (zum Erstellen von Zertifikaten für Mitarbeiter und interessierte Studenten) war die sogenannte LiDIA-Online-CA, die damals noch nicht die hier beschriebene FlexiTrust-Software, sondern die von Markus Tak entwickelte Vorgängerversion [Tak99] verwendete. Zum Erzeugen und Ausliefern der Schlüsselpaare und Zertifikate waren zahlreiche manuelle Bearbeitungsschritte notwendig.

3.1.2 Soll-Zustand

Aufgabe dieser Diplomarbeit (sowie anderer Weiterentwicklungen im Bereich RA und CA) war der Ausbau der FlexiTrust-Software zu einer einsatzfähigen Trustcenter-Lösung, die auch außerhalb des Fachbereichs in externen PKI-Projekten Verwendung finden kann.

Die neu hinzukommende Komponente IS sollte hierbei vor allem durch automatisierte Vorgänge manuelle Arbeitsschritte abbauen, soweit diese nicht bereits durch das Zusammenspiel von RA und CA eliminiert wurden. Insbesondere zielt IS damit also auf die Entgegennahme und Veröffentlichung der CA-Produkte.

Diese automatisierten Arbeitsschritte sollten „ab Werk“ bereits die üblichen Mechanismen der Veröffentlichung (Export in PKI-Standardformate, Email-Versand, LDAP, Bereitstellung zum Download) umfassen, aber sowohl erweiter-

bar als auch flexibel konfigurierbar sein, um eine Integration in bereits existierende IT-Landschaften und Abläufe der Anwender zu ermöglichen.

In erster Linie sollte IS ein fester Bestandteil der FlexiTrust-Gesamtlösung sein und als solcher eng mit CA und RA zusammenarbeiten. Insbesondere sollte es der RA ermöglicht werden, Vorgaben bezüglich der Verarbeitung der von ihr angeforderten Produkte zu machen. Auch war eine gemeinsame Administrationsoberfläche für alle drei Komponenten in Planung. Trotzdem sollte die Struktur von IS nicht grundlegend von Interna der anderen Programmteile abhängen, so daß eine spätere Anpassung an andere CA/RA-Implementierungen mit geringem Aufwand möglich sein würde.

3.1.3 Anwendungsszenarien

Um die Anforderungen zu konkretisieren, werden im folgenden Abschnitt einige wichtige Anwendungsszenarien geschildert, aus denen hervorgeht wie das System in den jeweiligen Situationen reagieren soll.

Auslieferung einer Chipkarte

Nachdem über RA und CA eine personalisierte Chipkarte beantragt und erzeugt worden ist, müssen von der IS die zur Auslieferung benötigten Daten bereit gestellt werden. Von der CA erhält die IS ein Processable-Objekt, in dem das Zertifikat sowie Informationen über die Karte enthalten sind.

Die Schritte, die nach dem Einlesen des Processables erfolgen, sind:

- Erzeugen des PIN-Briefes, in dem dem Kartenbesitzer die Zugangs-PIN zu seiner Karte mitgeteilt wird, falls die PIN von der CA gewählt wurde. Alternativ kann der Besitzer die PIN vorab selbst gewählt haben, so daß sie nicht herausgegeben wird. Das gleiche gilt für das Revokationspaßwort für das Zertifikat. Aber selbst wenn beide Informationen nicht im Brief erscheinen, ist es sinnvoll ihn zu erstellen, da hiermit auch die Zuordnung der Chipkartennummer zum Besitzer festgehalten wird.
- Warten auf die Empfangsbestätigung durch den Besitzer. Bevor das Zertifikat veröffentlicht wird, sollte der Besitzer der Karte den Empfang durch geeignete Mechanismen (z.B. durch Abruf einer SSL-geschützten Seite) bestätigen. Wenn dies nach einer gewissen Zeit nicht erfolgt, sollte ein Administrator benachrichtigt werden, der das Zertifikat unter Umständen revozieren wird, was auch automatisch erfolgen kann. Je nach Anwendungsbereich kann auf die Bestätigung auch verzichtet werden.
- Löschen der nicht mehr benötigten geheimen Informationen. Nachdem dem Benutzer Revokationspaßwort und Chipkarten-PIN zugestellt worden sind, sollten diese Daten aus Sicherheitsgründen gelöscht werden. Zum Überprüfen des Revokationspaßwortes muß nur dessen Hashwert aufgehoben werden, die PIN ist ohne Karte wertlos und sollte ohnehin vom Benutzer nach Erhalt geändert werden (es kann jedoch sinnvoll sein, eine eventuelle Administrator-PIN oder PUK aufzubewahren).

- Veröffentlichung des Zertifikats. Dies kann im Verzeichnisdienst LDAP und zusätzlich über die Datenbank oder Dateien geschehen.

Auslieferung eines Softtokens

Eine (weniger sichere) Alternative zu Chipkarten ist der Transport des privaten Schlüssels mittels sogenannter Softtoken. Dies sind verschlüsselte Dateien, die nur durch Eingabe eines Kennworts geöffnet werden können. Der dominierende Standard hierfür ist PKCS#12. Der Ablauf gleicht demjenigen für eine Chipkartenauslieferung, allerdings soll IS zusätzlich auch die automatische Verteilung der Softtoken leisten.

- Erzeugen des PIN-Briefes (wie oben, statt der Karten-PIN gibt es jetzt das Kennwort für die PKCS#12-Datei).
- Zustellung des Softtokens. Dies kann durch Herausschreiben in eine Datei oder durch Emailversand erfolgen.
- Warten auf die Empfangsbestätigung (wie oben).
- Löschen von geheimen Informationen. Revokationspaßwort und PKCS#12-Kennwort können wie oben erläutert gelöscht werden. Wenn keine Schlüsselverwahrung (zum späteren Wiederherstellen) erwünscht ist, muß auch das Softtoken gelöscht werden. Anmerkung: Idealerweise sollte eine Schlüsselverwahrung nur im CA-Kern erfolgen, und die IS sollte nicht auf Dauer sicherheitskritische Daten speichern. Für den Fall, daß auch die IS in einer gesicherten Umgebung betrieben werden kann und die entsprechende Funktionalität im CA-Kern nicht vorhanden ist, soll IS auch Möglichkeiten zur Schlüsselverwahrung haben. Bei Signierschlüsseln sollte jedoch keine Schlüsselverwahrung durchgeführt werden.
- Veröffentlichung des Zertifikats (wie oben).

Revokationslistenmanagement

- Veröffentlichung der Revokationsliste.
- Aktualisieren des Verzeichnisdienstes. Wenn dies gewünscht wird, soll IS die durch eine Revokationsliste ungültig gewordenen Zertifikate aus dem LDAP entfernen.
- automatische Erneuerung der Revokationsliste rechtzeitig vor Ablauf ihrer Gültigkeit. Die IS soll dafür sorgen, daß stets eine gültige Revokationsliste existiert. Dies ist bei regelmäßigen Revokationen automatisch der Fall (weil dann ständig aktualisierte Listen erzeugt werden) aber im Normalbetrieb muß die unveränderte Liste mit verlängerter Gültigkeit neu signiert werden. Hierzu soll die IS über eine Schnittstelle der RA bei Bedarf die Erstellung einer neuen Revokationsliste beantragen. Dies ist eine der wenigen Stellen, an denen IS direkt mit RA kommuniziert. Für den Inhalt der Liste bleiben RA und CA verantwortlich, IS soll den Prozeß lediglich auslösen.

Maßnahmen gegen Ende der Zertifikatslaufzeit

Neben der Auslieferung des Schlüsselpaares fallen unter Umständen weitere Aktivitäten kurz vor Ablauf des im Zertifikat angegebenen Gültigkeitszeitraums an.

- Benachrichtigungen über ablaufende Zertifikate. Es sollen automatisch Benachrichtigungen über demnächst ungültig werdende Zertifikate erstellt werden. Diese sollen den jeweiligen Inhabern per Email zugestellt werden. Darüber hinaus soll es auch zu jedem Zeitpunkt möglich sein, eine Liste der nicht mehr lange gültigen Zertifikate zu erzeugen.
- automatische Zertifikatserneuerung. Die IS soll in der Lage sein, automatisch über die RA Zertifikatserneuerungen anzustoßen. Dies besteht im Ausfüllen der entsprechenden RA-Formulare. Die weitere Verarbeitung (also etwa eine manuelle Prüfung/Korrektur oder die vollständige Automatisierung des Vorgangs) liegt dann bei der RA, gesteuert durch die dort definierten Schritte zur Formularbearbeitung. Aufgabe der IS ist es, im Formular alle zur Rezertifizierung benötigten Informationen einzutragen. Außerdem stellt sie keine Rezertifizierungsanträge für mittlerweile revozierte Zertifikate.
- Entfernen des abgelaufenen Zertifikats aus dem Verzeichnisdienst. Normalerweise soll ein abgelaufenes Zertifikat aus dem LDAP entfernt werden. In bestimmten Anwendungen hingegen soll es dort weiterhin abrufbar sein. Die IS soll beide Fälle handhaben können.

Fehlerhafte Konfigurationen

Die IS ist als langlebiger Hintergrundprozeß konzipiert, der ohne manuelle Eingriffe stabil und produktiv laufen soll. Insbesondere sollen während dem Programmablauf möglichst wenige Fehler auftreten, die ein Eingreifen des Administrators notwendig machen. Hierzu soll die IS mögliche Fehlerquellen weitestgehend durch Selbsttests ihrer Funktionalität beim Anstarten des Systems erkennen, anstatt erst zu einem späteren Zeitpunkt (wenn der Fehler tatsächlich akut wird aber kein Administrator direkt eingreifen kann) den Dienst zu versagen.

Insbesondere also sollen die Konfigurationseinstellungen und das Vorhandensein wichtiger Ressourcen (unter anderem Eingabedateien, Datenbankbindung, LDAP) durch einen Testlauf aller davon abhängenden Funktionen überprüft werden.

Probleme dieses Typs müssen durch entsprechende Korrekturen in den Konfigurationsdateien behoben werden. Unter Umständen wird es dabei notwendig sein (zum Beispiel beim Ausfall externer Dienste), gewisse IS-Teilaufgaben zeitweise zu deaktivieren. IS stellt diese Möglichkeit zur Verfügung.

Probleme mit CA-Processables

Nicht immer verläuft die Bearbeitung eines Processables erfolgreich. Wenn die CA ein Processable nicht vollständig bearbeiten kann (oder die Bearbeitung

aufgrund von CA-internen Validierungen verweigert), gelangt das Processable mit einem Fehlerstatus zur IS. In diesem Fall soll IS einen möglichst umfassenden Fehlerbericht erstellen, aus dem hervorgeht, welcher RA-Antrag betroffen ist und wie das Problem behoben werden kann.

Bei besonders schweren Fehlern kann es auch passieren, daß ein Processable auf dem Weg zur oder in der CA verloren geht. Ein solches Processable gelangt nicht mehr zur IS. Um diese Situation bemerken zu können, überprüft IS eine Liste noch ausstehender Processables (die Liste wird von RA angelegt) und erzeugt bei Überschreiten von einem Zeitlimit einen Fehlerbericht.

Da die genannten Fehler zu jedem Zeitpunkt im laufenden Betrieb auftreten können, sollen die Fehlerberichte den zuständigen Administratoren auch per Email zugestellt werden können, damit diese zeitnah eingreifen können (ohne IS ständig beobachten zu müssen).

Das Problem fehlerhafter oder verlorener CA-Processables kann nicht IS-seitig gelöst werden. In den meisten Fällen wird der RA-Antrag neu gestellt werden müssen. Die Aufgabe der IS hierbei beschränkt sich auf das Erkennen und Bekanntmachen des Problem.

Sonstige Probleme während dem Betrieb

Bei der Abarbeitung der unterschiedlichen Produkte durch die IS können ganz unterschiedliche Fehler auftreten, mit denen auch unterschiedlich verfahren werden muß. Wünschenswert wäre eine automatische Fehlerkorrektur, die aber nur in Einzelfällen möglich sein wird. Manche Fehler (zum Beispiel solche in Folge von Netzwerkproblemen) lassen sich durch einen erneuten Versuch zu einem späteren Zeitpunkt beheben. In den meisten Fällen wird ein Fehler zum Abbruch der Bearbeitung des jeweiligen Produktes führen.

Aufgabe der IS ist es, durch einen Fehlerbericht auf das Problem aufmerksam zu machen, das betroffenen Produkt zu isolieren und mit der Abarbeitung aller anderen Produkte fortzufahren. Ein Verlust von Daten, die der späteren Fehlerbehebung dienen können, ist unbedingt zu vermeiden. Außerdem soll es möglich sein, das Produkt nach einer durch den Administrator durchgeführten „Heilung“ wieder in die IS einzuschleusen.

3.2 Vorbemerkung

Beim Studium der Entwurfsdokumente zu den beiden anderen Teilprojekten CA und RA fällt auf, daß jeweils ein allgemein gehaltenes Framework entwickelt wurde, welches dann in einer zweiten Phase den Bedürfnissen eines PKI-Trustcenters entsprechend mit Inhalt gefüllt wurde. Bei der RA ist das Framework zur Bearbeitung beliebiger Formulare geeignet (es wird sogar das Beispiel einer Pizza-Bestellung angeführt [Sch01, S.10]) und wurde dann mit speziellen Formularen (und Formularkomponenten) versehen, die bei einer Registrierungsstelle benötigt werden. Die CA besteht aus der Advanced Processing Infrastructure (die nachträglich auch als eigenständige Entwicklung von dem FlexiTrust-Projekt getrennt publiziert wurde) und speziellen Typen von Processables und Processors zur Wahrnehmung der CA-Funktionen.

Im Gegensatz hierzu ist IS eine auf den tatsächlichen Einsatzbereich zugeschnittene Lösung. In dieser Arbeit wird kein allgemeineres Vorgehens-, Objekt- oder Programmiermodell entwickelt, als für die Implementierung der Infrastrukturkomponente eines Trustcenters nötig ist. Dies ist nur in geringem Maße dem (in der Tat vorhandenen) Ergebnisdruck bei diesem Projekt geschuldet, sondern vor allem der Erkenntnis, daß sich viele Probleme mit dafür angepaßten Werkzeugen leichter lösen lassen und sich die ohnehin notwendigen Wartungs- und Änderungsarbeiten bei beiden Ansätzen in ihrem Ausmaß nicht erkennbar unterscheiden. Auch framework-gestützte Projekte müssen in Folge der sich ändernden Anforderungen aktualisiert werden und häufig wird man dabei doch mit Problemen konfrontiert, für die im Framework keine Lösung vorgesehen ist.

Trotzdem soll natürlich die Forderung nach einem übersichtlichen, verständlichen, erweiterbaren Konzept erfüllt werden. Dieses Konzept will ich im folgenden darlegen.

3.3 Produkt-Datensätze

Wie bereits im vorigen Kapitel kurz beschrieben, arbeitet die IS mit Produkt-Datensätzen.

Ein solcher Datensatz ist in der Lage, beliebige Produktbestandteile (Komponenten) in sich aufzunehmen, auf die dann mit symbolischen Namen zugegriffen werden kann. Durch diesen flexiblen Mechanismus können die Datensätze ohne die Notwendigkeit einer Anpassung für völlig unterschiedliche Produkte verwendet werden, auch für zum Entwurfszeitpunkt noch gar nicht bekannte Produkttypen.

Neben dieser generischen Speicherfunktionalität, die nach dem Prinzip von Hash-Tabellen funktioniert, bieten Produkt-Datensätze darauf aufbauend für die bisher bekannten Komponenten zugeschnittene Zugriffspfade. Hierzu gehören Funktionen zum Ablegen und Auslesen von Zertifikaten, Softtoken, Revokationslisten, Kundennummern und von Informationen über Bearbeitungsvorgänge.

Produkt-Datensätze werden beim Importvorgang der CA-Processables aus den Transferdateien angelegt. Jedem Produkt wird dabei als eindeutige Kennung eine Produktnummer zugewiesen, die der Nummer des Processables entspricht. Anschließend werden sämtliche Bestandteile des CA-Processables in den Produkt-Datensatz kopiert. Es handelt sich hierbei um die Berechnungsergebnisse der CA (also Zertifikate, Softtoken, PINs und so weiter) sowie um zusätzliche Angaben seitens der RA, die Einfluß auf die weitere Verarbeitung haben können.

Nach dem abgeschlossenen Importvorgang werden die Datensätze in die Datenbank gelegt. Vor jedem der nun folgenden mit der Bearbeitung des Produkts verbundenen Schritte wird der Datensatz neu geladen und nach einem erfolgreichen Bearbeitungsschritt sofort wieder zurückgespeichert. Auf diese Weise wird sichergestellt, daß bei einem Absturz der IS keine Information über Produkte verloren gehen, sondern in einem konsistenten Zustand weiterhin zur Verfügung stehen.

3.4 Produkt-Profile

Die Aufgabe der IS ist die automatisierte Verarbeitung der Produkte gemäß einer vorgegebenen Abfolge von Bearbeitungsschritten. Da nicht für alle Produkte identische Arbeitsschritte zutreffen, lassen sich Produktprofile definieren, in denen jeweils die einzelnen Bearbeitungsschritte festgelegt sind.

Von entscheidender Bedeutung für die Einsatzfähigkeit der IS ist ein hohes Maß an Flexibilität in der Definition der Profile. Es muß den Administratoren möglich sein, eigene Profile zu definieren, in denen beliebige (auch selbst beigesteuerte) Bearbeitungsschritte vorkommen dürfen. Diese Bearbeitungsschritte müssen frei parametrisierbar sein, um beispielsweise Ausgabedateien, Formatvorlagen, Netzwerkadressen und dergleichen anpassen zu können. Außerdem soll es möglich sein, in gewissen Grenzen die Profile nachträglich zu ändern, wobei sich diese Änderungen auf bereits existierende Produkte auswirken sollen.

Um diesen Anforderungen gerecht zu werden, ist folgendes vorgesehen:

- die Produktprofile werden in XML-Dateien definiert, die von den Administratoren leicht geändert werden können. Es können beliebig viele Profile angelegt werden, jedes Profil ist mit einem eindeutigen Namen versehen.
- jedes Profil umfaßt eine Abfolge von Bearbeitungsschritten. Für jeden dieser Schritte (Operationen) wird ein Operator angegeben. Operatoren sind Programme für die Manipulation von Produkt-Datensätzen und werden im nächsten Abschnitt erläutert. Operatoren stellen die für die Bearbeitungsschritte benötigte Funktionalität bereit. Außerdem können für jeden Bearbeitungsschritt beliebig viele benannte Parameter eingestellt werden.
- bei der Erzeugung der Produkt-Datensätze wird jedem Produkt genau ein Profil zugeordnet. Diese Zuordnung sollte durch Vorgaben seitens der RA erfolgen. Hierzu muß das entsprechende RA-Formular an einen bestimmten Profilename gebunden werden, den die RA (über das Processable) an IS übermittelt. Falls dies nicht möglich oder erwünscht ist, gibt es auch einen Ausweichmechanismus, der anhand des Processable-Typs ein Profil auswählt.
- wenn ein Produkt bearbeitet werden soll, kann IS anhand der im Produkt vorhandenen Informationen den jeweils nächsten Bearbeitungsschritt berechnen. Hierzu dient das in Abschnitt 3.6 beschriebene Ablaufsteuerungssystem.

3.4.1 Beispiel für eine Profildefinition

Profile werden in XML definiert, wofür Abbildung 3.1 ein Beispiel bietet. Die genaue Beschreibung der Syntax hierfür findet sich als DTD in Anhang C, die Beschreibung der verschiedenen Operatoren und ihrer Parameter ist Gegenstand von Abschnitt 5.5. Hier sollen nur die Ausdrucksmöglichkeiten bei der Profildefinition illustriert werden.

An dieser Stelle hierzu nur zwei Bemerkungen: Erstens kann (wie bei den letzten beiden Operationen geschehen) der Name weggelassen werden, wenn

```

<profile name="X509P12Request">
  <operation type="WriteCertificates" name="write certificate files">
    <param key="outputDirectory">archive</param>
    <param key="encodings">DER,PEM</param>
  </operation>
  <operation type="WritePIN" name="make PIN letter">
    <param key="templateFile">templates/P12Letter.template.txt</param>
  </operation>
  <operation type="MailNotification" name="email softtoken">
    <param key="templateFile">templates/P12Email.template.txt</param>
    <param key="attachToken">true</param>
  </operation>
  <operation type="ConfirmDelivery" name="wait for confirmation"
    timeout-action="revoke" />
  <operation type="DeleteSecretInformation" />
  <operation type="UpdateLDAP" />
</profile>

```

Abbildung 3.1: Beispiel für die Definition eines Produktprofils zur Auslieferung von PKCS#12-Softtoken per Email.

```

<profile name="revoke">
  <operation type="DeleteSecretInformation" />
  <operation type="RevokeX509">
    <param key="reason">softtoken delivery could not be confirmed</param>
  </operation>
  <operation type="MailNotification">
    <param key="templateFile">templates/RevocationInfoForAdmin.txt</param>
  </operation>
</profile>

```

Abbildung 3.2: Beispiel für die Definition eines Produktprofils zur automatischen Zertifikatsrevokation im Fehlerfall

der Typ des Operators im Profil nur einmal vorkommt. Zweitens verlangt die Angabe einer Timeout-Aktion für den Operator `ConfirmDelivery` ein entsprechendes Profil, zum Beispiel das aus Abbildung 3.2.

3.5 Operatoren

Die in den Profilen genannten Bearbeitungsschritte werden von separaten Modulen implementiert, den sogenannten Operatoren. Die Operatoren zur Unterstützung der üblicherweise in einem Trustcenter anfallenden Aufgaben (siehe auch die Anwendungsszenarien in Abschnitt 3.1.3) sind bereits im Lieferumfang von IS vorhanden. Für speziellere Aufgaben können weitere Operatoren entwickelt und integriert werden.

Operatoren arbeiten auf Produktdatensätzen und werden durch die Einstellungen für den jeweiligen Bearbeitungsschritt (welche wiederum aus einem

Profil stammen) parametrisiert. Das Laden, Bereitstellen und Zurückspeichern dieser Daten wird von anderen Teilen der IS (dem IS-Dämon) übernommen, und fällt nicht in den Aufgabenbereich der Operatoren selbst. Diese Trennung ermöglicht die Entwicklung der Operatoren ohne Kenntnis interner IS-Details. Operatoren können sich somit ihrer Kernaufgabe widmen.

Außer dem eigentlichen Bearbeitungsgang sollen Operatoren auch eine Funktion zum Selbsttest und zur Validierung ihrer Konfigurationsparameter bereitstellen. Dies dient dem Entdecken von Fehlern bereits während des Systemstarts und nicht erst im laufenden Betrieb.

Um die genannten Basisfunktionen abzudecken, enthält IS Operatoren für die folgenden Aufgaben:

- Export von X509-Zertifikaten in übliche Dateiformate (DER, PEM)
- Export von X509-Revokationslisten
- Export von Softtoken als PKCS#12-Datei
- Neuverschlüsselung von Softtoken für eine sichere Schlüsselverwahrung
- Erzeugen von PIN-Briefen nach einer veränderbaren Formatvorlage
- Aktualisieren von LDAP-Einträgen gemäß dem vorliegenden Produkt
- Email-Versand von Nachrichten, bei Bedarf mit beigefügtem Zertifikat oder Softtoken
- Warten auf das bevorstehende Ende der Zertifikatslaufzeit
- Warten auf die Bestätigung der Schlüsselübergabe
- Löschen von sicherheitskritischen Produktbestandteilen
- Revokationslistenerneuerung
- Rezertifizierung
- sichere Schlüsselverwahrung

3.6 Ablaufsteuerungssystem für Produkte

Die Verbindung zwischen Produkten, Profilen und Operatoren besteht in dem Ablaufsteuerungssystem der IS, das für ein gegebenes Produkt anhand des ihm zugewiesenen Profils und seiner Vorgeschichte den jeweils anzuwendenden Operator (sowie dessen Parametrisierung) bestimmt.

Obwohl mit der *Advanced Processing Infrastructure* der CA bereits ein Ablaufsteuerungssystem zur Verfügung stand, das man auch für IS hätte nutzen können, verwendet die IS das in dieser Arbeit entworfene System. Dies liegt vor allem an der Notwendigkeit, dynamisch in die Ablaufprofile eingreifen zu können. Das von der IS benutzte System erlaubt erhebliche Eingriffe in die Ablaufprofile. Allerdings wird dies mit einem recht einfachen Aufbau der Profile erkauft, die sich im wesentlichen auf eine Sequenz von Operationen beschränken müssen, ohne Verzweigungen oder Schleifen.

3.6.1 Berechnung der nächsten Operation

Wie bereits geschildert besteht ein Produktprofil aus einer Abfolge von Bearbeitungsschritten (Operationen), die jeweils mit einem (profilweit eindeutigen) Namen versehen sind. Nach erfolgreichem Abschluß einer Operation wird im Produkt-Datensatz ein Bearbeitungsvermerk angelegt, aus dem hervorgeht, daß und wann die Operation durchgeführt wurde.

Um nun die nächste auszuführende Operation zu bestimmen, prüft das System für jeden im Profil vorkommenden Bearbeitungsschritt (anhand seines Names), ob dieser Schritt bereits für das gegebene Produkt durchgeführt wurde. Der erste im Profil angegebene Schritt, der noch nicht ausgeführt wurde, wird als nächstes angestoßen.

3.6.2 Wiedervorlagezeitpunkte und Zeitlimits

Es gibt bestimmte Operationen, die erst nach einem gewissen Zeitpunkt bearbeitet werden können. Ein Beispiel hierfür ist das Warten auf den bevorstehenden Ablauf eines Zertifikate, um dann eine Warnmeldung hierzu zu erzeugen. Andere Operationen können im Rahmen einer Fehlerbehandlung versuchen, zu einem späteren Zeitpunkt erneut ausgeführt zu werden. Um dies zu ermöglichen, können für die Bearbeitungsschritte Wiedervorlagezeitpunkte definiert werden. Diese Zeitpunkte werden auch in den Produkt-Datensätzen in der Datenbank sichtbar gemacht. Der IS-Dämon zieht immer nur solche Produkte zur Bearbeitung heran, deren Wiedervorlagezeitpunkt bereits erreicht ist.

Ein zweiter Termin für Operationen ist das Timeout als Zeitlimit für die erfolgreiche Bearbeitung der Operation. Wenn für einen Bearbeitungsschritt ein solches Zeitlimit existiert, dann wird die Bearbeitung bei Erreichen des Limits als gescheitert erachtet und abgebrochen. IS bietet drei Möglichkeiten an, wie in einem solchen Fall zu verfahren ist: Normalerweise wird eine Fehlermeldung erzeugt und das Produkt zur manuellen Weiterbearbeitung angesteuert. Alternativ kann die Operation auch ignoriert werden, so daß die Bearbeitung mit dem nächsten Schritt fortgesetzt wird. Als dritte Möglichkeit kann das Produkt einem neuen Produktprofil zugewiesen werden, durch welches automatisierte Fehlermaßnahmen definiert werden. Die dritte Variante kann beispielsweise eingesetzt werden, um Zertifikate, deren Empfang nicht bestätigt wurde, zu revozieren.

3.6.3 Möglichkeiten der dynamischen Profiländerung

Der geschilderte Mechanismus erlaubt es, die Profildefinitionen nachträglich zu ändern. Dies ist durch Anpassungen in der in XML vorliegenden Profilbeschreibung ohne besondere Tools leicht möglich. Die Änderungen werden beim nächsten Neustart der IS wirksam.

Die Auswirkungen der wichtigsten Änderungsarten (vor allem auf sich bereits im System befindende Produkte) sind wie folgt:

Änderung der Parametrisierung Eine Anpassung der Parameter für die verschiedenen Operationen wirkt sich auf jedes Produkt aus, für das diese

Operation noch durchgeführt werden muß. Produkte, die diesen Bearbeitungsschritt bereits hinter sich haben, werden davon nicht beeinflusst.

Hinzufügen eines Bearbeitungsschrittes Alle Produkte des jeweiligen Profils, die sich noch in Bearbeitung befinden, durchlaufen jetzt zusätzlich den neuen Bearbeitungsschritt, selbst dann, wenn das Produkt sich in einem eigentlich fortgeschrittenerem Stadium der Bearbeitung befindet. Es ist zu beachten, daß Produkte, die nach dem alten Profil bereits vollständig bearbeitet wurden, aufgrund des fehlenden Wiedervorlagezeitpunktes nicht automatisch erneut zum Nachholen der neuen Operation herangezogen werden. Falls dies erwünscht sein sollte, müssen deren Wiedervorlagezeitpunkte entsprechend angepaßt werden.

Entfernen eines Bearbeitungsschrittes Für Produkte des jeweiligen Profils entfällt der Bearbeitungsschritt. Wenn er bereits ausgeführt wurde, kann er jedoch nicht rückgängig gemacht werden. Ein Problem kann allerdings dadurch entstehen, daß bei Produkten, die unmittelbar vor diesem Schritt standen, der Wiedervorlagezeitpunkt durch den (jetzt obsoleten) Schritt unnötig weit in die Zukunft verschoben wurde. Dies müßte durch ein Rücksetzen (und Neuberechnen) der Wiedervorlagezeitpunkte behoben werden, welches aber nicht automatisch erfolgt.

3.6.4 Bedingte Verarbeitung durch Wechsel des Profils

Das Ablaufsteuerungssystem der IS geht von einer im wesentlichen sequentiell ablaufenden Operationenfolge aus. Entsprechend begrenzt sind die Möglichkeiten für parallele, bedingte und wiederholte Bearbeitung. Insbesondere die bedingte Verarbeitung, also die Möglichkeit, innerhalb eines Profils zu verzweigen, hat jedoch eine gewisse Relevanz, unter anderem im Rahmen automatischer Fehlerbehandlung.

Eine einfache Möglichkeit auf die weitere Verarbeitung Einfluß zu nehmen, besteht darin, dem Produkt ein neues Profil zuzuweisen. Die Bearbeitung des Produkts wird dann entsprechend dem neuen Profil durchgeführt. Die internen Mechanismen der IS zur Fehlerbehandlung machen von dieser Technik Gebrauch: bei Fehlern, die während der Durchführung eines Bearbeitungsschrittes auftreten, beschreibt ein Fehlerprofil, auf welche Weise der Administrator davon benachrichtigt werden soll, und wenn beim Überschreiten von Zeitlimits zu einem Fehlerprofil verzweigt wird, geschieht dies genauso.

Hierdurch wird es weiterhin möglich, Profile zu verketteten. Der Operator `ExecuteProfile` dient genau diesem Zweck und er unterstützt auch die Rückkehr zum ursprünglichen Profil nach Abarbeitung des eingeschobenen Profils. Aber dieser Mechanismus hat Grenzen. Schleifen oder Rekursion lassen sich mit ihm nicht bewirken, da Operationen kein zweites Mal ausgeführt werden, wenn ein Profil zum zweiten Mal eingebunden wird.

3.6.5 Fehlerbehandlung

Wenn während der Bearbeitung der Produkte (oder auch zuvor bei ihrem Import aus den CA-Processables) ein Fehler auftritt, schleust IS das Produkt aus seiner normalen Bearbeitung aus und führt Operationen auf ihm durch, die in einem Fehlerprofil beschrieben sind. Sinnvollerweise sollten diese Operationen den Administrator alarmieren. Eine automatische Fehlerbehebung wird aufgrund der zahlreichen unterschiedlichen Fehlerarten (und eventuell auch Produkttypen) kaum möglich sein.

Außerdem können Timeout-Profilen definiert werden, die ausgeführt werden, wenn eine Operation auf Dauer nicht erfolgreich durchgeführt werden kann. Diese sind eng an das jeweilige „normale“ Profil gekoppelt, so daß hier durchaus eine spezialisierte Behandlung möglich ist, beispielsweise das automatische Revozieren von Zertifikaten oder Warnmeldungen an den Zertifikatsinhaber.

3.7 Multi-CA-Fähigkeit

IS soll in der Lage sein, für mehrere CAs gleichzeitig tätig zu sein, wobei jede dieser CAs über einen eigenen Satz an Konfigurationsdaten (Profile, Bearbeitungsschritte) verfügen kann. Damit ist nicht unbedingt gemeint, daß tatsächlich auch mehrere Instanzen von FlexiTrust CA betrieben werden müssen. Vielmehr geht es hier um eine logische Trennung nach unterschiedlichen Zertifikatsausstellern, was etwa beim Hosting von Trustcenterdiensten für Dritte oder bei hierarchisch angeordneten CAs notwendig wird. FlexiTrust CA kann hierzu parallel unter mehreren Identitäten (*Issuer*) auftreten.

Die Multi-CA-Fähigkeit wird dadurch erbracht, daß alle Konfigurationsinformationen in der IS nach CAs getrennt gruppiert werden, jedem Produktdatensatz eine CA zugeordnet wird und alle zugehörigen Dateien (insbesondere die XML-Datei mit den Profilen) in jeweils eigene Verzeichnisse für die CAs abgelegt werden. CAs werden anhand des Ausstellernamens (*Issuer DN*) in den von ihnen signierten Zertifikaten unterschieden.

3.8 Datenbank

Die IS ist eine datenbankgestützte Anwendung. Bis auf die Konfigurationsdateien erfolgt die gesamte Datenhaltung über die Datenbank, die auch zur Kommunikation mit der RA verwendet wird. IS benutzt dabei die folgenden Tabellen siehe auch Anhang A):

- in `isproduct` werden die Produkt-Datensätze gespeichert. Um dem IS-Dämon einen gezielten Zugriff zu ermöglichen, werden die Produktnummer, der Profilname, sowie Angaben zum nächsten Bearbeitungsschritt (Operatortyp, Wiedervorlagezeitpunkt, Zeitlimit) als indizierbare Attribute herausgestellt.
- in `isnotification` weist die RA auf gestartete Processables hin und legt eine zeitliche Vorgabe für den Import durch die IS fest.

- zum erleichterten Zugriff auf Revokationslisten und Zertifikate werden diese zusätzlich zu ihrer Speicherung in den Produkt-Datensätzen in den weiteren Tabellen `x509cert` und `x509cr1` abgelegt. Der Zugriff wird durch indizierbare Attribute (wie Ausstellername, Inhabername, Gültigkeitsdaten, Zertifikatsnummer) unterstützt und kommt sowohl internen IS-Anfragen, als auch externen Anwendungen mit Zugriff auf die Datenbank (beispielsweise der RA) zugute.
- über `ranotification` kann die IS Formulare in die RA einspielen, um beispielsweise eine Rezertifizierung anzustoßen. Die Angaben für das Formular sind in einem als Zeichenkette abgelegten XML-Dokument enthalten.

Der Zugriff auf die Datenbank erfolgt mit den allgemein akzeptierten Standards SQL und JDBC, so daß keine Abhängigkeit der IS von einem bestimmten Datenbanksystem entsteht. Durch Kapselung aller datenbank-bezogenen Funktionen in einem einzelnen Zugriffsmodul wird erreicht, daß sich spätere Anpassungen des Datenbankschemas nur auf einen eng umrissenen Bereich der IS auswirken.

Ein Problem stellt die verwendete Speichertechnik dar, mit der die Objekte in die Datenbank gelegt werden sollen. Ohne implementierungstechnischen Fragestellungen allzusehr vorwegzugreifen, sei hier auf den Konflikt zwischen der (von seiten der Programmiersprache) bequemen Möglichkeit der Objektserialisierung, die es erlaubt beliebig komplexe Objekte als Binärdateien abzulegen, und der Notwendigkeit, gewisse Inhalte der Objekte dem Datenbanksystem zur Anfrageunterstützung (und vielleicht auch anderen Anwendungen) explizit zur Verfügung zu stellen, hingewiesen.

Dieser Konflikt wird dadurch gelöst, daß diejenigen Attribute der in der Datenbank gespeicherten Objekte, die in der IS und darüberhinaus für andere Anwendungen sinnvoll in Datenbankanfragen auftauchen können, aus dem serialisierten Objekt herausgezogen und in zusätzlichen Tabellenspalten abgelegt werden. Beispiele für solche Attribute sind oben bereits genannt. Außerdem wird bei der Objektserialisierung auf möglichst universelle Standards zurückgegriffen werden. Dies bedeutet insbesondere die Verwendung der in den PKI-Standards vorgesehenen Kodierungen für Zertifikate und Revokationslisten anstelle von einfacher Objektserialisierung durch die Programmiersprache.

3.9 IS-Dämon

In Aktion tritt die IS als Hintergrundprozeß, als IS-Dämon, der ständig auf Arbeit wartet. Hierzu überwacht der IS-Dämon ein Eingangsverzeichnis nach CA-Transferdateien und liest diese dann ein. Wenn ein Fehler beim Einlesen der Datei auftritt (zum Beispiel wenn die Signatur nicht korrekt ist), wird sie in ein Fehlerverzeichnis kopiert. Im Normalfall aber wird sie (nachdem ihr Inhalt auf die entsprechenden Produktdatensätze verteilt wurde) gelöscht. Außerdem prüft er regelmäßig durch Anfragen an die Datenbank, ob es Produkte gibt, auf denen Bearbeitungsschritte durchzuführen sind.

Der IS-Dämon kann von der Kommandozeile gestartet und gestoppt werden, außerdem ist er in das von FlexiTrust verwendete zentrale Managementframework eingebunden und kann darüber kontrolliert werden. Auch für das Anlegen von Logdateien wird das Managementmodul benutzt, was die Auswertung der Logmeldungen vereinfacht.

Kapitel 4

Implementierung

Die Implementierung der IS erfolgte in der Programmiersprache Java von Sun Microsystems. Dies war schon durch die Notwendigkeit der Zusammenarbeit mit den ebenfalls in Java gehaltenen Komponenten CA und RA vorgegeben, aber aufgrund der diversen Vorteile dieser Sprache (Plattformunabhängigkeit, Objektorientierung, weite Verbreitung, breite Unterstützung mit Bibliotheken) wäre die Wahl auch so auf Java gefallen.

Die Dokumentation der Implementierung besteht aus drei Teilen:

- einen Überblick über die Implementierung stellt dieses Kapitel der Diplomarbeit dar. Hier werden die einzelnen Teile, ihr Zusammenspiel, sowie die bei der Programmierung getroffenen Entscheidungen erläutert. Wenn bekannte Entwurfsmuster [GHJV96] zum Einsatz kamen, sind diese jeweils angegeben.
- In Java integriert ist eine Dokumentationsschnittstelle für Klassenbibliotheken namens JavaDoc. Hiermit lassen sich Java-Klassen für die Verwendung in anderen Programmen dokumentieren. Die von JavaDoc erzeugten (sehr ansehnlichen) HTML-Dateien dienen damit im wesentlichen der Weiterentwicklung von FlexiTrust.
- JavaDoc beschreibt die Schnittstelle einer Klasse wie eine Black Box. Interne Details der Programmierung werden nicht sichtbar gemacht. Hierzu kann man auf den Quellcode und die dort enthaltenen Kommentare zurückgreifen.

4.1 Übersicht

Die IS besteht aus vier Java-Klassenpaketen: Die Klassen für die Basisfunktionalität des IS-Dämons sind im Paket `de.tud.cdc.flexiTrust.is` enthalten (Abbildung 4.1), im Unterpaket `de.tud.cdc.flexiTrust.is.operators` sind die vom IS-Dämon verwendeten Operatoren zusammengefaßt (Abbildung 4.2).

Klassen des IS-Dämons (de.tud.cdc.flexiTrust.is.*)

CA	Behälterklasse für alle CA-bezogenen Informationen
CAImporter	implementiert das Einlesen von CA-Processables aus Transferdateien
DB	implementiert alle Datenbankzugriffsmethoden
IS	Hauptprogramm des IS-Dämons
Logger	implementiert die Logging-Funktionen (benutzt gemeinsames FlexiTrust Log-Modul)
Operator	abstrakte Basisklasse für alle IS-Operatoren
Product	entspricht einem Produktdatensatz
TransferFile	Funktionalität zum Einlesen und Verifizieren der CA-Transferdateien
UnsuccessfulTaskException	Exception, die von Operatoren geworfen werden kann

Abbildung 4.1: Übersicht über die Java-Klassen des IS-Dämons

Grundmenge an Operatoren (de.tud.cdc.flexiTrust.is.operators.*)

ConfirmDelivery	wartet auf die Auslieferungsbestätigung für das Schlüsselpaar
DeleteSecretInformation	löscht sicherheitsrelevante Daten aus dem Produktdatensatz
ErrorLogger	erzeugt Fehlerberichte
ExecuteProfile	weist das Produkt einem neuen Profil zu
KeyBackup	erzeugt verschlüsselte Archivdateien mit Benutzerschlüsseln
MailNotification	erzeugt Emailnachrichten
RenewCRL	erzeugt ein RA-Formular zur CRL-Erneuerung
RenewX509	erzeugt ein RA-Formular zur Rezertifizierung
UpdateLDAP	aktualisiert die Einträge im LDAP-Verzeichnisdienst
WaitForExpiration	wartet auf das bevorstehende Auslaufen der Zertifikatsgültigkeit
WriteCertificates	schreibt Zertifikate in Dateien
WriteCRL	schreibt Revokationslisten in Dateien
WritePIN	erzeugt PIN-Briefe
WriteSofttoken	schreibt PKCS#12-Dateien

Abbildung 4.2: Übersicht über die Java-Klassen der Operatoren

Eine wichtige Komponente der IS, die für praktisch alle Textausgaben benötigt wird, ist der Template-Mechanismus, der es erlaubt aus einer mit Ersetzungsstellen versehenen Formatvorlage und einem dazu passenden Datensatz parametrisierte Ausgaben zu erzeugen (vergleichbar einem Serienbrief). Die hierzu verwendeten Klassen bilden das Paket `de.tud.cdc.flexiTrust.is.util` (Abbildung 4.3).

Klassen des Template-Mechanismus (`de.tud.cdc.flexiTrust.is.util.*`)

<code>FreeObject</code>	definiert das Interface für die in Templates verwendeten Objekte
<code>FreeJavaObject</code>	Objekt-Adapter für Java-Klassen mit <code>get</code> -Methoden
<code>HashedFreeObject</code>	Implementierung von <code>FreeObject</code> mit Hashtables
<code>Decorator</code>	Basisklasse für „Dekorierer“, die Objekte zur Laufzeit mit neuen Methoden versehen können.
<code>PrettyX509</code>	dekoriert X509-Zertifikatsobjekte mit zusätzlichen Methoden (v.a. zur Ausgabe)
<code>PrettyProduct</code>	dekoriert IS-Produkt Datensätze mit zusätzlichen Methoden (zur Ausgabe)
<code>StringTemplate</code>	erlaubt das Ausgeben von Objekten anhand von Templates

Abbildung 4.3: Übersicht über die Java-Klassen des Template-Mechanismus

Zur Konfiguration der IS (vor allem zur Definition der Produktprofile) ist das Auslesen von XML-Dateien notwendig. Hier kommt das Castor-Framework zum Einsatz, das aus den hierarchisch angeordneten Tags einer XML-Datei einen Baum aus Java-Objekten aufbaut. Die Java-Klassen, die den XML-Tags zugeordnet sind, befinden sich im Paket `de.tud.cdc.flexiTrust.is.profiles` (Abbildung 4.4).

Klassen für den Zugriff auf XML-Konfigurationsdaten (`de.tud.cdc.flexiTrust.is.profiles.*`)

<code>CAInfoContainer</code>	entspricht dem Tag <code><ca></code> , dem Wurzeltag von <code>ca.xml</code>
<code>OperationInfo</code>	entspricht dem Tag <code><operation></code> , das eine Operation beschreibt
<code>ImporterInfo</code>	entspricht dem Tag <code><importer></code> , das den CA-Importer beschreibt
<code>ProfileInfo</code>	entspricht den Tags <code><profile></code> und <code><error-handler></code> , die Profile definieren
<code>PropertySet</code>	entspricht dem Tag <code><param-set></code> , das eine Parametermenge beschreibt

Abbildung 4.4: Übersicht über die Java-Klassen für den XML-Zugriff

4.2 Produkt-Datensätze

Alle Informationen über IS-Produkte, ihre Bestandteile, Historie und Bearbeitungszustände werden in der Klasse `de.tud.cdc.flexiTrust.is.Product` gespeichert. `Product` arbeitet nach dem Prinzip einer Hashtable, in der beliebige

Komponente	Name	Objekt-Typ	Zugriffsmethoden
Zertifikate	X509	byte[] []	verschiedene Methoden getCertificate(s) und setCertificates, die mit java.security.cert.X509Certificate arbeiten
Revokations- passworte	RevocPass	String[]	String getRevocPW() und String getRevocPW(int pos). Ge- setzt werden Revokationspassworte nur in Kombination mit dem Setzen von Zertifikaten.
Softtoken	SoftToken	byte[]	setSoftToken(codec.pkcs12.PFXpdu token, String password)
Revokations- listen	CRL	byte[]	java.security.cert.X509CRL getCRL() void setCRL(java.security.cert.X509CRL)
Tokeninfos	TokenInfo	Hashtable	void setTokenInfo(String key, String value) String getTokenInfo(String key)

Abbildung 4.5: reservierte Namen für übliche Produktkomponenten

Objekte als Produktbestandteile unter frei wählbaren Namen gespeichert werden können (im Hinblick auf die spätere Datenbankspeicherung ist darauf zu achten, daß es sich hierbei nur um Objekte aus dem Standardumfang der Java-Laufzeitumgebung handelt).

Beim Zugriff auf die Produktbestandteile ist folglich eine Namenskonvention einzuhalten, wobei die in Abbildung 4.5 aufgeführten Namen für häufig auftretende Bestandteile reserviert sind. Für diese Bestandteile existieren teilweise auch bequeme Zugriffsmethoden (siehe ebenfalls Abbildung 4.5), die auch die jeweiligen Typkonvertierungen vornehmen, so daß nur für unübliche Komponenten (und wo der Zugriff ohne Konvertierungen auskommt) auf die generischen Zugriffsmethoden zurückgegriffen werden muß.

4.3 Produkt-Profile

Wie die verschiedenen Produkte von der IS behandelt werden wird in Produkt-Profilen beschrieben, die im wesentlichen aus einer Abfolge von Operationen bestehen (siehe auch Abschnitt 3.4). Die Produkt-Profile werden beim Starten des IS-Dämons aus XML-Dateien eingelesen und im Speicher abgelegt.

4.3.1 Zusammenspiel mit Produkten und Operatoren

Jedes Produkt ist (zu einem bestimmten Zeitpunkt) genau einem Produkt-Profil zugeordnet. Dies ist realisiert über einen entsprechenden Eintrag im Produkt-Datensatz. Es ist allerdings möglich, daß ein Produkt im Laufe seiner Lebensdauer unterschiedlichen Profilen zugewiesen wird. Das IS-Ablaufsteuerungssystem benötigt diese Zuordnung um den jeweils nächsten Bearbeitungsschritt zu be-

stimmen, der sich immer als erste Operation ergibt, die im Profil angegeben, aber noch nicht ausgeführt wurde.

Wenn ein Produkt dem IS-Dämon zur Bearbeitung vorgelegt wird, führt dieser im einzelnen folgende Aktionen durch:

- aus dem Produktdatensatz wird der Name des Produkt-Profiles ausgelesen und eine entsprechende Instanz der Klasse `ProfileInfo` aufgesucht, die die Profilbeschreibung enthält. Diese Instanz wurde beim Hochfahren durch den IS-Dämon (gemäß den Einstellungen in `ca.xml`) erzeugt und abgespeichert.
- das Produkt-Profil enthält die Liste der Operationsbeschreibungen (implementiert durch die Klasse `OperationInfo`). Diese Liste wird verglichen mit der im Produktdatensatz gespeicherten Liste der bereits ausgeführten Operationen (identifiziert über ihren profilweit eindeutigen Namen).
- die erste Operation des Profils, die noch fehlt, ist diejenige, die jetzt ausgeführt werden muß.
- zu dieser Operation wird der entsprechende Operator (siehe 4.5) ausgeführt. Der Operator wurde bereits beim Hochfahren des IS-Dämons mit den Einstellungen aus der Operationsbeschreibung parametrisiert und hinterlegt.
- nach erfolgreicher Abarbeitung des Operators wird der Vorgang im Produktdatensatz mit Zeitstempel und optionalen Kommentaren protokolliert.
- falls die Bearbeitung durch einen Fehler abgebrochen wird, erstellt der IS-Dämon eine Fehlermeldung und verzweigt in ein eventuell vorhandenes Fehlerprofil.

4.4 XML

IS verwendet die *extensible markup language* XML als Format für die CA-Konfigurationsdateien `ca.xml` und um Formulare an die RA zu übermitteln. XML hat gegenüber anderen Datenformaten eine Reihe an Vorteilen: die Menschenlesbarkeit erlaubt es, Inhalte mit einem einfachen Editor zu inspizieren und zu verändern, aufgrund der Verfügbarkeit von XML-Parsern und -Generatoren läßt sich XML in eigenen Anwendungen ohne großen Implementierungsaufwand nutzen und schließlich ist XML wesentlich flexibler als andere verfügbare textbasierte Formate (zum Beispiel Java-Property-Dateien).

Das Verarbeiten der XML-Dateien geschieht mittels Castor, einem frei verfügbaren Transformationsframework für Javaklassen. Dabei prüft Castor die Wohlgeformtheit der XML-Dateien und wandelt sie in einen Objektgraphen um, der alle Informationen als Java-Datentypen enthält. Die Klassen, aus denen der Objektgraph aufgebaut ist, sind entweder Standardklassen, gemeinsam mit anderen FlexiTrust-Komponenten genutzte Klassen aus dem Paket `de.tud.cdc.flexiTrust.util.xml` oder spezielle IS-Klassen aus dem Paket

`de.tud.cdc.flexiTrust.is.profiles`. Umgekehrt kann Castor aus einem solchen Objektgraph die entsprechende Repräsentation in XML erzeugen. Man spricht hierbei von *marshalling* und *unmarshalling*. Die Abbildungsvorschrift, die angibt wie Castor die XML-Elemente verarbeiten soll, ist durch die Mapping-Dateien `cainfo-mapping.xml` (für die Konfigurationsdateien) und `forminfo-mapping.xml` (für die RA-Formulare) definiert, die sich im Konfigurationsverzeichnis `etc/xml` befinden.

4.5 Operatoren

Die Funktionalität der IS wird durch Operatoren implementiert, wobei eine Grundmenge an mitgelieferten Operatoren (in Kapitel 3.5 kurz vorgestellt und in Kapitel 5.5 beschrieben) für spezielle Anwendungen ergänzt werden kann. Operatoren werden durch die Operationsbeschreibungen aus den Produkt-Profilen parametrisiert und auf Produkte angewandt.

Damit der IS-Dämon die Operatoren laden und benutzen kann, gibt es eine gemeinsame Schnittstelle, die durch die abstrakte Basisklasse `Operator` definiert wird. Es gibt dort im wesentlichen drei Methoden, zwei zum Initialisieren und Testen und die eigentliche Dienstleistungsmethode:

`init()` Die erste Methode, die der IS-Dämon noch während dem Hochfahren aufruft, initialisiert den Operator. Zu diesem Zeitpunkt besteht noch kein Bezug zu konkreten Parametrisierungen oder gar Produkten. Der Operator soll diese Methode nutzen, um Ressourcen anzufordern, die er auf jeden Fall benötigen wird, die unabhängig von einem bestimmten Produkt einsetzbar sind und deren Initialisierung zeitaufwendig ist. Ein Beispiel hierfür ist ein kryptographisch sicherer Zufallszahlengenerator.

`init(OperationInfo)` Ebenfalls noch während dem Hochfahren des IS-Dämons, und zwar für jedes Operation-XML-Tag, das in den Produktprofilen auftaucht, wird die Methode `init` aufgerufen. Die Methode dient der inhaltlichen Prüfung der Parameter, die in der Operationsbeschreibung angegeben sind. Beispielsweise werden hier Vorlagendateien geladen, testweise Verbindungen zu Email- und LDAP-Servern hergestellt oder Typ- und Vollständigkeitsprüfungen der Parameter durchgeführt.

`process(Product)` Die eigentliche Arbeit erledigt der Operator in der Methode `process`, der das jeweilige Produkt übergeben wird.

Operatoren kann man sich als Werkbänke vorstellen, die nur einmal eingerichtet werden müssen, und dann für die gleichartigen Arbeitsgänge an den Werkstücken (Produkten) benutzt werden können. Insofern entsprechen sie den Prozessoren aus der Advanced Processing Infrastructure der CA (siehe Abschnitt 2.2). Dieses Einrichten der Operatoren geschieht denn auch nur einmal, direkt nach dem Einlesen der Produktprofile, da zu diesem Zeitpunkt bereits alle auftretenden Parametrisierungen bekannt sind. Alle Instanzen von `Operator`-Objekten werden dabei (vollständig konfiguriert) in einer internen Hashtable (identifiziert durch das `OperationInfo`-Objekt, das es beschreibt)

gespeichert und im folgenden nicht mehr neu erzeugt, sondern wiederverwendet. Die entsprechenden Methoden `register` und `getInstance` sind in der Basis-Klasse `Operator` als statische Methoden vorhanden. Die vom konkreten Operator abhängende Initialisierung wird hierbei als Schablonenmethode [GHJV96, Template Method, S. 327] aufgerufen und von den Unterklassen implementiert.

4.5.1 Codebeispiel

Um die internen Strukturen und die Arbeitsweise der IS anhand eines Beispiels deutlich zu machen, sind hier Auszüge aus dem Quellcode des Operators `MailNotification` abgebildet, der sich mit dem Versenden von Email-Nachrichten beschäftigt.

```
public class MailNotification extends Operator {  
  
    private Session session;  
    private StringTemplate tpl;  
    private boolean attachCert;  
    private boolean attachToken;  
    private String host;  
    private int port;  
    private String user;  
    private String pass;
```

Es sind drei Methoden vorgesehen, die ein Operator implementieren kann: die beiden Initialisierungsmethoden `init()` und `init(OperationInfo)` und die Methode `process(Product)`, die die eigentliche Arbeit erledigt. Für den Email-Operator benötigen wir nur die zweite Art der Initialisierungsmethode, in der die Parameter aus der Operationsbeschreibung ausgelesen und geprüft werden.

Initialisierung

```
public void init(OperationInfo o) throws Exception {
```

In dem Objekt `o`, das der Methode übergeben wird, befindet sich die Operationsbeschreibung, die im wesentlichen aus einer Reihe von Parametern besteht. Die Klasse `OperationInfo` bietet Methoden zum Auslesen dieser Parameter, wobei auch Typkonvertierungen und Defaultwerte unterstützt werden.

Als erstes wollen wir die Einstellungen des Email-Kontos zum Versenden finden. Die SMTP-Parameter (wie etwa der SMTP-Servername) sind in einer Parametermenge abgelegt (definiert in `ca.xml`), die normalerweise den Namen `SMTP` hat. Es ist möglich einen abweichenden Namen zu vergeben. Dies muß dem Operator aber auch mitgeteilt werden. Hierzu dient der Parameter `useSMTP`. Der Parameter gibt den Namen der SMTP-Parametermenge an. Er ist optional und wenn er fehlt wird der Standardwert `SMTP` angenommen.

Um einen optionalen Parameter vom Typ `String` mit einem Defaultwert einzulesen, wird die Methode `getParameter(String name, String default)` benutzt:

```
String smtpName = o.getParameter("useSMTP", "SMTP");
```

Nachdem der Name der Parametermenge nun feststeht, können wir versuchen, die Parametermenge zu laden. Parametermengen sind nicht den einzelnen Operationen oder Profilen zugeordnet, sondern zählen zu Attributen der CA. Wir müssen also erst die zuständige CA finden und diese nach der Parametermenge fragen:

```
Properties props = o.getCA().getPropertySet(smtpName);
```

Anschließend können wir die SMTP-Einstellungen den übrigen Einstellungen hinzufügen:

```
o.getProperties().putAll(props);
```

Als nächstens wollen wir prüfen, ob wir anhand der gefundenen Einstellungen tatsächlich eine Verbindung zu einem SMTP-Server herstellen können. Dazu müssen zunächst eine Reihe von (nicht optionalen) Parametern überhaupt vorhanden sein. Um das Vorhandensein eines Parameters zu überprüfen verwenden wir die Methode `checkStringParameter(String name)`. Falls der Parameter fehlt, wird gleich eine Exception geworfen und an den IS-Dämon weitergereicht, der dann den Benutzer informieren kann. An dieser Stelle würde auch bemerkt, wenn die weiter oben angeforderte Parametermenge gar nicht existierte.

```
host = o.checkStringParameter("mailServer");
user = o.checkStringParameter("smtpUserPass");
pass = o.checkStringParameter("smtpUser");
```

Eine weitere Einstellung ist der SMTP-Port. Dies ist keine Zeichenkette, sondern eine Zahl. Ausserdem hat der Parameter eigentlich immer den Wert 25, so daß er der optional ist. Wenn er dennoch eingetragen wurde, soll aber geprüft werden, ob es sich um einen numerischen Wert handelt. Hierfür gibt es die Methode `checkIntParameter (String name, int default)`, die auch einen `int` zurückgibt.

```
port = o.checkIntParameter("smtpPort", 25);
```

Jetzt kann versucht werden, eine Verbindung zum SMTP-Server aufzubauen. Wenn dies gelingt, waren die Einstellungen gültig, ansonsten entsteht eine Exception, die wiederum einfach durchgereicht wird. Auf die Einzelheiten des Verbindungsaufbaus wird hier nicht eingegangen und es sei auf das Java Mail API verwiesen.

```
props = new Properties();
props.put("mail.smtp.host", host);
session = Session.getDefaultInstance(props);

Transport transport = session.getTransport("smtp");
transport.connect(host,port,user,pass);
transport.close();
```

Es gibt auch Methoden für Parameter, die einen booleschen Wert (*true* oder *false*) enthalten. Entsprechend der Konventionen für Java-Properties gilt ein boolescher Parameter nur dann als gesetzt, wenn er vorhanden ist und den Wert `true` enthält (Kleinschreibung ist nicht wichtig). Die Parameter, die angeben ob der Email Zertifikate und Softtoken angehängt werden sollen, sind Beispiele hierfür:

```
attachCert = o.getBooleanParameter("attachCertificate", false);
attachToken = o.getBooleanParameter("attachToken", false);
```

Wie die meisten der Textausgaben der IS wird auch der Inhalt der Email mit dem Template-Mechanismus erzeugt. Hierzu wird eine Formatvorlage benötigt, die sich normalerweise in einer Datei befindet. Der Name der Datei wird in dem Parameter `templateFileName` spezifiziert.

```
String templateFileName = o.checkStringParameter("templateFile");
```

Nachdem der Name feststeht, kann die Vorlagendatei geladen und das Template damit initialisiert werden. Der Template-Mechanismus sieht hierfür die Methode `StringTemplate.getInstance(File)` vor.

```
File templateFile = o.getCA().getFile(templateFileName);
if (! templateFile.canRead()) {
    throw new IOException("template file not found or not readable: "
        +templateFile.getPath());
}
tmpl = StringTemplate.getInstance(templateFile);
}
```

Damit ist die Initialisierung abgeschlossen. Um den tatsächlichen Versand von Emailnachrichten unter Bezug auf das jeweils anliegende Produkt kümmert sich die zweite Methode `process(Product)`.

Leistungserbringung

```
public void process(Product p) throws Exception {
```

Für den Fall, daß an die Email Zertifikate und Softtoken angehängt werden sollen, müssen diese dem Produktdatensatz entnommen werden. Für die meisten üblichen Produktkomponenten gibt es spezielle Methoden, zum Beispiel `getCertificate()`. Das Softtoken hingegen kann nur über die allgemeine Methode `get(String componentName)` erhalten werden und liegt dann als DER-kodiertes Byte-Array vor. Falls Zertifikate oder Softtoken benötigt werden, aber fehlen, wirft der Operator eine Exception.

```
X509Certificate cert = p.getCertificate();
byte[] token = (byte[])p.get("SoftToken");

if (cert == null && attachCert)
    throw new RuntimeException("missing certificates");
if (token == null && attachToken)
    throw new RuntimeException("missing token");
```

Danach wird wieder eine Verbindung zum SMTP-Server aufgebaut, die Nachricht wird erstellt und verschickt und abschließend wird die Verbindung wieder geschlossen. Der Sourcecode hierfür wird nicht weiter kommentiert, da er sich nur mit Einzelheiten der Email-Erstellung befasst, die für ein allgemeines Verständnis der Arbeitsweise von Operatoren nicht hilfreich sind.

```
Transport transport = session.getTransport("smtp");
transport.connect(host,port,user,pass);

MimeMessageSource mail = new MimeMessageSource(tmpl);
PrettyProduct pp = new PrettyProduct(p);
if (cert != null) mail.put("cert", new PrettyX509(cert));
mail.put("product", pp);

MimeMessage msg = mail.getMimeMessage(session);

// attach certificate
if (attachCert){
    DataHandler handler = new DataHandler(new PKIDataSource(cert));
    MimeBodyPart mbp = new MimeBodyPart();
    mbp.setDataHandler(handler);
    mbp.setFileName(cert.getSerialNumber()+". "+certFileExtension);
    ((Multipart)msg.getContent()).addBodyPart(mbp);
}

// attach softtoken
if (attachToken){
    DataHandler handler = new DataHandler(new PKIDataSource(token));
    MimeBodyPart mbp = new MimeBodyPart();
    mbp.setDataHandler(handler);
    mbp.setFileName(cert.getSerialNumber()+". "+p12FileExtension);
    ((Multipart)msg.getContent()).addBodyPart(mbp);
}

transport.send(msg);

transport.close();
}
}
```

4.6 Datenbankbindung

FlexiTrust IS verwendet eine gemeinsame Datenbank mit der RA, über die die beiden Komponenten auch Nachrichten miteinander austauschen. Als Datenbanksystem wurde bisher immer nur MySQL eingesetzt, es sollte aber möglich sein, auch andere Datenbanksysteme anzusprechen.

4.6.1 Zugriffscod

Der Zugriff auf die Datenbank stellt eine Querschnittsfunktion dar, die an verschiedenen Stellen der IS benötigt wird. Trotzdem soll der dazugehörige Programmcode an einem einzigen Ort konzentriert sein, um die Auswirkungen von Änderungen im Datenbankschema einzugrenzen. Daher wird die gesamte Kommunikation mit der Datenbank in einer einzelnen Klasse

`de.tud.cdc.flexiTrust.is.DB` zusammengefasst. Diese Klasse stellt anhand der in `is.ini` festgelegten Einstellungen eine JDBC-Verbindung her, die als Singleton [GHJV96, Singleton, S. 139] wiederverwendet wird. Auf diese Weise müssen nicht ständig neue Verbindungen aufgebaut werden. Auf dieser JDBC-Verbindung aufbauend enthält die Klasse für alle von der IS direkt benötigten Datenbank-Operationen (Abspeichern und Auslesen von Zertifikaten, Revokationslisten und Produkten, sowie Zugriff auf die Tabellen zur Kommunikation mit der RA) entsprechende statische Methoden. Die hierbei benötigten SQL-Abfragen werden bereits beim Hochfahren des Systems als *prepared statements* eingerichtet, so daß Probleme mit der Datenbank frühzeitig erkannt werden und die Anfragen in der Folge auch sehr effizient ausgeführt werden können (zumindest theoretisch, denn in diesem Punkt ist man auf die Unterstützung des JDBC-Treibers angewiesen und der Treiber für MySQL nutzt die Informationen über *prepared statements* leider nicht).

4.6.2 Objektserialisierung

Um die im Hauptspeicher als Java-Objekte vorliegenden Informationen in einer Datenbank ablegen zu können, müssen diese in eine geeignete externe Repräsentation gebracht werden. Wie im Entwurfskapitel bereits angesprochen (siehe Abschnitt 3.8), ist hierbei darauf zu achten, daß Teile des Objekts zur Anfrageunterstützung dem Datenbanksystem sichtbar gemacht werden und das gesamte Objekt in einem möglichst weit verbreiteten Format gespeichert wird. Der für den Programmierer bequemste Weg der in Java integrierten Objektserialisierung (ein Objekt wird dabei automatisch in ein Bytefeld umgewandelt und kann daraus auch wieder erzeugt werden) kann hier nur unter Vorbehalt eingesetzt werden, da eine solche Speicherung nur von Java-Programmen verstanden werden kann, die zusätzlich noch über die gleichen Klassenbibliotheken verfügen müssen, und zwar in identischen (oder zumindest binärkompatiblen) Versionen, was speziell bei selbst entwickelten Paketen, die ständig weiterentwickelt werden, zum Problem werden kann.

Kommunikation mit der RA Die Tabellen zur Kommunikation mit der RA sind textbasiert, so daß hier kein Problem entsteht. Die Tabelle `isnotification`, mit der die IS auf noch ausstehende Processables hingewiesen wird, enthält lediglich Datumsangaben und Produktnummern. Die gegensätzliche Tabelle `ranotification`, in der die IS Formulare an die RA übermittelt, enthält den Inhalt des Formulars als XML-Dokument, also als druckbare Zeichenfolge, sowie ebenfalls Datumsangaben und Produktnummern.

PKI-Produkte Bei Zertifikaten und Revokationslisten liegt die Darstellung als DER-kodiertes Binärobjekt (Blob) auf der Hand. Zur Unterstützung von Suchanfragen werden wesentliche Attribute (IssuerDN, SubjectDN, Kundennummer, SerialNumber, NotBefore, NotAfter, ThisUpdate, NextUpdate) als einzelne Spalten repliziert gespeichert.

Produkt-Datensätze Schwieriger gestaltet sich die Situation bei der Tabelle `isproduct`, in der die Produkt-Datensätze gespeichert werden. Aufgrund der komplexen Struktur der Datensätze, die ja praktisch beliebige Einträge aufnehmen können, erscheint hier eine Speicherung als serialisiertes Java-Objekt angebracht. Da es für diese spezielle Datenstruktur ohnehin kein Standardformat gibt, und sich folglich nur FlexiTrust-eigene Anwendungen dafür interessieren können, ist dies vertretbar. Um eventuelle Probleme mit der Langzeitspeicherung bei sich weiterentwickelnden IS-Klassen zu vermeiden, werden jedoch nicht die IS-Produkte selbst (`de.tud.cdc.flexiTrust.is.Product`) serialisiert, sondern nur ihre als Hashtable gespeicherte Komponenten. Bei der Klasse `java.util.Hashtable` ist davon auszugehen, daß sie auch in allen zukünftigen Java-Versionen noch deserialisiert werden kann. Die Komponenten selbst werden folglich auch nur als Standard-Javaklassen in diese Hashtable gelegt, also zum Beispiel PKI-Produkte nicht als `sun.*`- oder `codec.*`-Objekte, sondern DER-kodiert als Bytefelder.

4.7 Template-Mechanismus

```
Hello ${SubjectPrettyCN},  
  
your distinguished name is ${SubjectDN}.  
Your certificate expires at ${NotAfter}.
```

Abbildung 4.6: Beispiel für ein Template passend zum Code-Beispiel in Abbildung 4.7

Zum Erzeugen von Benutzerausgaben stellt IS einen Template-Mechanismus zur Verfügung (siehe Abschnitt 5.6), durch den es möglich wird, in Formatvorlagen über symbolische Namen auf Java-Objekte zuzugreifen (und sie auszugeben). Er findet sich im Paket `de.tud.cdc.flexiTrust.is.util`.

Bei der Implementierung des Template-Mechanismus sind einige Schwierigkeiten mit Spracheigenschaften von Java aufgetreten. Da die im Template jeweils angeforderten Objekte erst zur Laufzeit aus dem dort angegebenen Namen ermittelt werden können, ist die starke Typisierung von Java-Objekten hinderlich, bei der Methoden- und Attributnamen bereits beim Kompilieren feststehen müssen. Um dieses Problem zu umgehen, nutzt IS die Möglichkeiten der Introspektion und Reflektion (im Paket `java.lang.reflect`) zur Bereitstellung von sogenannten freien Objekten (`FreeObject`), die sich hinreichend flexibel verhalten.

Ein `FreeObject` simuliert eine schwache Typisierung, wie sie in Sprachen wie Perl oder Smalltalk vorkommt. Ein `FreeObject` ist ein Objekt ohne Klassenzugehörigkeit. Es kann zur Laufzeit um Attribute und Methoden erweitert werden. Der Zugriff auf Attribute und Methoden erfolgt über Zeichenketten. Die hierzu benötigten Java-Methoden sind im Interface `FreeObject` festgelegt.

Es gibt drei elementare Typen, die dem Interface `FreeObject` entsprechen:

`HashedFreeObject` ein neu erzeugtes Objekt, ohne Methoden oder Attribute.

Attribute können zur Laufzeit hinzugefügt werden. Sie werden dann intern unter ihrem Namen in einer Hashtable gespeichert (daher der Name). Das Objekt kann nachträglich mit Methoden „dekoriert“ werden.

`FreeJavaObject` ein Objektadapter [GHJV96, Adapter, S.151], der es ermöglicht, normale Javaklassen als `FreeObject` zu verwenden. Es kann dann (lesend) auf alle Attribute zugegriffen werden, für die entsprechende get-Methoden existieren (z.B. wird `getName()` auf ein Attribut `Name` abgebildet). Wenn neue Attribute hinzugefügt werden, überschreiben diese gleichnamige Objektmethoden.

`Decorator` dekoriert [GHJV96, Decorator, S.177] ein Objekt, um ihm zur Laufzeit neue Methoden hinzuzufügen. Wenn es sich dabei um get-Methoden handelt, werden diese auch als Attribut sichtbar gemacht. Die Basisklasse selbst stellt nur die Funktionalität zur Dekoration zur Verfügung und muß im konkreten Fall abgeleitet werden, wie beispielsweise mit `StringTemplate` geschehen (siehe unten).

Auf diesem Objektmodell aufbauend besteht der Template-Mechanismus aus dem Dekorierer `StringTemplate`, der für eine vorlagenbasierte Abwandlung der `toString()`-Methode des Objekts sorgt. Unterstützt wird er dabei von zwei weiteren Dekorierern `PrettyX509` und `PrettyProduct`, die die beiden gebräuchlichen Objekte zur Speicherung von Zertifikaten und IS-Produkten mit nützlichen Funktionen zur Ausgabe von Strings erweitern, beispielsweise den Zugriff auf den `SubjectCN` eines Zertifikats, der aus dem `SubjectDN` extrahiert wird.

Damit die verwendeten Templates, die normalerweise in Dateien gespeichert werden, nur einmal eingelesen (und interpretiert) werden müssen, werden sie zunächst in eine interne Datenstruktur überführt und im Hauptspeicher abgelegt, wo sie wiederverwendet werden können. Deshalb werden Objekte vom Typ `StringTemplate` nicht über einen Konstruktor, sondern durch die Klassenmethode `getInstance()`, die die Speicherung der Vorlagen verwaltet.

Das Code-Beispiel in Abbildung 4.6 und Abbildung 4.6 soll die Arbeitsweise des Template-Mechanismus verdeutlichen, ansonsten sei auf die JavaDoc-Dokumentation hingewiesen.

4.8 FlexiTrust Administrationsframework

In einer Diplomarbeit von Christoph Ender, die zeitgleich mit der vorliegenden Arbeit angefertigt wurde, entstand ein Framework zur Erleichterung und Ver-

```

// Erzeugen eines freien Objektes, dessen Inhalt ausgegeben werden soll
// zum Beispiel ein java.security.cert.X509Certificate cert

FreeObject obj = new FreeJavaObject(cert);

// Hinzufügen eines Attributes mit dem Revokationspaßwort

obj.put("revoc", "12345678");

// Zugriff auf Attribute
String dn = obj.getString("SubjectDN");
    // wird realisiert durch Methodenaufruf cert.getSubjectDN()
String revoc = obj.getString("revoc");
    // realisiert durch Zugriff auf eine interne Hashtable

// Beispiel für einen Zugriff auf nicht-Zeichenketten
BigInteger serial = obj.get("SerialNumber").typecast("java.math.BigInteger")

// Beispiel für einen geschachtelten Zugriff
int bit = obj.getInt("SerialNumber.LowestSetBit")
    // eigentlich cert.getSerialNumber().getLowestSetBit()

// Dekorieren mit PrettyX509, um weitere Ausgabefunktionen zu erhalten
obj = new PrettyX509(obj);

// Dekorieren mit einem StringTemplate
Decorator tpl = StringTemplate.getInstance(
    new java.io.File("template.txt"));
tpl.decorate (obj);

// Ausgabe auf den Bildschirm
System.out.println(tpl.toString());

```

Abbildung 4.7: Code-Beispiel für den Template-Mechanismus

einheitlichung bestimmter administrativer Funktionen für die drei FlexiTrust-Komponenten. Die beiden für IS zu diesem Zeitpunkt relevanten Teile dieses Admin-Moduls sind die Anbindung an die Java-Management-Erweiterung JMX und die Implementierung eines Log-Mechanismus.

4.8.1 JMX

Von Sun Microsystems gibt es ein Framework namens JMX (*Java Management Extension*) zum Management von Java-Komponenten über ein Netzwerk, beispielsweise mit einem HTML-Browser. JMX-taugliche Komponenten melden sich bei einem zentralen Managementmodul an, wobei sie von ihnen angebotene Attribute und Operationen bekannt geben. In der frei verfügbaren Referenzimplementierung, auf die das Admin-Modul aufbaut, ist es leider nicht vorgesehen, ein Managementmodul anzusprechen, welches in einem anderen Prozeß oder gar auf einem anderen Rechner läuft. Diese Funktionalität wird unter Verwendung

von RMI vom Admin-Modul hinzugefügt. Weitere Vorteile des Admin-Moduls sind die Integration in den Servlet-Container der RA und ein Mechanismus zum Anstarten von Ant-Tasks.

Der IS Dämon unterstützt ein geregeltes Anhalten und Herunterfahren sowie Statusabfragen über JMX. Das Verändern von Parametern und Produktprofilen zur Laufzeit ist allerdings nicht vorgesehen. Hierzu muß der IS Dämon weiterhin heruntergefahren, mit neuen Konfigurationsdaten versehen und wieder gestartet werden. Eine wünschenswerte Erweiterung wäre somit die Austauschmöglichkeit der Konfigurationsdateien über JMX (bisher geschieht dies durch normale Dateisystemoperationen).

4.8.2 Logging

Das Admin-Modul stellt auch eine Klasse zum Anlegen von Logs bereit. Die Logeinträge werden nach Logleveln unterschieden, mit Modulkennzeichen und Zeitstempeln versehen und in einem gemeinsamen Format abgelegt, das die spätere Auswertung der Logs ermöglicht.

Der IS-Dämon verwendet diese Logging-Klasse, kapselt sie jedoch noch in einer eigenen Klasse, die zusätzliche Bildschirmausgaben erzeugt und auch der Internationalisierung dient (siehe folgender Abschnitt). Es wird dabei von den Logleveln für Kommentare, Warnungen und Fehler Gebrauch gemacht.

4.9 Internationalisierung

Eine wichtige Eigenschaft einer Software ist die Möglichkeit, sie an verschiedene sprachliche und kulturelle Gegebenheiten anzupassen. Der größte Aufwand bei einer solchen Lokalisierung ist die Anpassung der ausgegebenen Texte, also deren Übersetzung sowie die Anpassung von Datums-, Zeit- und Mengenangaben. Um diese Aufgabe bewältigen zu können, versucht man, die landes- und sprachspezifischen Anteile der Software vom übrigen Teil zu isolieren (Internationalisierung), so daß man diesen Teil später austauschen kann (Lokalisierung) ohne Änderungsbedarf an anderen Stellen zu verursachen.

Die Ausgaben der IS lassen sich in drei Arten unterteilen:

Logeinträge Alle Logeinträge, die der IS-Dämon erzeugt, laufen über die zentrale Logger-Klasse, die sie dann an das FlexiTrust Log-Modul weiterreicht (siehe Abschnitt 4.8.2). Hier ist auch der Ansatzpunkt für die Lokalisierung: Logeinträge werden unter Benutzung des Template-Mechanismus aus einem Formatstring mit variablen Komponenten zusammengesetzt. Die zentrale Logging-Klasse übersetzt den Formatstring und damit die konstanten Bestandteile des Logeintrags durch Nachschlagen in einem Wörterbuch für die jeweilige Sprache (Java bietet hier sogenannte ResourceBundles an). Der variable Teil bleibt unangetastet. Hierbei handelt es sich normalerweise auch um Kennziffern oder Namen, die keiner Übersetzung bedürfen.

menschenlesbare IS-Produkte Wenn die IS PIN-Briefe, Emailnachrichten und dergleichen erzeugt, dann wird hierzu der Template-Mechanismus be-

nutzt. Normalerweise befinden sich alle Textpassagen in der Vorlagenda-
tei, die dann nur noch durch Namen, Nummern und Zeitangaben ergänzt
wird. Eine Sprachanpassung kann also auf einfache Weise in der Vorlage
selbst erfolgen. Anmerkung: Sollen verschiedene Sprachfassungen gleich-
zeitig unterstützt werden (beispielsweise je nach Herkunft des Zertifikats-
inhabers), so sind unterschiedliche Produkt-Profile zu definieren.

Fehlerberichte Wenn im laufenden Betrieb Fehler auftreten, die ein Eingrei-
fen der Administratoren notwendig machen, kann der IS-Dämon Fehler-
berichte erzeugen, aus denen das Problem hervorgeht. Der Inhalt eines
solchen Fehlerberichts wird durch eine Vorlagendatei festgelegt, so daß
der nicht-variable Teil wie oben geschildert in der Vorlagendatei locali-
siert werden kann.

4.10 Benutzte Bibliotheken

IS greift auf eine Reihe von Java-Klassenbibliotheken zurück, die Basisfunktio-
nalität aus verschiedenen Bereichen implementieren. Alle diese Bibliotheken
werden in Form von JAR-Dateien bei der Installation der FlexiTrust-Software
mit installiert.

Java-Plattform IS wurde für die Java2 Standard Edition in der Version des
JDK 1.3 entwickelt.

Ant Während der Entwicklung und auch zum Anstarten der Anwendungen
wird Ant (entwickelt vom Jakarta-Projekt der Apache Software Founda-
tion) eingesetzt.

Kryptographie-Erweiterungen Neben den Möglichkeiten des JDK1.3 be-
nötigt IS die Kryptographie-Erweiterung `javax.crypto` und zusätzliche
kryptographische Algorithmen aus dem CDC-Provider des Fachgebiets.

ASN.1-Codec Zum Umgang mit den ASN.1-Datenstrukturen wird das Codec-
Paket des Fraunhofer-Instituts verwendet, das mittlerweile ebenfalls vom
Fachgebiet gepflegt wird.

LDAP Der LDAP-Zugriff geschieht mittels einer Bibliothek von Netscape und
einer (in IS integrierten) darauf aufbauenden Helferklasse.

Email Der Email-Versand erfolgt mit dem Java-Mail-API, welches wiederum
das JavaBeans-Activation-Framework benötigt. Beide Pakete sind als Teil
der Java2 Enterprise Edition erhältlich und können auch in der Standard
Edition verwendet werden.

Datenbank Als Datenbank kommt MySQL zum Einsatz, so daß ein JDBC-
Treiber hierfür benötigt wird.

XML Die Verarbeitung von XML-Dateien wird Castor überlassen, einem Open-
Source-Paket zum Transformieren von Java-Klassen in (unter anderem)
XML.

JMX IS verwendet die Referenzimplementierung der Java Management Extensions von Sun Microsystems

FlexiTrust Die IS benötigt zum Einlesen von CA-Processables die aktuelle Binärversion von FlexiTrust CA, für das Logging und die JMX-Funktionen das FlexiTrust Admin-Modul und für den Umgang mit XML einige gemeinsam genutzte Klassen aus dem RA-Paket.

Kapitel 5

Betriebsanleitung

5.1 Installation

5.1.1 Verzeichnisstruktur

Es gibt eine Reihe von für den Betrieb der IS wichtigen Verzeichnissen, deren Funktion hier erklärt wird und deren Ort unter Umständen den jeweils vorhandenen Gegebenheiten angepaßt werden muß.

IS-Basisverzeichnis Unterhalb dieses Verzeichnisses liegen die internen IS-Dateien. Außerdem sollte der IS-Dämon aus diesem Verzeichnis heraus gestartet werden. Normalerweise zu finden als `$FLEXITRUST_HOME/is`.

IS-Konfigurationsverzeichnis Im IS-Basisverzeichnis befindet sich ein Unterverzeichnis *etc*, in welchem alle Konfigurationsdateien der IS abgelegt sind, insbesondere das globale Konfigurationsfile `is.ini` und die Konfigurationen der verschiedenen CAs.

CA-Konfigurationsverzeichnisse IS ist in der Lage, für mehr als eine CA gleichzeitig tätig zu sein. Für jede unterstützte CA existiert im IS-Konfigurationsverzeichnis ein Unterverzeichnis, dessen Name unbedeutend ist und lediglich an einigen Stellen als Kurzbezeichner für die CA verwendet wird (Beispiel: `etc/testCA`). In diesem Verzeichnis befinden sich unter anderem das CA-Zertifikat und die Konfigurationsdatei `ca.xml`. Letztere bestimmt sämtliche Einstellungen für diese CA und ist damit die zentrale Konfigurationsdatei für die Arbeitsweise der IS.

Log-Verzeichnis Ausgabeverzeichnis für Log-Dateien des IS-Dämons. Wird in `is.ini` festgelegt, mit Standardvorgabe `isLogs` unterhalb des IS-Basisverzeichnisses.

Fehler-Verzeichnisse Ausgabeverzeichnis für Fehlerberichte des IS-Dämons. Dieses kann für jede konfigurierte CA über deren Konfigurationsdateien `ca.xml` einzeln eingestellt werden, mit `isErrors` unterhalb des IS-Basisverzeichnisses als Voreinstellung.

CA-Importverzeichnisse aus diesen Verzeichnissen werden CA-Transferdateien gelesen. Wenn die IS über ein gemeinsames Dateisystem mit der CA verbunden ist, kann hier direkt das Ausgabeverzeichnis `p70ut` der CA verwendet werden. Transferdateien, die nicht bearbeitet werden können, werden in ein Fehlerverzeichnis verschoben, das ebenfalls anzugeben ist. Beide Verzeichnisse werden über `ca.xml` konfiguriert.

IS-Ausgabeverzeichnisse von der IS erzeugte Dateien können je nach CA und Dateityp in ganz unterschiedliche Verzeichnisse geschrieben werden, die jeweils bei der Konfiguration der Operationen anzugeben sind. Viele Installationen verwenden jedoch ein einziges Ausgabeverzeichnis `is0ut` im IS-Basisverzeichnis.

5.2 Konfiguration

5.2.1 `is.ini`

In der Datei `is.ini` sind einige globale Einstellungen für den IS-Dämon festgehalten.

Datenbankverbindung Angaben, die notwendig sind, um über JDBC die Verbindung zur Datenbank herzustellen (URL, Benutzername, Passwort, eventuell zu registrierende Treiber)

Log-Datei das Verzeichnis, in dem Log-Dateien angelegt werden (relativ zum Arbeitsverzeichnis, das mit dem IS-Basisverzeichnis identisch sein sollte)

Screen-Log-Level es kann ein minimales Log-Level für die Bildschirmausgaben des IS-Dämons eingestellt werden. Auf die eigentliche Log-Datei hat dies keinen Einfluß.

Krypto-Provider zusätzliche JCA-Provider, die eingebunden werden sollen.

5.2.2 `ca.xml`

Für jede CA (nicht unbedingt im Sinne von Installationen der CA-Software, sondern im Sinne einer logischen CA, einem *issuer*) existiert in deren Konfigurationsverzeichnis eine Datei `ca.xml`, in der die Behandlung von Produkten (Processables) dieser CA seitens des IS-Dämons vollständig beschrieben wird. Der Aufbau und Inhalt dieser Datei wird im folgenden geschildert, weiterhin findet sich in Anhang B eine kommentierte Beispieldatei.

Produkt-Profile In den Produkt-Profilen werden die von der IS durchzuführenden Aktionen für einen Produkttyp als Abfolge von Bearbeitungsschritten definiert. Hierzu stehen eine Reihe von mitgelieferten Grundoperatoren zur Verfügung, die durch selbstentwickelte ergänzt werden können.

CA-Importer Ein spezieller Operator sorgt für den Import der CA-Processables aus den Transferdateien. Er muß durch Angabe des Import- und Fehlerzeichnisses konfiguriert werden.

Fehlerbehandlung Sollen zusätzlich zu den in die Log-Dateien geschriebenen Fehlermeldungen Maßnahmen getroffen werden (um zum Beispiel den Administrator zu alarmieren), kann eine Folge von Fehleroperationen angegeben werden, die dann auf jedes Produkt, in dem ein Fehler entsteht, angewandt werden.

Definition von Parameter-Mengen es können Mengen (Sätze) von zusammengehörigen Parametern definiert werden, die dann an anderer Stelle verwendet werden können. Genau wie die Parameter der Operationen werden diese als Zeichenfolgen angegeben und ebenfalls durch eine Zeichenfolge benannt. Parameter-Mengen kommen beispielsweise zur Konfiguration des LDAP-Zugriffs zum Einsatz.

XML-Elemente im einzelnen In `ca.xml` werden die folgenden XML-Elemente (Tags) verwendet (die DTD ist in Anhang C zu finden):

`<ca>` ist das Wurzelement, das den restlichen Inhalt der Datei einrahmt.

`<importer>` beschreibt den zum Import von CA-Transferdateien notwendigen Importer. Das Tag verhält sich ansonsten wie ein `<operation>`-Element.

`<error-handler>` definiert die im Fehlerfall auszuführenden Schritte (zur Benachrichtigung des Administrators), wobei es analog einem `<profile>`-Element eine Folge von `<operation>`-Elementen umfaßt.

`<param-set>` gruppiert eine Menge von `<param>`-Elementen zu einer (durch das Attribut `name`) benannten Parameter-Menge.

`<param>` setzt einen Parameter als Key-Value-Paar. Der Name des Parameters wird durch das Attribut `key` gesetzt, der Wert wird als Zeichenkette im Elementkörper angegeben. Das Element kann innerhalb von `<param-set>`, `<ca>` und `<operation>` vorkommen.

`<use-param-set>` innerhalb von `<operation>` bewirkt die Einbindung sämtlicher im durch das Attribut `key` angegebenen Parameter-Set enthaltenen Parameter.

`<profile>` definiert ein Produktprofil. Jedes Profil benötigt einen eindeutigen Namen (Attribut `name`). Ein Profil besteht aus einer Reihe von `<operation>`-Elementen.

`<operation>` beschreibt einen einzelnen Bearbeitungsschritt. Auch Operationen benötigen einen innerhalb des Profils eindeutigen Namen, der über das Attribut `name` angegeben werden kann und ansonsten aus dem Operationstyp abgeleitet wird. Der Typ der Operation (also die implementierende Java-Klasse) wird mit dem Attribut `type` als voll qualifizierter Klassenname übergeben. Wenn es sich um einen eingebauten IS-Operator handelt, kann auf die Angabe des Paketnames (`de.tud.cdc.flexiTrust.is.operators`) verzichtet werden. Parameter für diese Operation werden als geschachtelte

<param>- und <use-param-set>-Elemente angegeben. Ein Zeitlimit (relativ zum Beginn der Bearbeitung der Operation) kann mit dem Attribut `timeout` angegeben werden, falls bei Erreichen dieses Zeitlimits eine besondere Aktion ausgelöst werden soll, kann ein entsprechender Profilname als `timeout-action` spezifiziert werden.

Konfiguration des LDAP-Zugriffs Zum Zugriff auf das LDAP sind eine größere Menge von Konfigurationsparametern vorzugeben, wie etwa der Servername, der Benutzername, das Kennwort und Informationen über das Schema, nach dem beim Erzeugen von LDAP-Einträgen vorgegangen wird. Der IS-Operator `UpdateLDAP` erwartet diese Einstellungen in einer Parametermenge mit Namen „LDAP“ (falls pro CA mehr als ein LDAP bedient werden muß, kann auch ein abweichender Name spezifiziert werden).

Die LDAP-Parameter in dieser Parametermenge sind

`serverName` der Hostname des LDAP-Servers (oder die IP-Adresse)
`userName` der Benutzername zur Anmeldung beim Zugriff auf das LDAP
`userPass` das zugehörige Passwort
`serverRootDN` das Wurzelverzeichnis in diesem LDAP (z.B. `c=de`)
`serverPort` der Port des LDAP-Servers
`protocolVersion` die verwendete Protokollversion, normalerweise 3
`insertionRule` die Abbildungsvorschrift zwischen SubjectDN und LDAP-Hierarchie (z. B. `o=org,ou=orgUnit,cn=certUser`)
`orgObjectClasses` LDAP-Objektklassen für die Organisationsebene (z.B. `top,organization`)
`orgRequiredAttribs` notwendige Attribute (z.B. `objectclass,o`)
`orgUnitObjectClasses` LDAP-Objektklassen für die Organisationseinheitenebene (z.B. `top,organizationalUnit`)
`orgUnitRequiredAttribs` notwendige Attribute (z.B. `objectclass,ou`)
`certUserObjectClasses` LDAP-Objektklassen für die Personenebene (z.B. `top,person,strongAuthenticationUser`)
`certUserRequiredAttribs` notwendige Attribute (z.B. `objectclass,cn,sn,userCertificate;binary`)
`schemaCheck` gibt an, ob der LDAP-Server das Schema überprüft (`true` oder `false`)

Die Aktionen, die auf dem so spezifizierten LDAP ausgeführt werden, also wann welche Zertifikate und CRLs veröffentlicht und wieder entfernt werden, hängen von der jeweiligen Konfiguration des LDAP-Operators ab (siehe Abschnitt 5.5.9).

Konfiguration des Email-Versands Auch die Einstellungen für den zum Email-Versand eingesetzten SMTP-Server werden in einer Parametermenge zusammengefaßt, die standardmäßig den Namen „SMTP“ trägt. Eine abweichende Namensgebung ist den Operatoren, die SMTP benötigen, mitzuteilen.

`mailServer` der Hostname des LDAP-Servers (oder die IP-Adresse)

`smtpUser` der Benutzername zur Anmeldung

`smtpUserPass` das zugehörige Passwort

5.2.3 CA-Zertifikate

Für jede CA muß ein CA-Zertifikat vorliegen, anhand dessen die Signaturen dieser CA auf Zertifikaten, CRLs und Transferdateien verifiziert werden können. Das CA-Zertifikat wird im Konfigurationsverzeichnis für die jeweilige CA in der Datei `caCert.cer` erwartet. Eine abweichende Namensgebung kann in `ca.xml` eingetragen geben.

5.3 Starten

Kommandozeile Von der Kommandozeile kann die IS als eigenständige Applikation durch Eingabe von `ant itd` gestartet werden (das Kürzel `itd` hat historische Gründe und bedeutete *IS transfer demon* in Anspielung auf eine Komponente der RA). Wichtig ist hierbei, daß das aktuelle Arbeitsverzeichnis dem IS-Basisverzeichnis (`is`) entspricht, da sonst die Bezüge zu den Ein- und Ausgabeverzeichnissen nicht stimmen. Außerdem ist durch Setzen der Umgebungsvariablen `ANT_HOME` und `JAVA_HOME` das Funktionieren von Java und Ant sicherzustellen.

JMX Die IS kann auch mittels des Java Management Frameworks JMX über einen Webbrowser von einem anderen Rechner aus gestartet werden.

5.4 Stoppen

Zur Änderung der meisten Konfigurationsoptionen muß der IS-Dämon beendet werden: Die Konfigurationsdateien `is.ini` und `ca.xml` werden nur beim Start gelesen und ihre Einstellungen in den Hauptspeicher kopiert. Ebenso werden die Vorlagendateien zum Erzeugen der Benutzerausgaben und die CA-Zertifikate nur bei Hochfahren geladen. Dies führt zu Performanzvorteilen und erlaubt eine einmalige Überprüfung der Korrektheit der Parameter beim Systemstart, andererseits können die Einstellungen zur Laufzeit nicht mehr geändert werden.

Der IS-Dämon kann zwar durch das Prozeßmanagement des Betriebssystems beendet werden, ohne daß dadurch mit Verarbeitungsfehlern zu rechnen ist. Besser aber ist es, ein geregeltes Herunterfahren über die JMX-Schnittstelle auszulösen. Hierzu wird eine entsprechende Option in dem JMX-Frontend angeboten.

5.5 die mitgelieferten Operatoren

5.5.1 ConfirmDelivery

`ConfirmDelivery` verzögert die Abarbeitung des Profils solange bis ein bestimmter Datenbankeintrag erzeugt wurde, mit dem die Übergabe eines Schlüssel-paares an den Besitzer bestätigt wird. Dieser Eintrag muß nicht in der IS-Datenbank selbst erfolgen, sondern kann in beliebigen über JDBC erreichbaren Datenquellen getätigt werden.

Der Eintrag besteht aus dem Namen des Ausstellers und der Seriennummer des Zertifikats, sowie optional noch einem Zeitstempel und Kommentar.

Parameter:

`database` die JDBC-URL der Datenbank, optional, default ist die IS-Datenbank

`username` Benutzername für eine externe Datenbank, wird nur ausgewertet, wenn auch `database` gesetzt ist

`userpass` Zugangskennwort für eine externe Datenbank, wird nur ausgewertet, wenn auch `database` gesetzt ist

`table` Name der Tabelle, in der die Bestätigungen vermerkt werden, optional, default `confirmation`

`issuerDNColumn` Name der Spalte, in der der Ausstellernamen des Zertifikats vermerkt wird, optional, default `issuerDN`

`serialNumberColumn` Name der Spalte, in der die Zertifikatsnummer vermerkt wird, optional, default `serialNumber`

`commentColumn` Name der optionalen Spalte, in der ein Auslieferungskommentar eingetragen werden kann. Optional, falls nicht vorhanden, wird nach einer Spalte `comment` gesucht, und diese verwendet, wenn sie existiert. Wenn nicht, wird kein Kommentar mitgeführt.

`timestampColumn` Name der optionalen Spalte, in der der Auslieferungszeitpunkt eingetragen werden kann. Optional, falls nicht vorhanden, wird nach einer Spalte `timestamp` gesucht, und diese verwendet, wenn sie existiert. Wenn nicht, wird der Zeitpunkt nicht mitgeführt.

5.5.2 DeleteSecretInformation

`DeleteSecretInformation` löscht geheimzuhaltende Daten, die der IS nur zum Zwecke der Übergabe an den Empfänger vorlagen, aus der Datenbank. Es handelt sich dabei um Softtoken, deren Passworte, Chipkarten-PINs und Revokationspassworte.

Parameter:

`keepToken` falls gesetzt und mit Wert „true“ wird das Softtoken nicht gelöscht

`keepTokenPass` falls gesetzt und mit Wert „true“ werden Chipkarten-PINs und Passworte für Softtoken nicht gelöscht

`keepRevocPass` falls gesetzt und mit Wert „true“ werden Revokationspassworte nicht gelöscht. Ansonsten werden sie gelöscht und durch ihren Hashwert ersetzt, der zur Überprüfung des Passwortes ausreicht.

5.5.3 ErrorLogger

Zum Erstellen von Fehlerberichten dient der **ErrorLogger**. Er kann somit in den Fehlerprofilen des IS-Dämons verwendet werden. Fehlerberichte werden mithilfe des Template-Mechanismus erzeugt, so daß eine Vorlagendatei benötigt wird. Der Fehlerbericht wird dann in das Ausgabeverzeichnis geschrieben.

Parameter:

`outputDirectory` das Ausgabeverzeichnis für den Bericht, optional, default: `errors`

`templateFile` die Datei, die die Formatvorlage für den Bericht enthält

5.5.4 ExecuteProfile

`ExecuteProfile` unterbricht die Bearbeitung des aktuellen Profils und fährt statt dessen mit einem anderen Profil fort. Das Produkt wird dabei dem neuen Profil zugeordnet. Es ist möglich, nach Abarbeitung des neuen Profils wieder zum ursprünglichen zurückzukehren.

Parameter:

`profile` der Name des neuen Profils

`return` falls gesetzt und mit Wert „true“ wird nach Beendigung des neuen Profils mit dem alten fortgefahren

5.5.5 KeyBackup

`KeyBackup` speichert die in einem Softtoken enthaltenen Informationen in ein neues Softtoken um, das mit einem vorzugebenden Administratorschlüssel gesichert wird, und schreibt das neue Softtoken in eine Datei. Danach kann das alte Token gelöscht werden.

Parameter:

`keepToken` falls gesetzt und mit Wert „true“ wird das ursprüngliche Softtoken nicht automatisch gelöscht

`keepSigningKeys` falls gesetzt und mit Wert „true“ werden auch Signierschlüssel in das neu erzeugte Token aufgenommen, was normalerweise nicht geschieht

`outputDirectory` das Ausgabeverzeichnis für die Datei

`useCert` der Name der Datei, die den öffentlichen Schlüssel enthält, mit dem das neue Softtoken geschützt werden soll

5.5.6 MailNotification

`MailNotification` versendet Nachrichten per Email. Der Inhalt der Nachricht (inklusive Empfänger, Absender und Betreff) wird in einer Vorlagendatei beschrieben, wobei auf Zertifikats- und eventuell Kundendaten zurückgegriffen werden kann (siehe 5.6.1). Es können Zertifikate und Softtoken als Anhänge (Attachments) beigefügt werden. Die Einstellungen für den SMTP-Server werden in einer Parametermenge erwartet (siehe Abschnitt 5.2.2).

Der Operator kann auch Fehlerberichte per Email versenden. Hierzu dienen die beiden Einstellungen `errorTemplateFile` und `attachErrorReport`. Die Vorlagendatei für den Fehlerbericht entspricht in ihrem Aufbau exakt derjenigen für den Operator `ErrorReporter`. Um den Administrator beim Auftreten großer Mengen von Fehlern vor zahllosen Emails zu bewahren, gibt es weiterhin die Einstellung `interval`, die die Länge eines Zeitintervalles angibt, während dessen nach dem Versand einer Meldung weitere Fehlermeldungen unterdrückt werden.

Parameter:

`useSMTP` der Name der Parametermenge mit den SMTP-Einstellungen, optional, default: `SMTP`

`templateFile` die Datei, die die Formatvorlage enthält

`attachCertificate` falls gesetzt und mit Wert „true“ wird das im Produkt enthaltene Zertifikat als Anhang beigefügt (DER-codiert)

`attachToken` falls gesetzt und mit Wert „true“ wird das im Produkt enthaltene Softtoken als Anhang beigefügt

`attachErrorReport` falls gesetzt und mit Wert „true“ wird ein Fehlerbericht angefügt

`errorTemplateFile` die Datei mit der Formatvorlage für einen eventuellen Fehlerbericht

`interval` Anzahl an Minuten, die zwischen zwei Email vergehen muß (alle Nachrichten dazwischen werden verworfen)

5.5.7 RenewCRL

RenewCRL veranlaßt eine Erneuerung der Revokationsliste. Dies geschieht durch Ausfüllen eines entsprechenden RA-Formulars in der Datenbank.

Parameter:

days gibt an wieviele Tage vor Ablauf der CRL die Erneuerung angestoßen werden soll, optional, default 5

5.5.8 RenewX509

RenewX509 veranlaßt eine Rezertifizierung. Dies geschieht durch Ausfüllen eines entsprechenden RA-Formulars in der Datenbank, wobei das alte Zertifikat mit übergeben wird

Parameter:

days gibt an wieviele Tage vor Ablauf des Zertifikats die Erneuerung angestoßen werden soll, optional, default 5

5.5.9 UpdateLDAP

UpdateLDAP veröffentlicht Zertifikate und Revokationslisten im LDAP. Außerdem können alte und revozierte Zertifikate automatisch zurückgezogen werden. Der Operator verwendet eine Parametermenge, in der die LDAP-Einstellungen verzeichnet sind (siehe Abschnitt 5.2.2).

Parameter:

useLDAP der Name der Parametermenge mit den LDAP-Einstellungen, optional, default: LDAP

removeRevoked falls gesetzt und mit Wert „true“ werden die in einer CRL enthaltenen revozierten Zertifikate aus dem LDAP entfernt

5.5.10 WaitForExpiration

WaitForExpiration verzögert die restliche Ausführung des Produktprofils bis zu einem durch den Zertifikatsverfallszeitpunkt vorgegeben Datum.

Parameter:

days gibt an wie lange gewartet werden soll, in Tagen vor Ablauf des Zertifikats, optional, default 5

5.5.11 WriteCertificate

`WriteCertificate` schreibt die im Produkt enthaltenen Zertifikate in verschiedenen gängigen Formaten in Dateien heraus.

Parameter:

`outputDirectory` das Ausgabeverzeichnis für Zertifikate, optional, default: `certs`

`encodings` die gewünschten Zertifikatsformate, als durch Kommata getrennte Liste. Unterstützt werden DER, PEM und ein spezielles HTML-Format für den Microsoft Internet Explorer 5 (IE5).

`IE5TemplateFile` das HTML-Format IE5 benötigt eine Formatvorlagendatei

5.5.12 WriteCRL

`WriteCRL` schreibt die im Produkt enthaltene Revokationsliste in eine Datei heraus.

Parameter:

`outputDirectory` das Ausgabeverzeichnis für die Datei, optional, default: `crl`

5.5.13 WritePIN

`WritePIN` erzeugt anhand des Produkts und einer Formatvorlage einen PIN-Brief, der in eine Ausgabedatei geschrieben wird. Der genaue Inhalt ist durch die Formatvorlage vorgegeben, wobei die unten aufgeführten Elemente zur Verfügung stehen.

Parameter:

`outputDirectory` das Ausgabeverzeichnis für PIN-Briefe, optional, default: `PIN`

`templateFile` die Datei, die die Formatvorlage enthält

Vorlagenelemente:

`cert.*` zertifikatsbezogene Angaben (siehe 5.6.1)

`cert.revoc` das Revokationspaßwort für das Zertifikat

`token.filename` der Dateiname des Softtokens

`token.pass` das Paßwort für das Softtoken

`token.AdminPIN` die Admin-PIN (PUK) für die Chipkarte

`token.UserPIN` die User-PIN für die Chipkarte

`token.label` die Beschriftung der Chipkarte

`token.ID` die Identifikationsnummer der Chipkarte

5.5.14 WriteSoftToken

`WriteSoftToken` erzeugt eine PKCS#12-Datei, mit der das Schlüsselpaar an den Empfänger weitergegeben werden kann. Die Datei wird unverändert von der CA übernommen und ist üblicherweise durch ein Passwort geschützt.

Parameter:

`outputDirectory` das Ausgabeverzeichnis für die Datei, optional, default: `token`

5.6 Formatvorlagen

Zum Erzeugen von Textnachrichten (zum Beispiel für PIN-Briefe, Emails und Fehlerberichte) verwendet IS einen einfachen Template-Mechanismus. Dabei wird in einer Datei eine Formatvorlage definiert, die aus den festen Textstellen und Platzhaltern für die variablen Teile besteht.

Die Platzhalter sind mit symbolischen Namen versehen, ihre genaue Bedeutung und Verfügbarkeit hängt vom jeweiligen Kontext ab. Die Vorlagendatei ist eine gewöhnliche Textdatei, in der Platzhalter als `$${name}` verwendet werden können. Wenn ein Platzhalter nicht verfügbar ist, wird einfach eine leere Zeichenfolge eingefügt.

Die Namen der Platzhalter können geschachtelt werden. Beispielsweise definiert der Operator `WritePIN` zum Zugriff auf das Zertifikat einen Platzhalter `cert`, so daß die Seriennummer als `cert.SerialNumber` verfügbar wird.

5.6.1 Standardnamen für Zertifikatselemente

Wenn Informationen über ein Zertifikat eingebracht werden sollen, stehen die folgenden Bezeichner für Zertifikatsinhalte zur Verfügung:

`SerialNumber` die Seriennummer

`NotAfter` der Zeitpunkt, bis zu dem das Zertifikat gültig ist

`NotBefore` der Zeitpunkt, ab dem das Zertifikat gültig ist

`IssuerDN` der *distinguished name* des Zertifikatsausstellers

`SubjectDN` der *distinguished name* des Zertifikatsinhabers

`SubjectCN` der *common name* (Bestandteil des DN) des Zertifikatsinhabers

`SubjectPrettyCN` eine formatierte Version des CN, ohne die künstlichen Endungen `_ENC` oder `_SIGN`

`SubjectEmail` die Email-Adresse des Zertifikatsinhabers (aus dem DN entnommen)

```

to="$$${cert.SubjectEmail}"
subject="Your certificate expires soon"
from="flexiTrust"

Dear $$${cert.SubjectPrettyCN},

please be informed that your certificate expires soon,
on $$${cert.NotAfter}.

You will not be able to use it afterwards.

You should consider having it renewed.
For more information visit our web site
at http://flexiTrust.cdc.informatik.tu-darmstadt.de

Best regards,

your FlexiTrust team

```

Abbildung 5.1: Beispiel für die Definition einer Email-Nachricht.

5.6.2 Vorlagen für Emails

Speziell für Emailnachrichten gibt es ein erweitertes Schema für Formatvorlagen, mit dessen Hilfe der Absender, Empfänger und die Betreffzeile (*subject*) der Email festgelegt werden können, wobei auch im Email-Kopf variable Inhalte verwendet werden dürfen. Eine Beispielvorgabe für eine Email befindet sich in Abbildung 5.1.

5.7 Datenbankinhalt

Teile der IS-Datenbank sind auch für RA und externe Anwendungen einfach und sinnvoll nutzbar. Dabei ist vorwiegend ein lesender Zugriff angedacht. Änderungen des Datenbankinhalts sind nicht ohne genauere Kenntnis der internen Arbeitsweise der IS ratsam.

5.7.1 IS-Produkte

Die Information über IS-Produkte sind in der Tabelle `isproduct` gespeichert. Dies ist die wichtigste Tabelle für die Arbeit des IS-Dämons. Die Produkte werden als serialisierte Java-Objekte gespeichert, so daß sie nur mit speziellen Anwendungen sichtbar gemacht werden können, beispielsweise dem FlexiTrust „ProductInspector“, der am Fachgebiet erhältlich ist.

5.7.2 Zertifikate

In der Tabelle `x509cert` werden alle von der IS bearbeiteten Zertifikate abgelegt. Die wichtigsten Attribute sind für das Datenbanksystem sichtbar als

Spalten herausgestellt und stehen somit für Suchanfragen zur Verfügung. Das eigentliche Zertifikat liegt als DER-kodiertes Binärobjekt vor.

5.7.3 Revokationslisten

In der Tabelle `x509crl` werden alle von der IS bearbeiteten Revokationslisten abgelegt. Die wichtigsten Attribute sind für das Datenbanksystem sichtbar als Spalten herausgestellt und stehen somit für Suchanfragen zur Verfügung. Die eigentliche CRL liegt als DER-kodiertes Binärobjekt vor.

5.8 Fehlerbehandlung

5.8.1 Konfigurationsfehler der IS

Während dem Hochfahren des IS-Dämons überprüft dieser die in den verschiedenen Konfigurationsdateien getroffenen Einstellungen und bricht beim Auftreten von Fehlern ab (es sei denn, er befindet sich in dem im nächsten Abschnitt geschilderten Toleranzmodus). Es wird eine normalerweise aussagekräftige Fehlermeldung in die Logs geschrieben und auf dem Bildschirm ausgegeben, mithilfe derer es möglich sein sollte, einen Konfigurationsfehler zu beheben.

Änderungen an den Konfigurationsdateien werden erst nach einem Neustart des IS-Dämons wirksam (was nach einem abgebrochenem Startvorgang ohnehin notwendig ist).

5.8.2 fehlende Verfügbarkeit von externen Ressourcen

Gelegentlich scheitert das Laden der IS-Operatoren nicht an fehlerhaften Konfigurationsdaten sondern der fehlenden Verfügbarkeit von externen Ressourcen, beispielsweise dem Verzeichnisdienst LDAP. Um die Abarbeitung der davon nicht betroffenen Produkte nicht zu beeinträchtigen, bietet der IS-Dämon die Möglichkeit an, auf das Laden problematischer Operatoren zu verzichten und mit den verbliebenen weiterzuarbeiten.

Durch das Setzen des Properties `is.ignoreBrokenOperators` mit dem Wert `true` in der Datei `is.ini` schaltet der IS-Dämon in einen toleranten Modus und läuft auch mit einer Teilmenge an verfügbaren Operatoren weiter. Die von fehlenden Operatoren betroffenen Produkte werden verzögert (bis zum Erreichen ihrer Timeouts).

Es ist zu beachten, daß dieser Betriebsmodus nicht für den Dauereinsatz gedacht ist. Sobald das Verfügbarkeitsproblem behoben ist, sollte der IS-Dämon in den Normalzustand zurückgesetzt werden. Wenn die externen Ressourcen dauerhaft nicht verfügbar bleiben, muß über eine Entfernung der entsprechenden Operationen aus den Produktprofilen nachgedacht werden.

5.8.3 Fehler beim Einlesen von Transferdateien

Wenn beim Einlesen einer CA-Transferdatei ein Fehler auftritt, wird kein einziges der darin enthaltenen Processables bearbeitet. Die Datei wird geschlossen und in das Fehlerverzeichnis verschoben.

Die beiden häufigsten Fehlerursachen (und die einzigen, die während der Testphase aufgetreten sind) sind fehlerhafte Signaturen und eine inkompatible Kodierung der Processables. Ersteres kann eigentlich nur auftreten, wenn das von der IS verwendete CA-Zertifikat nicht dem von der CA tatsächlich verwendeten entspricht und kann entsprechend einfach korrigiert werden (oder die Signatur ist tatsächlich ungültig, und die Datei damit gefälscht). Die Datei kann nach einem Neustart der IS mit dem richtigen Zertifikat eingelesen werden.

Aufgrund der Arbeitsweise der Java-Objektserialisierung müssen die kompilierten Klassen der CA in identischer Form von der CA selbst und der IS verwendet werden. In einer tatsächlichen Installation sollte dies immer der Fall sein, während der Entwicklungszeit ist das Problem inkompatibler Klassenversionen jedoch oft aufgetaucht. Der Fehler ist im nachhinein (für bereits bestehende Transferfiles) nur dadurch zu beheben, daß der IS die jeweils verwendete Version der CA-Klassen untergeschoben wird (falls diese noch existieren).

5.8.4 Processables mit Fehlerstatus

Ein Processable, das einen Fehlerstatus meldet, wird von der IS nicht weiter bearbeitet. Statt dessen wird ein Fehlerbericht erzeugt, aus dem die Fehlermeldung des Processables hervorgeht. Da es sich hierbei um eine erfolglos abgebrochene Aufgabe der CA handelt, kann das Processable nicht mehr gerettet werden. Nach Behebung des Problems in der CA muß der Antrag von der RA erneut gestellt werden.

5.8.5 Fehler bei der Bearbeitung von Produkten

Wenn bei der Bearbeitung von Produkten durch einen IS-Operator ein Fehler auftritt, wird das Produkt von der weiteren Bearbeitung ausgeschlossen. Auf die Behandlung der übrigen Produkte und den IS-Dämon insgesamt hat dies keinen Einfluß. Ob und wie das fehlerhafte Produkt allerdings „geheilt“ werden kann, hängt ganz von der Art des aufgetretenen Fehlers ab. Es sind auf jeden Fall manuelle Eingriffe in den Datenbankinhalt notwendig. Hierzu sei auf den noch in der Entwicklung begriffenen „ProductInspector“ hingewiesen.

Kapitel 6

Ausblick

Diese Arbeit sollte als Schnappschuß über den Entwicklungsstand von FlexiTrust Anfang 2002 betrachtet werden. Da das Projekt in der letzten Zeit und sicher auch in Zukunft sehr aktiv war und über eine große Zahl an Mitarbeitern und Kooperationen verfügte, werden sich viele der hier geschilderten Gegebenheiten verändern, vor allem die Details der Implementierung. Eine Reihe von notwendigen und möglichen Änderungen und Erweiterungen lassen sich bereits heute absehen, andere werden sich erst später ergeben. Untenstehende Aufzählung erhebt folglich auch keinen Anspruch auf Vollständigkeit, noch soll sie eine Priorisierung andeuten.

- es ist durchaus möglich, daß sich das auf sequentielle Verarbeitung ausgerichtete Ablaufsteuerungssystem als zu simpel für bestimmte Anwendungen herausstellt. In diesem Fall müßte man es um parallele, bedingte und iterierte Verarbeitung erweitern, oder durch ein ganz anderes Modell ablösen.
- die Verarbeitungslogik der IS ist stark an die Produkte gekoppelt. Vielleicht möchte man auch nicht produktbezogene Tätigkeiten, zum Beispiel regelmäßige Archivierungsvorgänge, in die IS integrieren.
- wie bei den Ausführungen zum Template-Mechanismus bemerkt, muß man für verschiedene Sprachversionen unterschiedliche Produktprofile anlegen. Man kann sich eine Integration der Internationalisierung in den Template-Mechanismus vorstellen, derart, daß je nach der Sprache des Empfängers einer Nachricht automatisch das entsprechende Template gewählt wird.
- die mögliche Anbindung anderer RA/CA-Produkte könnte die IS zu einer eigenständiger Anwendungen machen, was aber im Rahmen von FlexiTrust vermutlich nicht erforderlich sein wird.
- zur Zeit sind die Möglichkeiten der Steuerung der IS über die zentrale Managementkomponente beschränkt. Sie läßt sich eigentlich nur starten und stoppen, und es können die Logfiles ausgewertet werden. Für fast alle Änderungen müssen die Konfigurationsdateien geändert werden, was mit einem Neustart der IS verbunden ist.

- gegenwärtig ist die IS so ausgelegt, daß sie keine sicherheitskritischen Informationen dauerhaft speichert, so daß ihre Betriebsumgebung weniger hohe Sicherheitsforderungen gestellt werden müssen. Infolge dessen verfügt sie allerdings auch über keinen eigenen Signierschlüssel. Wenn man beispielsweise die Datenbankverbindung absichern möchte oder signierte Nachrichten an Benutzer erzeugen will, wird dies notwendig. Dann muß aber auch der Zugang zur IS selbst abgesichert werden.
- ein wichtiger Punkt ist die Fehlertoleranz. Die IS ist in ihrer heutigen Form gut gegen Abstürze in Folge fehlerhafter Einstellungen und Eingaben geschützt. Auch ist die Bearbeitung unproblematischer Produkte von den auftretenden Fehlern bei anderen Produkten unbeeinflusst. Allerdings fehlen noch ausgefeilte Möglichkeiten, um jene fehlerhaften Produkte wieder zu „heilen“.

Wie man sieht, ist noch einiges zu tun, und ich möchte abschließend die Hoffnung zum Ausdruck bringen, daß die in dieser Arbeit entstandene Version der Infrastrukturkomponente sich als solide Basis für die zukünftigen Entwicklungen erweist und dem FlexiTrust-Projekt beim Erreichen seiner ehrgeizigen Ziele hilft.

Anhang A

Datenbankschema

```
-- RA to IS notification table
DROP TABLE IF EXISTS isnotification;
CREATE TABLE isnotification(
    productID varchar(255) NOT NULL,
    timeout datetime NOT NULL,
    received datetime,
    timedout datetime,
    primary key (productID)
);

-- IS to RA notification table (pre-filled forms)
DROP TABLE IF EXISTS ranotification;
CREATE TABLE ranotification(
    productID varchar(255) NOT NULL,
    form longblob,
    received datetime,
    primary key (productID)
);

-- delivery confirmation table
DROP TABLE IF EXISTS confirmation;
CREATE TABLE confirmation(
    issuerDN varchar(255) NOT NULL,
    serialNumber varchar(30) NOT NULL,
    timestamp timestamp NOT NULL,
    comment varchar(255),
    primary key (issuerDN, serialNumber)
);

-- CRL table
DROP TABLE IF EXISTS x509crl;
CREATE TABLE x509crl (
    IssuerDN varchar(255) NOT NULL,
    thisUpdate datetime NOT NULL,
    nextUpdate datetime NOT NULL,
    productID varchar(255),
    crl longblob,
```

```

    PRIMARY KEY (IssuerDN, thisUpdate)
);

-- main IS table (products)
DROP TABLE IF EXISTS isproduct;
CREATE TABLE isproduct (
    productID varchar(255) NOT NULL default '0',
    customerID varchar(255) default NULL,
    profile varchar(50) default NULL,
    nextOperation varchar(255) default NULL,
    resumeOn datetime default NULL,
    timeOut datetime default NULL,
    product longblob,
    PRIMARY KEY (productID)
) TYPE=MyISAM;

-- certificate table
DROP TABLE IF EXISTS x509cert;
CREATE TABLE x509cert (
    IssuerDN varchar(255) NOT NULL,
    serialNumber varchar(30) NOT NULL,
    SubjectDN varchar(255) NOT NULL,
    notBefore datetime NOT NULL,
    notAfter datetime NOT NULL,
    productID varchar(255),
    revocPWHash varchar(30) binary,
    revocPWHashAlg varchar(12),
    cert longblob,
    PRIMARY KEY (IssuerDN, serialNumber)
);

```

Anhang B

Beispiel für eine CA-Konfigurationsdatei

```
<?xml version="1.0"?>

<ca>

  <!-- ***** LDAP parameters -->

  <param-set name="ldap">
    <param key="serverName">130.83.23.174</param>
    <param key="userName">cn=root, c=de</param>
    <param key="userPass">secret</param>
    <param key="serverRootDN">C=DE</param>
    <param key="serverPort">1235</param>
    <param key="protocolVersion">3</param>
    <param key="insertionRule">o=org,ou=orgUnit,cn=certUser</param>
    <param key="orgObjectClasses">top,organization</param>
    <param key="orgRequiredAttribs">objectclass,o</param>
    <param key="orgUnitObjectClasses">top,organizationalUnit</param>
    <param key="orgUnitRequiredAttribs">objectclass,ou</param>
    <param key="certUserObjectClasses"
      >top,person,strongAuthenticationUser</param>
    <param key="certUserRequiredAttribs"
      >objectclass,cn,sn,userCertificate;binary</param>
    <param key="schemaCheck">true</param>
  </param-set>

  <!-- ***** CA importer parameters -->

  <importer>
    <param key="inputDirectory">../../../../P7Out</param>
    <param key="errorDirectory">../../../../P7Err</param>
  </importer>
```

```

<!-- ***** error handler -->

<error-handler>
  <operation type="ErrorLogger">
    <param key="templateFile">templates/ErrorLogger.template.txt</param>
    <param key="outputDirectory">../../isErrors</param>
  </operation>
</error-handler>

<!-- ***** SMTP settings -->

<param-set name="SMTP">
  <param key="mailServer">mail.cdc.informatik.tu-darmstadt.de</param>
  <param key="smtpUser">flexiPKI</param>
  <param key="smtpUserPass">flexiPKI</param>
</param-set>

<!-- ***** product profiles -->

<profile name="X509P12Request">
  <operation type="WriteCertificates" >
    <param key="outputDirectory">../../isOut</param>
    <param key="encodings">DER,PEM,IE5</param>
    <param key="IE5TemplateFile">templates/IE5.html</param>
  </operation>
  <operation type="WriteSoftToken">
    <param key="outputDirectory">../../isOut</param>
  </operation>
  <operation type="WritePIN">
    <param key="outputDirectory">../../isOut</param>
    <param key="templateFile">templates/P12Letter.template.txt</param>
  </operation>
  <operation type="MailNotification" name="MailP12" timeout-action="revoke">
    <param key="templateFile">templates/P12Email.template.txt</param>
    <param key="attachCertificate">true</param>
    <param key="attachToken">true</param>
  </operation>
  <operation type="ConfirmDelivery" />
  <operation type="DeleteSecretInformation" />
  <operation type="WaitForExpiration">
    <param key="days">1000</param>
  </operation>
  <operation type="MailNotification" name="ExpirationNotice">
    <param key="templateFile">templates/CertExpirationEmail.template.txt</param>
  </operation>
  <operation type="UpdateLDAP" />
</profile>

<profile name="X509P11Request">
  <operation type="WriteCertificates" />
  <operation type="DeleteSecretInformation" />
  <operation type="UpdateLDAP" />

```

```
        <operation type="RenewX509" />
    </profile>

    <profile name="X509CrtRequest">
        <operation type="WriteCertificates" />
        <operation type="DeleteSecretInformation" />
        <operation type="UpdateLDAP" />
        <operation type="RenewX509" />
    </profile>

    <profile name="X509CrlRequest">
        <operation type="WriteCRL" />
        <operation type="RenewCRL" />
        <param key="outputDirectory">../../isOut</param>
        <param key="days">3</param>
    </operation>
    </profile>

</ca>
```

Anhang C

Struktur der Konfigurationsdatei

Der Aufbau einer XML-Datei kann in einer sogenannten DTD (*document type definition*) in einer maschinell verarbeitbaren Form angegeben werden. Eine DTD selbst ist kein XML-Dokument, sondern ein SGML-Dokument mit einer recht komplexen Syntax. Durch die Angabe einer DTD können XML-Dateien von entsprechenden Tools validiert werden. Außerdem kann sie von einigen XML-Editoren verwendet werden.

```
<!--
  XML DTD for the ca.xml configuration files used by IS.
  Corresponds to castor mapping file cainfo-mapping.xml.

  Author: Thilo Planz
-->

<!ELEMENT ca (param-set*, importer, error-handler?, param*, profile*)>

<!ELEMENT param-set (param*)>
<!ATTLIST param-set name CDATA #REQUIRED>

<!ELEMENT param (param-value)>
<!ATTLIST param key CDATA #REQUIRED>

<!ELEMENT param-value (#PCDATA)>

<!ELEMENT importer (param*)>

<!ELEMENT error-handler (operation*)>

<!ELEMENT operation (param*, use-param-set*)>
<!ATTLIST operation
  type CDATA #REQUIRED
  name CDATA #IMPLIED
  timeout-action CDATA #IMPLIED
  >
```

```
<!ELEMENT use-param-set EMPTY >  
<!ATTLIST use-param-set key CDATA #REQUIRED>  
  
<!ELEMENT profile (operation*)>  
<!ATTLIST profile name CDATA #REQUIRED>
```

Anhang D

Abkürzungsverzeichnis

ASN.1 *abstract syntax notation.*

eine Beschreibungssprache für Datenstrukturen. Wird unter anderem bei X.509 eingesetzt. ASN.1-Daten werden im Gegensatz zu XML-Daten in einem kompakten Binärformat gespeichert.

CA *certification authority.*

Zertifizierungsinstanz, Trustcenter. Vertrauenswürdige Stelle, die die eindeutige Zuordnung von öffentlichen Schlüsseln zu den Mitgliedern der Kommunikationsgemeinschaft garantiert.

DN *distinguished name.*

Nach dem X.500-Standard für Verzeichnisdienste verfügt jeder Teilnehmer über einen eindeutigen Namen, der aus mehreren Namensbestandteilen (z.B. Organisationsname, Ländername, Name der Person) aufgebaut ist, zwischen denen eine Hierarchie besteht. Auch die Teilnehmer einer X.509-PKI werden durch ihren DN identifiziert.

JDBC nicht *java database connectivity.*

Schnittstelle zum plattformunabhängigen Zugriff auf Datenbanken (und andere Datenquellen) für die Programmiersprache Java

LDAP *light-weight directory access protocol.*

vereinfachte (und verbreitetere) Alternative zum X.500-Verzeichnisdienst

OCSP *online certificate status protocol.*

Online-Verfahren zur Abfrage der Gültigkeit von Zertifikaten. Als Alternative zu Revokationslisten können Clients den Status von Zertifikaten bei einem OCSP-Server abfragen.

PGP *pretty good privacy.*

populäres Programmpaket zur Erstellung signierter und verschlüsselter Nachrichten. PGP basierte ursprünglich auf einem System von Vertrauensketten anstelle einer gewöhnlichen PKI.

- PIN** *persönliche Identifikationsnummer.*
geheimzuhaltende Nummer, mit der die Verwendung von Chipkarten abgesichert wird. Normalerweise numerisch und vier- oder sechstellig mit einer sehr geringen Anzahl an Fehlversuchen.
- PKCS** *public key cryptography standards.*
von den RSA-Laboratories entwickelte Familie von Standards. Bedeutend sind vor allem PKCS#11 für den Zugriff auf Chipkarten und PKCS#12, in dem ein geschütztes Transportformat für Schlüsselpaare beschrieben ist.
- PKI** *Public-Key-Infrastruktur.*
System aus Richtlinien, Abläufen, Institutionen und Datenformaten zur Verwaltung der in der asymmetrischen Kryptographie benötigten öffentlichen Schlüssel
- PUK** *PIN unblocking key.*
Admin-PIN, mit der eine gesperrte PIN einer Smartcard wieder freigeschaltet werden kann
- RMI** *remote method invocation.*
von der Programmiersprache Java angebotener Mechanismus zur rechnerübergreifenden Kommunikation von Anwendungen
- S/MIME** *secure multipurpose internet mail extensions.*
Standard für Verschlüsselung und digitale Signaturen in Email-Nachrichten
- SQL** *structured query language.*
text-basierte Abfragesprache als Standardschnittstelle zu fast allen Datenbanksystemen
- SSL** *secure socket layer.*
Protokoll zur sicheren Kommunikation über das Internet. Server und optional Client authentifizieren sich mit Zertifikaten, die übertragenen Daten werden mit einem Einmalschlüssel verschlüsselt.
- X.509** *ITU Empfehlung X.509, auch ISO/IEC 9594-8.*
dominierender PKI-Standard. Enthält unter anderem ein flexibles Zertifikatsformat, das mittlerweile in fast allen Anwendungen verwendet wird.
- XML** *extensible markup language.*
universelles Austauschformat für strukturierte Daten. XML zeichnet sich gegenüber dem älteren SGML durch eine stark vereinfachte Syntax aus, die die Entwicklung von XML-fähigen Anwendungen vereinfacht hat. XML hat in letzter Zeit eine große Verbreitung erreicht. XML ist im Gegensatz zu ASN.1 kein Binärformat, sondern textbasiert.

Index

- Ablaufsteuerungssystem, 29
- Advanced Processing Infrastructure,
 - 18
- ASN.1, 75
- Authentifizierung, 5
- bulk encryption key, 8
- CA, 11, 75
- ca.xml, 53
- Castor, 39
- certification authority, 11
- Chipkarte, 22
- ConfirmDelivery, 57
- Dämon, 33
- Datenbank, 32, 44, 63
 - Tabellen, 32
- DeleteSecretInformation, 57
- digitale Signaturen, 8
- Direct Trust, 10
- distinguished name, 15, 75
- DN, 75
- Einmal-Schlüssel, 8
- ErrorLogger, 58
- ExecuteProfile, 58
- fingerprint, 9
- FlexiTrust, 16
 - CA, 17
 - IS, 18
 - RA, 16
- Formatvorlage, 62
- Framework, 25
- FreeObject, 47
- Hashfunktionen, 9
- Hosting, 32
- Integrität, 5
- Internationalisierung, 49
- IS-Dämon, 33
- is.ini, 53
- Java, 35
- JavaDoc, 35
- JDBC, 75
- JMX, 48
- key escrow, 12
- key recovery, 12
- KeyBackup, 58
- Kryptographie, 5
 - asymmetrische, 7
 - hybride, 8
 - symmetrische, 6
- LDAP, 14, 55, 75
- Logging, 49
- Lokalisierung, 49
- MailNotification, 59
- man-in-the-middle, 10
- message digest, 9
- OCSP, 12, 75
- Operation, 30
- Operator, 28, 57
- Operator, 40
- PGP, 10, 75
- PIN, 75
- PKCS, 14, 76
- PKI, 9, 76
- Processable, 18
- Produkt, 26
- Profil, 27
- Public-Key-Infrastruktur, 9
- PUK, 76
- RA, 11

- registration authority, 11
- Registrierungsstelle, 11
- RenewCRL, 59
- RenewX509, 60
- Revokationsliste, 11, 12
 - X.509, 14
- Rezertifizierung, 12
- RMI, 76

- S/MIME, 13, 76
- Schlüsselaustausch, 10
- Schlüsselerzeugung, 11
- Serialisierung, 33
- session key, 8
- Softtoken, 14, 23
- SQL, 76
- SSL, 13, 76

- Tabellen, 32
- Template, 46, 62
- Timeout, 30
- Transferdatei, 17
- Trustcenter, 12

- UpdateLDAP, 60

- Verbindlichkeit, 5
- Verifikation, 8
- Vertrauenskette, 10
- Vertraulichkeit, 5

- WaitForExpiration, 60
- Wiedervorlagezeitpunkt, 30
- WriteCertificate, 60
- WriteCRL, 61
- WritePIN, 61
- WriteSoftToken, 62

- X.509, 13, 76
- XML, 39, 76

- Zeitlimit, 30
- Zertifikat, 11
 - X.509, 13
- Zertifizierung, 11

Literaturverzeichnis

- [Buc01] BUCHMANN, JOHANNES: *Einführung in die Kryptographie*. Springer, zweite, erweiterte Auflage, 2001.
- [CDC] Technische Universität Darmstadt, Fachbereich Informatik, Fachgebiet Kryptographie und Computeralgebra. Alexanderstraße 10, 64283 Darmstadt <http://www.informatik.tu-darmstadt.de/TI/>.
- [CKLW00] CAMPHAUSEN, INGMAR, KELM, STEFAN, LIEDKE, BRITTA und WEBER, LARS: *Aufbau und Betrieb einer Zertifizierungsinstanz*. Deutsches Forschungsnetz, DFN-PCA – Vogt-Kölln-Straße 30 – 22527 Hamburg, März 2000.
- [Dam01] DAMBRUCH, JENS: *OpenRA – Framework zur flexiblen Formularverarbeitung im Rahmen von FlexiTRUST*. Diplomarbeit, Technische Universität Darmstadt, Juni 2001.
- [GHJV96] GAMMA, ERICH, HELM, RICHARD, JOHNSON, RALPH und VLISIDES, JOHN: *Entwurfsmuster*. Addison-Wesley, 1996.
- [PKCS] RSA LABORATORIES: *Public-Key Cryptography Standards*. <http://www.rsa.com/rsalabs/pkcs/>.
- [Sch96] SCHNEIER, BRUCE: *Angewandte Kryptographie*. Addison-Wesley, 1996.
- [Sch01] SCHUSTER, MARKUS: *OpenRA – Framework zur flexiblen Formularverarbeitung*. Diplomarbeit, Technische Universität Darmstadt, Juni 2001.
- [SigG01] DEUTSCHER BUNDESTAG: *Gesetz über Rahmenbedingungen für elektronische Signaturen*.
- [SigV01] REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND: *Verordnung zur elektronischen Signatur*.
- [Tak99] TAK, MARKUS: *Public Key Infrastrukturen – ein Java-basiertes Trustcenter*. Diplomarbeit, Technische Universität Darmstadt, Juli 1999.
- [Wie01] WIESMAIER, ALEXANDER: *FlexiTrust CA – Ein flexibles, skalierbares und dynamisch erweiterbares Trustcenter*. Diplomarbeit, Technische Universität Darmstadt, Februar 2001.