UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Public-Key-Infrastrukturen nach X.509 – simuliert als JCrypTool-Plugin

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers
im Studiengang Informatik

vorgelegt von

Johannes Lohrum

Erstgutachter:	Prof. Dr. R. Grimm Institut für Wirtschafts- und Verwaltungsinformatik
Zweitgutachter:	Dipl.-Inf. D. Pähler Institut für Wirtschafts- und Verwaltungsinformatik

Koblenz, im März 2011

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Kapitel 1

Einleitung

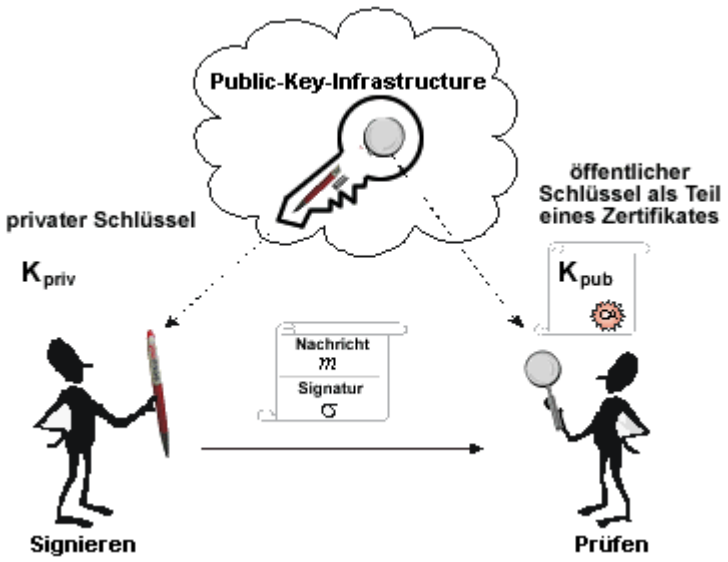

Abbildung 1.1: Signatursystem

[1]

Im Rahmen dieser Abschlußarbeit wurde ein Plugin zur Visualisierung/Simulation von Public-Key-Infrastrukturen für die Kryptographie-Lernsoftware *JCrypTool* entwickelt und implementiert. Public-Key-Infrastrukturen stellen einen Bereich in der Kryptographie dar, mit dem viele Anwender von IT-Systemen in Berührung kommen. Meist geschieht dieser Kontakt jedoch eher unbemerkt, da

[1] https://www.bsi.bund.de/cln_183/ContentBSI/Publikationen/Faltblaetter/F10ElektronischeSignatur.html

die Aufgaben von Anwendungen im Hintergrund erledigt werden. Ein häufiges Beispiel wird wohl der Zugriff auf Webinhalte mittels des Protokolls *https* sein. In diesem Fall überprüft der Browser die Authentizität des entfernten Servers über ein Zertifikat. Dahinter steckt eine Public-Key-Infrastruktur, mit deren Hilfe die Anwendung prüfen kann, ob das durch den Server präsentierte Zertifikat authentisch ist – oder eben nicht. Der *normale* Anwender sieht Public-Key-Infrastrukturen allerdings immer nur aus der Benutzerperspektive – meist als Zertifikatanwender, seltener als Zertifikatinhaber. Die Abläufe und Vorgänge in Public-Key-Infrastrukturen lassen sich aus dieser Perspektive allerdings nicht so leicht durchschauen.

Die Grafik zu Beginn zeigt die Anwendung von digitalen Signaturen. Der Signierer hat eine Nachricht mit seinem privaten Schlüssel signiert und an die zweite Person gesandt. Der Prüfer verifiziert nun die Echtheit der Nachricht mit Hilfe des zum Signierschlüssel gehörenden öffentlichen Schlüsses. Aufgabe der Public-Key-Infrastruktur ist, kurz gesagt, die Zusicherung, dass der öffentliche Schlüssel auch wirklich zu dem, für die Signatur verwendeten, privaten Schlüssel des Signierers gehört. Hierfür sind verschiedene Stellen zuständig, die unterschiedliche Dokumente miteinander austauschen. Ein Teil hiervon soll mit der entwickelten Simulation anschaulich gemacht werden.

JCrypTool ist eine auf der Eclipse Rich Client Platform basierende Open-Source-Software. Es ist 2007 als Zweig aus dem CrypTool-Projekt entstanden. Die Entwicklung von CrypTool begann 1998 im Auftrag der Deutschen Bank mit dem Ziel Mitarbeiter im Bereich IT-sicherheit zu sensibilisieren. Die aktuelle Version von CrypTool ist 1.4.30 aus 2010.

Motivation Kann ein Simulationsprogramm beim Nachvollziehen der Funktionsweise von Public-Key-Infrastrukturen nützlich sein?

> „If I tell you, you will forget. If I show you, you will remember. If I involve you, you will understand.“

Dieses Zitat, das Konfuzius zugeschrieben wird, soll verdeutlichen warum die Entwicklung eines Programms zur Simulation von Public-Key-Infrastrukturen

sinnvoll ist. Passives Zuhören führt dazu, dass Dinge schnell wieder vergessen werden. Werden Sachverhalte visuell vermittelt, können sie zumindest so abgespeichert werden, dass sie wiedergegeben werden können. Verstanden wird etwas aber nur, wenn sich aktiv damit beschäftigt wird. Dies deckt sich mit modernen Theorien, die besagen, dass Lernstoff – abhängig von der Art der Vermittlung – zu unterschiedlichen Anteilen behalten wird:

Behalten von Lernstoff:

- 10% durch lesen,
- 20% durch hören,
- 30% durch sehen,
- 50% durch hören und sehen,
- 70% durch erklären bzw. wiedergeben,
- 90% durch anwenden.

Gliederung der Arbeit Ganz zum Schluss!

Hinweis In der entwickelten Lernsoftware wurde Wert darauf gelegt, die Strukturen und Prozessabläufe innerhalb einer Public-Key-Infrastruktur anschaulich darzustellen. Alle Schlüssel- und Zertifikatspeicher sind lediglich mit einem Standardkennwort geschützt. Wenn Schlüssel oder Zertifikate exportiert werden, liegen sie frei zugänglich im Dateisystem. Ebenso wurden keine besonderen Vorkehrungen getroffen, um private Schlüssel im Speicher zu schützen. Im Gegenteil, dem Anwender soll ganz bewusst die Möglichkeit gegeben werden mit fremden Schlüsselpaaren zu experimentieren und so z.B. die Situation nach einem Schlüsseldiebstahl durchzuspielen.

Warnung Aus diesem Grund wird ausdrücklich davor gewarnt Passwörter oder (private) Schlüssel aus dem Produktiveinsatz in die Simulationsumgebung zu importieren bzw. in ihr zu verwenden. „Echte“ Zertifikate können jedoch bedenkenlos importiert werden, da sie aus ihrer Bestimmung heraus automatisch öffentlich zugänglich sind/sein sollen und somit nicht besonders schützenswert sind.

Kapitel 2

Grundlagen

In diesem Abschnitt werden die für die Pluginentwicklung relevanten Grundlagen zusammengestellt. Dies stellt einerseits die Grundlage für die Entwicklung der Simulationsumgebung dar, andererseits für die Begleitinformationen, die der Anwender im JCrypTool-Anwenderhandbuch nachlesen kann.

2.1 Asymmetrische Verschlüsselung

Die Notwendigkeit von Public-Key-Infrastrukturen ist in der Verwendung asymmetrischer Verschlüsselung begründet – ohne diese wären Public-Key-Infrastrukturen nicht erforderlich. Aus diesem Grund werden asymmetrische Kryptosysteme zunächst kurz vorgestellt und mit der symmetrischen Verschlüsselung verglichen. Schließlich werden die Probleme vorgestellt, die mit Public-Key-Infrastrukturen gelöst werden können.
Verschlüsselungsverfahren dienen ... dem Erreichen der Sicherheitsziele *Vertraulichkeit*, *Authentizität*, *Nichtabstreitbarkeit*, *Zurechenbarkeit*, ... Integrität Vertraulichkeit wird durch Verschlüsselung der jeweiligen Daten gewährleistet, zur Erreichung der anderen Sicherheitsziele werden die Daten digital signiert. Beispiele

Dies

symmetrische Verschlüsselung Bei der symmetrischen Verschlüsselung wird derselbe Schlüssel zum Ver- und Entschlüsseln genutzt. Möchte eine Person A einen verschlüsselten Text an B senden, so muss sie erst mit B einen Schlüssel austauschen. Da zu diesem Zeitpunkt noch keine abgesicherte Kommunikation

zwischen A und B möglich ist, muss der Schlüsselaustausch auf einem anderen Weg stattfinden.
Solange nur A und B miteinander kommunizieren wollen, reicht der eine zwoschen beiden ausgetauschte Schlüssel aus. Möchten jetzt aber A und B auch noch mit C kommunizieren, so sind insgesamt schon drei Schlüssel notwendig (A-B, B-C und A-C). Kommen noch weitere Kommunikationspartner hinzu, wächst die benötigte Schlüsselzahl schnell an.

Im Gegensatz zur symmetrischen Verschlüsselung, arbeiten asymmetrische Verfahren arbeitet mit zwei verschiedenen Schlüsseln für die Ver- und Entschlüsselung mit folgenden Eigenschaften:

- mathematischer Zusammenhang
- ein Schlüssel kann aus dem anderen nicht abgeleitet werden

Schlüsselpaare aus privaten und öffentlichen Schlüsseln öffentlicher Schlüssel kann ungeschützt übertragen werden, in Verzeichnisse eingestellt werden, ... pro Teilnehmer ein Schlüsselpaar notwendig

Anwendungsfall Verschlüsselung Vereinfacht kann man sich die Funktionsweise asymmetrischer Verschlüsselung anschaulich machen mit folgender Analogie: ein Briefkasten hat einen Einwurfschlitz und eine Tür mit Schlüssel. Der Einwurfschlitz entspricht dem öffentlichen Schlüssel und die Tür entspricht dem privaten Schlüssel. Jeder kann in den Briefkasten etwas hineinwerfen – jeder kann mit einem öffentlichen Schlüssel etwas verschlüsseln. Sobald etwas in den Briefkasten geworfen wurde – also mit dem öffentlichen Schlüssel verschlüsselt wurde, kann man es ohne den Briefkastenschlüssel nicht mehr „entschlüsseln“ .

Anwendungsfall Signatur Eine weitere Anwendung asymmetrischer Verfahren stellt die Signatur dar. Ziel der Signatur ist es die Integrität und die Authentizität eines Dokuments zu schützen. Der Signierer bildet einen Hashwert aus dem Inhalt und verschlüsselt diesen mit seinem privaten Schlüssel. Dokument und diese Signatur werden zusammen ausgeliefert. Zur Verifizierung wird die mitgelieferte Signatur (der verschlüsselte Hashwert) mit dem öffentlichen Schlüssel entschlüsselt. Über den Inhalt des Dokuments wird wieder ein Hashwert gebildet. Stimmern dieser Hashwert und das Ergebnis der Entschlüsselung überein, so ist sichergestellt, dass das Dokument mit dem zum öffentlichen

Schlüssel gehörenden privaten Schlüssel verschlüsselt und danach nicht mehr verändert wurde.

Probleme asymmetrische Verschlüsselung Das Schlüsselaustauschproblem bei der symmetrischen Verschlüsselung existiert in der asymmetrischen Variante nicht. Der private Schlüssel verbleibt immer beim Besitzer, während der öffentliche Schlüssel – wie der Name schon sagt – frei zugänglich ist. Dafür ergeben sich aber andere Probleme[1]:

Authentizität der Schlüssel Der öffentliche Schlüssel enthält keine Informationen, an welchen erkennbar ist, zu welchem Subjekt er gehört. Möchte eine Person A einer Person B ein verschlüsseltes Dokument senden, so wird sie hierzu den öffentlichen Schlüssel von B verwenden.

Sperrung von Schlüsseln Hat sich ein Dritter des privaten Schlüssels von A bemächtigt, so kann er sämtliche für A verschlüsselten Dokumente lesen. A muss ein neues Schlüsselpaar verwenden und die Information, dass der alte öffentliche Schlüssel nicht mehr genutzt werden darf verbreiten.

Verbindlichkeit Solange öffentlich Schlüssel nicht einem Subjekt zugeordnet sind, taugt eine Signatur lediglich der Feststellung, dass das signierte Dokument seit der Signatur mit dem zugehörenden privaten Schlüssel nicht mehr verändert wurde. Der Signierer abstreiten die Signatur angefertigt zu haben, indem er behauptet nicht im Besitz des privaten Schlüssels zu sein.

Die Lösung für alle drei Problemsituationen ist die verbindliche Zuordnung von öffentlichen Schlüsseln und den Inhabern des jeweiligen privaten Schlüssels.

2.2 Zertifikate (allgemein)

Ein Zertifikat ist ein signiertes Dokument, in dem die Zusammengehörigkeit eines öffentlichen Schlüssels und eines Subjekts besätigt wird. Dies löst das Problem, dass ein öffentlicher Schlüssel einem Subjekt nicht zuzuordnen ist. Hierfür muss ein Zertifikat mindestens aus der Beschreibung (Name) des Subjekts, seinem öffentlichen Schlüssel und der Signatur bestehen. Darüber hinaus können

[1] siehe [Sch07, S. 442f]

noch Informationen zur Gültigkeit, zum Aussteller, eine Seriennummer und der öffentliche Schlüssel des Ausstellers enthalten sein.

Zertifikate haben unser Problem allerdings nicht gelöst, sonder lediglich verlagert. Durch das Zertifikat haben wir eine Bestätigung, dass der öffentliche Schlüssel und das Subjekt zusammen gehören. Die nächste Frage, die sich stellt ist, ob der Aussteller des Zertifikats die Tatsachen gründlich genug überprüft hat oder ob er vielleicht vorsätzlich einen falschen Zusammenhang bescheinigt hat. Vielleicht hat er die Zugehörigkeit eines eigenen öffentlichen Schlüssels zum anderen Subjekt beschinigt – jetzt könnte er alle verschlüsselte Kommunikation, die an das andere Subjekt gerichtet ist entschlüsseln. Die lösung des Problems finden wir im Vertrauen, dass wir in den Zertifikataussteller haben.

2.3 Vertrauensmodelle

Stellen die Kommuniksationspartner sich für ihre öffentlichen Schlüssel jeweils Zertifikate aus, um sie schließlich auf sicherem Weg an den anderen zu übermitteln, stellt dies der einfachste Fall eines Vertrauensmodells dar - Direct Trust[2]. Je nachdem muss nach Übermittlung der Zertifikate noch überprüft werden, ob sie während dem Transport nicht ausgetauscht oder verändert wurden, dies geschieht am einfachsten durch Abgleich der Hashwert über einen weiteren Übertragungsweg (z.B. per Telefon). Für eine kleine Menge an bekannten Kommunikationspartnern ist dieses Modell brauchbar und ohne dritte Stellen nutzbar. Bei einer größeren Anzahl an Gesprächspartnern wird dieses System jedoch sehr aufwendig.

Web of Trust Eine Verbesserung stellt das „Web of Trust“dar. Hierbei wird das bestehende direkte Vertrauen eines Partners in einen Dritten ausgenutzt. Vertraut eine Person A einer anderen Person B und B wiederum einem Dritten C, so kann auch A dem Dritten C vertrauen. Die Grundlage dieses Vertrauens ist die Zuversicht von A, dass sich B die Kommunikationspartner, denen sie vertraut sehr sorgfältig auswählt. Diese Art der Vertrauensbezoehung zwischen A und C nennt man auch *transitives Vertrauen*. Das Modell des Web of Trust ist schon deutlich brauchbarer als das Direct Trust.

[2]vgl. [Sch07, S. 446]

Vertrauenshierarchie

In diesem Modell gibt es eine zentrale Stelle, die Zertifikate ausstellt – die Zertifizierungsinstanz. Diese Variante ist am aufwendigsten in der Realisierung, stellt doch gerade die Zertifizierungsinstanz einen Angriffspunkt dar, der besonders geschützt sein muss. Je nachdem ob sich zwischen der Zertifizierungsinstanz und den Nutzern noch weitere Zwischeninstanzen befinden, spricht man von ein- oder mehrstufigen Hierarchien. Existieren verschiedene Zertifizierungsinstanzen ohne einen hierarchischen Bezug untereinander, spricht man vom Webmodell. Hier stellen alle Instanzen unabhängig voneinander Zertifikate für die Nutzer aus – ein Nutzer kann Zertifikate bei verschiedenen Zertifizierungsinstanzen beantragen.

Kapitel 3

Public-Key-Infrastrukturen nach X.509

Public-Key-Infrastrukturen nutzen das Modell der Vertrauenshierarchie. An der Spitze steht die Wurzelinstanz, die den Vertrauensanker der PKI darstellt. Alle Teilnehmer der PKI müssen der Wurzelinstanz vertrauen und alle Zertifikate können auf das Zertifikat der Wurzelinstanz zurückgeführt werden. Im diesem Kapitel werden die einzelnen Komponenten einer Public-Key-Infrastrukturen[1], die verwendeten Datenobjekte, die Abläufe innerhalb der Komponenten sowie deren Kommunikation untereinander beschrieben. Dies erfolgt mit Schwerpunkt auf die für die Entwicklung des Programms wichtigen Aspekte – andere Dinge werden entweder nur kurz erwähnt oder weggelassen.

Die Abbildung[2] 3.1 veranschaulicht die Komponenten, Datenobjekte und Vorgänge innerhalb einer Public-Key-Infrastruktur sehr schön. Der Benutzer (oder Endteilnehmer) besitzt ein asymmetrisches Schlüsselpaar. Er beantragt ein Zertifikat bei der Registrierungsinstanz (RA), die den Antrag prüft und an die Zertifizierungsinstanz (CA) weiterleitet. Diese stellt das Zertifikat aus und händigt es an den Antragsteller aus. Ebenso übermittelt sie die von ihr ausgestellten Zertifikate an den Validierungsdienst (VA). Damit ist die Ausstellung des Zertifikats abgeschlossen. Der Benutzer erstellt nun ein, mit dem zum Zertifikat gehörenden privaten Schlüssel signiertes, Dokument und sendet es an eine dritte Stelle. Die-

[1] PKI nach X.509 werden u.a. beschrieben in [C+08] und [ITU05]

[2] vgl. http://de.wikipedia.org/w/index.php?title=Datei:Public-Key-Infrastructure.svg

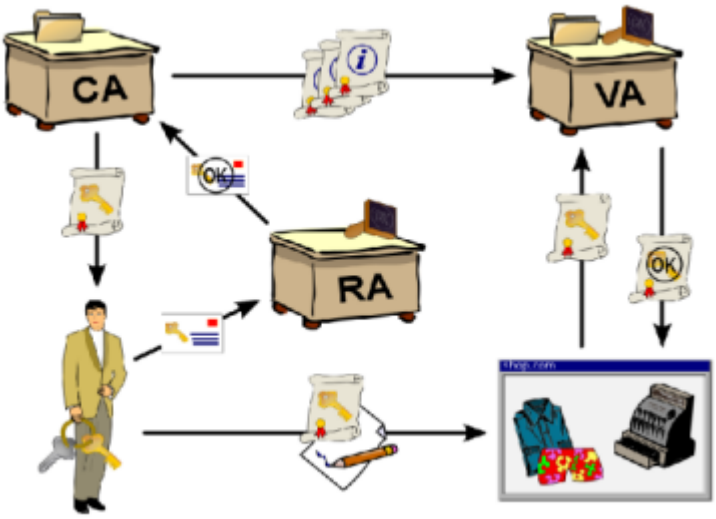

Abbildung 3.1: Beispiel einer Public-Key-Infrastruktur

se überprüft mit Hilfe des Validierungsdienstes die Echtheit des Zertifikats und kann im Erfolgsfall das signierte Dokument anerkennen.
Dies ist ein Beispiel Modell für eine Public-Key-Infrastruktur mit einer Zertifizierungsinstanz.

3.1 Komponenten in einer X.509-PKI

Im folgenden Abschnitt werden die Beteiligten einer Public-Key-Infrastruktur und ihre Aufgaben vorgestellt.

3.1.0.1 Endteilnehmer

sind die eigentlichen Nutznießer einer Public-Key-Infrastruktur. Als Zertifikatinhaber signieren sie Dokumente mit einem von einer Zertifizierungsinstanz ausgestellen Zertifikat. Als Zertifikatnutzer überprüfen sie die Gültigkeit einer Signatur mit Hilfe der PKI. Endteilnehmer können neben Personen auch Computersysteme (als Zertifikatinhaber) oder Anwendungen (als Zertifikatnutzer) sein.

3.1.0.2 Zertifizierungsinstanzen

sind die Hauptdienstleister in Public-Key-Infrastrukturen. sie bilden (ggf. zusammen mit Registrierungsinstanzen das sogenannte Trust Center. Ihre Aufgaben sind das Ausstellen von zertifikaten für andere PKI-Teilnehmer – sowohl Endteilnehmer, als auch andere Zertifizierungsinstanzen. Eine weitere Aufgabe ist das Sperren von Zertifikaten und die Ausstellung von Zertifikatsperrlisten. Gegebenenfalls kann auch die Generierung von Schlüsselpaaren für die Antragsteller von der Zertifizierungsinstanz übernommen werden. In diesem Fall kann sie, falls der Zertifikatinhaber den privaten Schlüssel verliert, diesen wiederherstellen. Zum Signierschlüssel der Zertifizierungsstelle gehört wiederum ein Zertifikat, dass von einer übergeordneten Instanz ausgestellt wurde.
Zertifizierungsinstanzen sind kritische Stellen einer PKI. Wird ihr privater Schlüssel gestohlen, müssen alle von ihr ausgestellten Zertifikate widerrufen werden.

Wurzelinstanz Die Wurzelinstanz steht an der Spitze einer Public-Key-Infrastruktur und ist eine besondere Zertifizierungsinstanz. Das Zertifikat für ihren Schlüssel ist von ihr selbst ausgestellt und signiert. Die Wurzelinstanz stellt den Vertrauensanker einer PKI dar – alle Zertifikate können auf sie zurückgeführt werden.

Bridge-CA Die Bridge-CA ist eine spezielle Form der Wurzelinstanz[3]. Sie ist eine unabhängige Stelle, die von verschiedenen Public-Key-Infrastrukturen zur Crosszertifizierung genutzt wird. Wäre die Bridge-CA keine Wurzelinstanz, wäre sie ja wieder von einer PKI abhängig. Bei Nutzung einer Bridge-AA zur Verbindung von Public-Key-Infrastrukturen, muss sich keine PKI der anderen unterordnen. Die Verbindung kann auf „gleicher Augenhöhe“ eingegangen werden.

intermediate CA In einer mehrstufigen Hierarchie bilden *intermediate CAs* die mittlere Schicht von Zertifizierungsinstanzen zwischen der Wurzelinstanz und den Endteilnehmern.

Unterscheidung[4] Im englischsprachigen Raum wird überwiegend der Begriff „Certification Authority“ an Stelle von „Trust Center“ benutzt. Dies kann leicht zu Missverständnissen mit dem deutschen Begriff „Zertifizierungsstelle“ führen,

[3] steht zwar nirgends, ist aber sinnvoll
[4] (vgl. [Sch01, S. 298])

da dieser die nahe liegendere Übersetzung für den Begriff Certification Authority ist.

3.1.0.3 Registrierungsinstanzen

sind eine optionale Komponente eines Trust Centers. Sie übernehmen Aufgaben mit eher administrativem Charakter. Registrierungsinstanzen nehmen Zertifizierungsanträge entgegen, prüfen Anträge und Identität der Antragsteller und leiten sie weiter an die Zertifizierunginstanz.

Die Registrierungsinstanz muss die Identität des Antragstellers auf geeignete Weise überprüfen. Gemäs §3 der Signaturverordnung [5] z.B. durch Personalausweis, Reisepass.

Bei der geteilten Wahrnehmung der Augaben des *Trust Centers* kann man die Registrierungsinstanz als „Frontoffice“und die Zertifizierungsinstanz als „Backoffice“bezeichnen.

3.1.0.4 Verzeichnisdienste

Zertifikateserver sind in einer Pubulic-Key-Infrastruktur für die Veröffentlichung der ausgestellten Zertifikate zuständig. Hierzu werden Zertifikate nach der Erstellung durch die Zertifizierungsinstanz an den Verzeichnisdienst übermittelt und dort abgelegt.

Weitere Komponenten

Über die genannten Komponenten hinaus können in einer Public-Key-Infrastruktur noch Validierungsdienste existieren. Ebenso sind häufig Zertifizierungsrichtlinien (Certificate Policies) vorhanden, die Vorschriften über Abläufe und sicherheitsmaßnahmen enthalten. Beides ist wird jedoch im Plugin nicht implementiert.

3.2 Datenobjekte in einer X.509-PKI

In einer Public-Key-Infrastruktur werden zwischen den einzelnen Beteiligten verschiedene Daten ausgetauscht. Die Beschreibung dieser Datenobjekte erfolgt in

[5] http://www.gesetze-im-internet.de/sigv_2001/index.html

diesem Abschnitt.

3.2.1 X.509-Zertifikate

Das Kernelement einer Public-Key-Infrastruktur stellen die Zertifikate dar. X.509-Zertifikate sind u.a. in RFC 5280[6] beschrieben. Sie können für Personen, Computersysteme oder Personengruppen ausgestellt werden. Zur Zeit existieren dre Versionen von X.509-Zertifikate, die aufeinander aufbauen. Die erste Version wurde im Jahr 1988 veröffentlicht und enthält die folgenden Angaben:

Feld	Beschreibung
Version	Versionsnummer des X.509-Zertifikats
Seriennummer	eindeutig in CA
Signaturalgorithmus	
Name des Ausstellers	Name der Zertifizierungsinstanz
Gültigkeitszeitraums	
Name des Inhabers	

Im Jahr 1993 wurde der Standard durch X.509v2-Zertifikate erweitert. Hinzugefügt wurden die Felder „subject unique identifier“und „issuer unique identifier“. Diese Ergänzung erwies sich jedoch schnell als unzureichend, so dass schon 1996 in der dritten Version als flexiblere Lösung die Ergänzung durch Zertifikatserweiterungen ermöglicht wurde.

Syntax Ein X.509-Zertifikat ist eine aus drei Objekten bestehende ASN1-Sequence[7]:

```
Certificate ::= { SEQUENCE
     tbsCertificate      TBSCertificate,
     signatureAlgorithm  AlgorithmIdentifier,
     signatureValue      BIT STRING }
```

Das Objekt `tbsCertificate` vom Typ TBSCertificate und enthält die eigentlichen Nutzdaten des Zertifikats – diese werden mit dem Signierschlüssel des Ausstellers signiert. Der object identifier des verwendeten Signaturalgorithmus wird in `signatureAlgorithm` abgelegt, während `signatureValue` die Signatur enthält.

[6] vgl. [C+08]
[7] vgl. [C+08, S. 16ff.]

TBSCertificate ist wiederum eine ASN1-Sequence, die aus bis zu 10 Objekten besteht. Bei X.509v2-Zertifikaten fehlt das Objekt `extensions`, bei X.509v1 fehlen zusätzlich noch `issuerUniqueID` und `subjectUniqueID`:

```
    TBSCertificate ::= { SEQUENCE
         version               [0] EXPLICIT Version DEFAULT v1,
 serialNumber            CertificateSerialNumber,
         signature    AlgorithmIdentifier,
         issuer            Name,
         validity    Validity,
         subject    Name,
         subjectPublicKeyInfo    SubjectPublicKeyInfo,
         issuerUniqueID        [1] IMPLICIT UniqueIdentifier optional,
         subjectUniqueID       [2] IMPLICIT UniqueIdentifier optional,
         extensions            [3] EXPLICIT Extensions optional }
```

Das getaggte Feld `version` enthält einen der drei Integer-Werte 0 (=X.509v1), 1 (=X.509v2) oder 2(=X.509v3), standardmäßig[8] ist es mit 0 belegt. `serialNumber` enthält die Seriennummer des Zertifikats, die eindeutig sein muss für alle Zertifikate, die eine Zertifizierungsinstanz ausgestellt hat. Der object identifier des verwendeten Signaturalgorithmus ist nochmals im `sginature` abgelegt. Dieser muss identisch sein mit dem im Feld `signetureAlgorithm` hinterlegten Wert. Der Name der Instanz, die das Zertifikat ausgestellt hat, ist als ASN1-Sequenz von Relative Distinguished Names im Feld `issuer`, ebenso ist die Information zum Zertifikatinhaber in `subject` hinterlegt. Die Zertifikatsgültigkeit (`validity`) ist die Sequenz der Werte Gültigkeitsbeginn (*notBefore*) und Gültigkeitsende (*notAfter*). Im Feld `extensions` sind schliesslich die Zertifikatserweiterungen enthalten – diese werden im nächsten abschnitt beschrieben.

Wurzelzertifikat Ein Wurzelzertifikat (Root-Zertifikat, Stammzertifikat) ist ein Zertifikat, bei dem Aussteller und Inhaber identisch sind. Es wird in der Wurzel einer hierarchischen Public-Key-Infrastruktur verwendet, alle Zertifikate innerhalb der PKI lassen sich auf dieses Zertifikat zurückführen. Ein Wurzelzertifikat ist meistens vom Inhaber selbstsigniert.

[8] Hat das Feld den Standardwert, so wird es nicht in die Struktur kodiert.

3.2.2 X.509-Zertifikaterweiterungen (Extensions)

Zertifikatserweiterungen sind eine flexible Ergänzung in X.509v3-Zertifikaten. Da sie jedoch auch in Zertifikatsanforderungen und in Zertifikatsperrlisten Verwendung finden, wird ihnen ein eigener Abschnitt gewidmet. Zertifikatserweiterungen bestehen aus einem Tripel ErweiterungsID, critical-Flag und Erweiterungswert[9]. Die ErweiterungsID identifiziert eine spezielle Erweiterung eindeutig. Mit dem critical-Flag wird zwischen kritischen und unkritischen Erweiterungen unterschieden, was sich auf die Behandlung von Zertifikaten auswirkt, die dem Validierer unbekannte Erweiterungen enthalten: Bekannte Erweiterungen müssen – unabängig von ihrer Einstufung als kritisch bzw. unkritisch – immer verarbeitet werden. Bei unbekannten unkritischen Erweiterungen können diese ignoriert und die Prüfung des Zertifikats fortgesetzt werden. Existieren unbekannte Erweiterungen, die als kritisch markiert sind, so muss die Zertifikatsprüfung abgebrochen werden – das Zertifikat ist als ungültig einzustufen.
Durch die Möglichkeit der Definition eigener Zertifikatserweiterungen, bieten sie eine sehr flexible Möglichkeit zum Anfügen von Attributen zu X.509-Zertifikaten. Die definierten[10] Zertifikatserweiterungen lassen sich in verschiedene Kategorien[11] einteilen:

Key and policy information Diese Erweiterungen stellen zusätzliche Informationen zu den in den Zerrtifikaten enthaltenen Schlüsseln und den angewandten Zertifizierungsrichtlinien an.

Authority key identifier

Subject key identifier

Key usage gibt den Zweck an, für den der Schlüssel genutzt werden darf. Sie kann einen oder mehrere der folgenden Werte haben: digitalSignature, nonRepudiation (bzw. contentCommitment), keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, decipherOnly. Die Erweiterung sollte als kritische gekennzeichnet werden.

Extenden key usage

[9](vgl. [**?**, S. 17])
[10]vgl. [C+08, S.26-55]
[11]vgl. [ITU05, S.18ff]

Private key usage period

Certificate policies

Policy mappings

Subject and issuer attributes bieten die Möglichkeit zuätzliche Attribute zum Aussteller oder Inhaber eines Zertifikats anzugeben.

Subject alternative name

Issuer alternative name

Subject directory attribues

CertificationPathConstraints definieren Einschränkungen, die bei der Validierung des Zertifizierungspfads beachtet werden müssen.

Basic constraints geben an, ob das Zertifikat grundsätzlich zur Ausstellung weiterer Zertifikate geeignet (Attribut: cA) ist. Darüber hinaus wird optional die maximale Länge des Zertifizierungspfads angegeben (Attribut: pathLenConstraint). Die Erweiterung ist kritisch.

Name constraints schränken den Namensraum ein, für den Zertifikate ausgestellt werden können.

Policy constraints

Inhibit any policy

CRL distribution points and delta-CRLs unterstützen den Zugriff auf Zertifikatsperrlisten

CRL distribution points

3.2.3 Zertifikatsanforderung

Die Beantragung eines X.509 Zertifikats bei einer Zertifizierungsstelle erfolgt mittels einer Zertifikatsanforderung[12]. Diese kann durch den Antragsteller auf elek-

[12] auch: Zertifikatregistrierungsanfrage, certificate signing request, certification request

tronischem oder nicht-elektronischem Weg eingereicht werden. Zertifikatsanforderungen sind in RFC 2986[13] definiert. Sie enthalten Informationen, die die Zertifizierungsinstanz zur Ausstellung des Zertifikats benötigt. Mindestens sind Angaben zum Zertifikatsinhaber und der öffentliche Schlüssel erforderlich. Optional können in Attributen noch ein Passwort (challengePassword) zur Zurückziehung des Zertifikats und Zertifikatserweiterungen, die beantragt werden sollen, angegeben werden.

Syntax Eine Zertifikatsanforderung sollte im Format `CertificationRequest` vorliegen. Dies ist eine ASN1-Sequenz, die aus drei Elementen besteht:

```
CertificationRequest ::= { SEQUENCE
  certificationRequestInfo    CertificationRequestInfo,
  signatureAlgorithm          AlgorithmIdentifier{{ SignatureAlgorithms
  signatureValue              BIT STRING }
```

Das Feld `certificationRequestInfo` beinhaltet die eigentlichen Daten der Zertifizierungsanforderung. Sie wird mit dem zum öffentlichen Schlüssel gehörenden privaten Schlüssel signiert unter Verwendung des in `signatureAlgorithm` angebenen Signaturverfahrens signiert. Die Signatur selbst ist als BIT STRING im Feld `signature` abgelegt.

```
CertificationRequestInfo ::= { SEQUENCE
  version  INTEGER { v1(0) } (v1, ...),
  subject  Name,
  subjectPKInfo  SubjectPublicKeyInfo{{ PKInfoalgorithms }},
  attributes  Attributes{{ CRIAttributes }},}
```

Das Feld `version` enthält die Versionsnummer der Anforderung, es ist für künftige Zwecke vorgesehen – zur Zeit existiert nur eine Version. Dieses Feld ist nicht mit dem gleichnamigen Feld eines X.509-Zertifikats zu verwechseln. Der distinguished name des Antragstellers ist in `subject` enthalten, deröffentliche Schlüssel in `attributes`. Das Feld `attributes` kann neben dem challengePassword auch beantragte Zertifikatserweiterungen enthalten.

[13] sh. [] (wurde durch RFC5967 aktualisiert, aber ohne Auswirkungen auf den betrachteten Bereich)

3.2.4 Zertifikatssperrlisten

Aus verschiedenen Gründen kann es dazu kommen, dass ein Schlüsselpaar vor Ablauf der Gültigkeitsperiode des dazugehörenden Zertifikats nicht mehr genutzt werden darf, beispielsweise weil der private Schlüssel bekannt geworden ist. Hierüber müssen alle Zertifikatanwender unterrichtet werden, damit sie künftig den öffentlichen Schlüssel nicht mehr nutzen.
Der Eigentümer beantragt die Sperrung des Zertifikats unter Angabe des Grunds für die Sperre bei der ausstellenden Zertifizierungsinstanz. Nach Prüfung des Antrags wird das Zertifikat gesperrt. Hierfür gibt es mehrere Möglichkeiten. Sperrliste oder bei Nutzung eines Validierungsdienstes über OCSP oder SCVP

Die Veröffentlichung der gesperrten Zertifikate erfolgt über die Zertifikatssperrliste (Certificate Revocation List). Die Liste ist eine sogenannte „Schwarzliste“, d.h., alle auf der Liste enthaltenen Zertifikate können nicht mehr verwendet werden. Die Liste gibt die Situation zum Zeitpunkt ihrer Erstellung wieder, d.h. die benutzte Sperrliste sollte immer so aktuell wie möglich sein.

3.2.5 Certificate Management Protocol Messages

Das Certificate Management Protocol – beschrieben in RFC 4210 (09/2005) – dient der Kommunikation der Komponenten einer PKI untereinander. Für verschiedene Kommunikationsanlässe sind Nachrichten vorgesehen. Für den Kontext relevant ist allerdings nur das *CertificationRequest*, es wird im entwickelten Plugin zur Weiterleitung der von der Registrierungsinstanz geprüften Zertifikatsanforderungen an die jeweilige Zertifizierunginstanz genutzt. Eine PKIMessage (Nachricht des Certificate Management Protocols) besteht aus einer Sequenz von bis zu 4 Objekten:

```
PKIMessage ::= SEQUENCE {
    header PKIHeader,
    body PKIBody,
    protection    [0] PKIProtection OPTIONAL,
    extraCerts    [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate OPTIONAL
}
```

Das Feld `header` enthält als Pflichtbestandteile die Version der CMP-Nachricht, sowie die Angabe des Senders und Empfängers. Der PKIBody enthält immer ge-

nau eine Nachricht als ASN1-getaggtes Objekt, zum Beispiel das schon erwähnte CertificationRequest mit dem Tagwert 4.

3.3 Vorgänge

Innerhalb der einzelnen Komponenten einer Public-Key-Infrastruktur sind verschiedene Vorgänge erforderlich, damit Zertifikatinhaber Zertifikate erhalten und Zertifikatnutzer Zertifikate überprüfen können. Die einzelnen Abläufe werden in diesem Abschnitt beschrieben.

3.3.1 Zertifikatsanforderung erstellen

Endteilnehmer (Zertifikatinhaber) und Zertifizierungsstellen benötigen Zertifikate für ihre privaten Schlüssel. Zur Erstellung einer Zertifikatsanforderung führt der Antragsteller folgende Schritte aus:

1. Auswahl eines asymmetrischen Schlüsselpaares
2. Festlegung Distinguished Name
3. Festlegung der zu beantragenden Zertifikatserweiterungen
4. Auswahl des Signaturalgorithmus und Signieren der Zertifikatregistrierungsanforderung
5. Übertragen der Zertifikatregistrierungsanforderung an die entsprechende Zertifizierungsstelle

Schlüsselerzeugung in der Zertifizierungsinstanz Das zu zertifizierende Schlüsselpaar kann auch in der Zertifizierungsinstanz erzeugt werden. Vorteile hierbei sind: Schlüsselwiederherstellung, ist dem Zertifikatsinhaber der private Schlüssel abhanden gekommen[14], kann er durch die Zertifizierungsinstanz wiederhergestellt werden Einheitliche Qualität der Schlüssel (im Hinblick auf Algorithmen und Stärke), Benutzer brauchen Schlüsselpaar nicht selbst erzeugen (Laien). Als Nachteil wären ein erhöhter Aufwand in der Zertifizierungsinstanz

[14]Voraussetzung ist allerdings, das kein Dritter den Schlüssel „gefunden“hat.

zu nennen, zum einen durch das Erzeugen der Schlüsselpaare und durch den notwendigen gesicherten Transport. Darüberhinaus ist ein besonderes Vertrauen des Benutzers in die Zertifizierungsinstanz nötig, da diese ja den privaten Schlüssel kennt.

3.3.2 Zertifikat für einen öffentlichen Schlüssel ausstellen

Nach Eingang einer Zertifikatsanforderung Durchführende: Zertifizierungsstellen Ablauf, Vorbedingung, Nachbedingung, ...

1. Überprüfen der Signatur der Zertifikatsanforderung
2. Überprüfen der Identität des Antragstellers
3. Anpassen der beantragten Zertifikatserweiterungen
4. Auswahl eines Gültigkeitszeitraumes für das Zertifikat
5. Auswahl des privaten Schlüssels zum Signieren des Zertifikats
6. Auswahl des Signaturalgorithmus und Signieren des Zertifikats
7. Übertragen des Zertifikats an den Antragsteller
8. Übertragen des Zertifikats an den Verzeichnisdienst

Die ersten beiden Punkte entfallen in der Zertifizierungsinstanz, falls die Anforderung vorher bereits in einer Registrierungsinstanz überprüft wurde.

3.3.3 Anfrage Verzeichnisdienst

Verfügt der Prüfer einer Signatur nicht über alle erforderlichen Zertifikate zur Kondtruktion des Zertifizierungspfads, so kann er eine Anfrage beim Verzeichnisdienst starten und die Zertifikate, falls vorhanden, von dort abrufen.

3.4 Überprüfung von Signaturen

Signaturen dienen dazu die Echtheit und Integrität von Dokumenten zu bestätigen. Eine Signatur ist jedoch wertlos, solange ihre Gültigkeit nicht verifiziert wurde. Hierzu muss festgestellt werden, ob

1. die Signatur für das vorgelegte Dokument erstellt wurde bzw. dass das Dokument nach dem Signiervorgang nicht mehr verändert wurde,
2. das zum Signierschlüssel gehördende Zertifikat zu einem bestimmten Zeitpunkt zeitlich gültig war oder ist (vgl. der folgende Abschnitt „Gültigkeitsmodelle “
3. das Zertifikat zu diesem Zeitpunkt nicht revoziert oder gesperrt ist,
4. das Zertifikat zu einem vom Prüfer[15] als vertrauenswürdig eingestuften Punkt zurückgeführt werden kann und
5. alle Richtlinien innerhalb der Public-Key-Infrasruktur eingehalten wurden.

Zur Überprüfung von Punkt 1 sind für den Prüfer das vorgelegte Dokument mit Signatur und das Zertifikat zum Signierschlüssel ausreichend. Zur Prüfung der weiteren Punkte, wird in Public-Key-Infrastrukturen nach X.509 der Zertifizierungspfad validiert. Dieser Prozess verläuft in zwei Phasen, zunächst die Konstruktion des Zertifizierungspfads und anschließend – falls die Konstruktion erfolgreich war – die eigentliche Validierung des Zertifizierungspfads.

3.4.1 Konstruktion des Zertifizierungspfads

In dieser Phase wird eine Kette von Zertifikaten gesucht, die das zu validierende Zertifikat mit einem vertrauenswürdigen Zertifikat – dies kann zum Beispiel das Wurzelzertifikat der eigenen PKI sein, aber auch ein anderes vom Anwender als vertrauenswürdig eingestuftes Zertifikat. Im einfachsten Fall verfügt der Validierer[16] über alle Zertifikate, die zur Konstruktion des Zertifizierungspfads erforderlich sind.

Die konstruktion des Zertifizierungspfads ist abgeschlossen, sobald ein vom Validierer als vertrauenswürdig eingestuftes Zertifikat erreicht ist – die ist mindestens das Wurzelzertifikat der eigenen PKI, jedoch können auch andere vom Prüfer als vertrausenswürdig eingestuften Zertifikate sein.

In einem gültigen Zertifizierungspfad darf ein Zertifikat nur einmal vorkommen.

[15] ein Anwender oder eine Anwendung, der/die das Zertifikat prüft
[16] zum Beispiel ein Benutzer oder eine Anwendung

3.4.2 Validierung des Zertifizierungspfads

Wurde ein Zertifizierungspfad vom Zertifikat des Signaturerstellers bis zu einem Vertrauensanker gefunden, wird nun überprüft, ob dieser Pfad in gültig ist.

Gültigkeitsmodelle Zur Überprüfung, ob eine Signatur gültig ist, existieren verschiedene Gültigkeitsmodelle[17], die sich hinsichtlich des Aufwands der Überprüfung sowie der Gültigkeitsbeschränkung der Zertifikate unterscheiden. Zur Kennzeichnung, welches Gültigkeitsmodell bei der Zertifikasprüfung verwendet werden muss, gibt es die private Zertifikatserweiterung *ValidityModel*[18]

Schalenmodell Das Schalenmodell ist die einfachste Variante. Damit eine Signatur gültig ist, müssen zum Verifikationszeitpunkt alle Zertifikate im Zertifizierungspfad gültig sein. Dies hat zur Folge, dass nur dann im ganzen Gültigkeitszeitraum des Zertifikats gültige Signaturen erstellt werden können, wenn alle übergeordneten Zertifikate mindestens den gleichen Gültigkeitszeitraum wie das fragliche Zertifikat haben. Eine zweite Folge ist, dass Signaturen nie über den Gültigkeitszeitraums des Zertifikats hinaus positiv verifiziert werden können. Der große Vorteil des Schalenmodells liegt in der einfachen Verifikation und an der Tatsache, dass alle hierfür erforderlichen Daten in den Zertifikaten bereits enthalten sind. Von Nachteil ist, dass insbesondere das Wurzelzertifikat und die Zertifizierungsstellenzertifikate sehr lange gültig sein müssen.

Kettenmodell Das Kettenmodell hat einen anderen Ansatz. Es betrachtet nicht den Verifikationszeitpunkt, sondern den Zeitpunkt der Erzeugung einer Signatur oder eines Zertifikats. Nur zu diesem Zeitpunkt muss das Zertifikat zum Signierschlüssel gültig sein. Die hat zur Folge, dass eine große Unabhängigkeit in der Gültigkeit von übergeordneten zertifikaten besteht. Zertifikate von Wurzelinstanzen und Zertifizierungsstellen können mit kürzeren Laufzeiten auskommen, ohne dass dies Auswirkungen auf die Gültigkeit der ausgestellten Zertifikate hat. Großer Nachteil beim Kettenmodell ist die Tatsache, dass der Ausstellungszeitpunkt auf geeignete Weise festgehalten werden muss und die Verifikation kom-

[17] vgl. [Sch07, S.514ff]

[18] vgl. [Rap09, S.22] und http://www.cdc.informatik.tu-darmstadt.de/TI/Forschung/FlexiPKI/validitymodel/index.html

plizierter ist. Trotzdem ist es heute am verbreitesten und von vielen Signaturgesetzen gefordert[19]

Modifiziertes Schalenmodell Das modifizierte Schalenmodell ist etwas flexibler, wie das Schalenmodell. Hier wird – wie auch beim Kettenmodell – der Erzeugungszeitpunkt einer Signatur zur Prüfung herangezogen. Zu diesem Zeitpunkt müssen jedoch – wie beim Schalenmodell – aller übergeordneten Zertifikate gültig sein. Damit bietet dieses Modell die Möglichkeit Signaturen zu erzeugen, die auch über den Gültigkeitszeitraum des Zertifkats hinaus gültig sind.

3.5 Verbindung von Public-Key-Infrastrukturen

Innerhalb einer Public-Key-Infrastrukturen besteht eine Vertrauenshierarchie – jedes Zertifikat kann auf das Wurzelzertifikat zurückgeführt werden. Ein Problem tritt allerdings auf, wenn Kommunikation über die Grenzen einer PKI hinaus stattfinden soll – hier gibt es keinen gemeinsamen Vertrauensanker mehr.
Eine Lösung bietet die Verknüpfung von Public-Key-Infrastrukturen durch die Ausstellung von Crosszertifikaten[20].

3.5.1 Crosszertifizierung

Ein Crosszertifikat ist ein Zertifikat, dass für einen öffentlichen Schlüssel ausgestellt wird, für den bereits ein Zertifikat durch eine andere Zertifizierunginstanz ausgestellt wurde[21].

3.5.2 Bridge-CA

Eine besondere Variante der Crosszertifizierung bietet der Einsatz einer Bridge-CA. Die Zertifizierungsinstanzen der zu verknüpfenden PKIen stellen sich nun nicht mehr gegenseitig Crosszertifikate aus, sondern führen alle eine Crosszertifizierung mit der Bridge-CA durch. Die Bridge-CA ist eine dritte Stelle, die unabhängig von den anderen Public-Key-Infrastrukturen ist.

[19] vgl. [Sch07, S.514]
[20] vgl. [Ham01]
[21] vgl. [Ham01]

Probleme bei der Verbindung von Public-Key-Infrastrukturen können u.a. auftreten durch verschiedene Zertifizierungsrichtlinien in den beteiligten PKIs (v.a. wenn sie nicht vergleichbar sind), durch Änderung der vorgesehenen maximalen Länge des Zertifizierungspfads – dadurch kann es zu Problemen bei der Validierung kommen, falls der Wert für PathLen in den Basic Constraints nicht mehr ausreicht – sowie durch unterschiedliche Berechnungsmethoden für Zertifikatserweiterungen, die zur Identifizierung genutzt werden, z.B. *Subject Key Identifier* und *Authority Key Identifier*.

Kapitel 4

Pluginentwicklung

Diese Kapitel behandelt das Kernstück der Arbeit – die eigentliche Entwicklung des JCrypTool-Plugins. Zunächst wird allgemein die Erweiterung von RCP-Anwendungen und im Speziellen die Entwicklung von JCrypTool-Plugins beschrieben. Im Anschluss erfolgt der Entwurf eines Modells für die Simulation von Public-Key-Infrastrukturen nach X.509. Hieraus wird schliesslich das konkrete Plugin entworfen und implementiert. Das Kapitel schliesst mit einer Zusammenfassung der Anwendungsmöglichkeiten des neu entwickelten Plugins.

4.1 Vorarbeiten

4.1.1 Anforderungen aus der Aufgabenstellung

Die der Arbeit zu Grunde liegende Aufgabenstellung umfasst allgemein die Simulation von Public-Key-Infrastrukturen gemäß X.509. Spezielle zu betrachtende Punkte sind:

- Erstellung eines PKI-Zertifikats / Format eines PKI-Zertifikats
- Zertifizierung durch eine Zertifizierungsinstanz mit CA und RA
- Zertifizierungsbaum (nach X.509)
- Crosszertifizierung bei X.509 und Zertifizierungsnetz (nach X.509)
- Ketten- und Schalen-Gültigkeitsmodelle

4.1.2 Entwicklung von JCrypTool-Plugins

JCrypTool ist eine auf der Eclipse RCP basierende Anwendung. Die Rich-Client-Platform wurde mit Eclipse 3.0 eingeführt. Sämtliche Funktionalität wird in Plugins bereitgestellt und als Erweiterung an Erweiterungspunkten eingebunden. Zur Entwicklung eigener Plugins bietet JCrypTool eine Reihe von Erweiterungspunkten an. Informationen zur Erweiterung von JCrypTool wurden „Getting started with JCrypTool“[1] entnommen. Zur Einarbeitung in die Thematik *Plugin-Entwicklung für RCP-Anwendungen* wurde [Dau06] genutzt.

4.1.3 Kryptographie mit Java

Zur Einarbeitung in die Thematik *Kryptrographie mit Java* im allgemeinen und die Verwendung der Bouncycastle-Bibliothek im speziellen wurden neben Internetrecherche [LFB+00] und [Hoo05] genutzt.

4.2 Modell einer X.509-Public-Key-Infratruktur

Public-Key-Infrastrukturen nach X.509 stellen eine umfangreich Thematik mit einer Vielzahl von spezifizierten Datenobjekten dar. Aus diesem Grund wurde für die Implementierung des Lernprogramms wurde ein Modell definiert. Ein Modell ist ein Abbild der Wirklichkeit, dass nur die im jeweiligen Kontext relevanten Aspekte enthält. Grundlage des Modells sind die inhaltlichen Anforderungen aus der Aufganebstellung, die an das Programm gestellt werden. Während der Erarbeitung der Thematik Public-Key-Infrastrukturen nach X.509 wurde das Modell angepasst, um möglichst authentisch zu sein. Im Folgenden soll das Modell als Simulationsumgebung bezeichnet werden.

Anforderungen an die Simulationsumgebung

Die Anforderungen an das Modell ergeben sich aus den Anforderungen an das Programm aus Kapitel **??**, den Erkenntnissen aus der Aufarbeitung der kryptographischen Grundlagen, sowie praktischen Erwägungen. Die Anforderungen an das Modell setzen sich zusammen aus den verschiedenen Entitäten, die existieren

[1] [JCT10]

sollen, den Datenobjekten, die zwischen ihnen ausgetauscht werden sollen und den Aktionen, die die Entitäten ausführen können sollen.

Entitäten

In der Simulationsumgebung sollen die folgenden Entitäten existieren:

- Benutzer (realisieren gleichzeitig Zertifikatsinhaber und -anwender)
- Zertifizierungsinstanzen
- Registrierungsinstanzen
- Verzeichnisdienste

Zertifizierungsinstanzen sollen in den Varianten *Wurzelinstanz*, *Bridge-CA* und *Zertifizierungsinstanz mit integrierter Registrierungsinstanz* realisiert werden.

Datenobjekte

Die folgenden Datenobjekte sollen in der Simulationaumgebung realisiert werden:

- X.509-Zertifikate
- Zertifikatsanforderungen nach PKCS#10
- Nachrichten entsprechend dem certificate management protocol (tlw.)
- Zertifikatsperrlisten

Zertifikatserweiterungen Aus der Menge der definierten Zertifikatserweiterungen[2] wurden zunächst einige zur Simulation ausgewählt. Im Laufe der Entwicklung wurde jedoch diese jedoch nicht weiter verfolgt.

[2] vgl. [C+08, S.26-55] Kap. 4.2

Aktionsmöglichkeiten

Im Modell sollen die jeweiligen Entitäten (in der Liste in () notiert) die folgenden Aktionen ausführen können:

- Beantragen von Zertifikaten mittels PKCS#10-Anforderung bei einer Zertifizierungsinstanz (*Benutzer, Zertifizierungsinstanzen*)
- Prüfen einer PKCS#10-Zertifikatsanforderung (*Zertifizierungsinstanzen, Registrierungsinstanzen*)
- Weiterleiten einer geprüften zertifikatsanforderung (*Registrierungsinstanzen*)
- Ausstellen eines X.509-Zertifikats (*Zertifizierungsinstanzen*)
- Validierung des Zertifizierungspfads in Verbindung mit der Visualisierung verschiedener Gültigkeitsmodelle
- Abruf von Zertifikaten aus dem Verzeichnisdienstsicht

4.2.1 Sonstige Anforderungen

Die Simulationsumgebung soll ausserdem folgende Merkmale erfüllen:

- Um die Verbindung von Public-Key-Infrastrukturen mittels Crosszertifizierung simulieren zu können, muss die das Modell mehrere Public-Key-Infrastrukturen enthalten können.
- Um verschiedene Situationen darzustellen und zur Verbesserung der Übersichtlichkeit soll die Möglichkeit der Verwendung unterschiedlicher Simulationsumgebungen bestehen.
- Zur Veranschaulichung der Zertifikatsgültigkeiten zu unterscheidelichen Zeitpunkten, sollte die aktuelle Zeit – somit der Zeitpunkt, zu dem Zertifikate überprüft werden – veränderbar sein. Dies soll auf Ebene der Simulationaumgebung geschehen.
- Um besser hinter die Struktur der Simulationsumgebung blicken zu können, sollen alle Daten möglichst auch ohne das entwickelte Programm betrachtet werden können.

Kapitel 5

Implementierung

Der folgenden Abschnitt behandelt die konkrete Implementierung des Plugins und ist unterteilt in die Realisierung der Simulationsumgebung, die Gestaltung der Benutzeroberfläche und die Entwicklung der Wizards.

Einbindung in JCrypTool

Erweiterungspunkte Die Einbindung des Plugins in JCrypTool erfolgt über die folgenden Erweiterungspunkte:

org.jcryptool.core.operations.visuals Hier erfolgt die Erweiterung mit Plugins, die in die Kategorie „Visualisierungen“ eingeordnet werden. diese Erweiterung sorgt unter anderem dafür, dass das Plugin in die entsprechende Menüstruktur aufgenommen wird.

org.eclipse.ui.views ist für die Erweiterung der Sichten zuständig. Die Sicht ist die oberste Instanz der Benutzeroberfläche des Plugins.

org.eclipse.ui.cheatsheets.cheatSheetContent ist der Erweiterungspunkt für Spickzettel

org.eclipse.help.toc erweitert die Hilfe, (realisiert im Anwenderhandbuch)

Benutzerdaten Gemäß den Richtlinien von JCrypTool werden sämtliche Benutzerdaten, die das Plugin speichert, innerhalb des Verzeichnisses *.jcryptool* im aktuellen Benutzerverzeichnis abgelegt. Für das Plugin wird hier ein Unterordner *pki* angelegt.

Zertifikats- und Schlüsselspeicher Der in JCrypTool realisierte *KeyStore* wird bewusst nicht genutzt, um jede Entität der Simulationsumgebung einen separaten Speicherort für Zertifikate und Schlüssel zu ermöglichen.

Verwendete Bibliotheken Als Security-Provider wird *Bouncycastle* verwendet. Diese Bibliotheken enthalten eine große Auswahl an Klassen mit Bezug zu X.509-Public-Key-Infrastrukturen.

Modellzeit

Um während der Arbeit mit der Simulationsumgebung Aktionen zu verschiedenen Zeitpunkten ausführen zu können – z.B. die Überpfügung der Zertifikatsgültigkeit – wird die Möglichkeit gegeben die durch die Simulationsumgebung verwendete Zeit zu verändern. Dies erfolgt durch Eingabe eines gültigen Datums in der Fußzeile der Benutzeroberfläche. Diese wird in der Simulationsumgebung gespeichert und künftige Aktionen finden zu diesem Zeitpunkt statt. Ein Neustart des Plugins oder das Zurücksetzen der Modellzeit bewirken, dass anschliessend wieder die Systemzeit Grundlage der Zeit im Modell ist.

5.1 Entitäten der Simulationsumgebung

Die verschiedenen Entitäten werden jeweils durch einzelne Klassen repräsentiert. Entsprechend dem aufeinander aufbauenden Funktionsumfang stehen die Klassen in einer hierarchischen Beziehung: Die Klasse `User` ist die Oberklasse der verschiedenen Entitätsklassen. Damit die GUI-Objekte über Änderungen am Modell informiert werden können erweitert sie die Klasse `Observable`.

Verzeichnisstruktur Die Organisation der Entitäten der Simulationsumgebung erfolgt über eine besondere Verzeichnisstruktur. Unterhalb des Plugin-Verzeichnisses (*.jcryptool/pki*) existiert für jede Simulationsumgebung ein separates Unterverzeichnis. Hierin reprästentieren wiederum Unterverzeichnisse die jeweiligen Enitäten. Jede Entität verfügt über einen Zertifikats- und Schlüsselspeicher (*Entitätsname.jce*), zwei Verzeichnisse für eingehende Daten (*inbox* für z.B. Zertifikate und Zertifikatsanforderungen, sowie *cmp* für eingehende CMP-Nachrichten.

KeyStore Alle Schlüsselpaare und Zertifikate, die eine Entität der Simulationsumgebung besitzt, werden in einem eigenen KeyStore des Typs `BKS` (Bouncycastle KeyStore) gespeichert. Als Alias wurde bei Zertifikaten der SHA1-Hash des Zertifikats gewählt. Muss ein Schlüsselpaar unabhängig von einem Zertifikat gespeichert werden – zum Beispiel bei einer Zertifikatsanforderung – wird ein Dummy-Zertifikat erzeugt und unter einem Alias der dem SHA1-Hash des öffentlichen schlüssels entspricht. Dies erleichtert bei Import des von der Zertifizierungsstelle ausgestellten Zertifikats ide Zuordnung zum entsprechenden schlüsselpaar.

vertrauenswürdige Zertifikate Für verschiedene Aktionen in der Simulationsumgebung ist es erforderlich, dass Zertifikate als vertrauenswürdig gekennzeichnet werden können. Hierzu verfügt jede Entität über eine Textdatei mit den SHA1-Fingerprints der als vertrauenswürdig markierten Zertifikate. Diese Datei kann problemlos auch per Hand editiert werden.

Public-Key-Infrastrukturen in der Simulationsumgebung werden immer durch ihre Wurzelinstanz repräsentiert. Jeder Entität wird die Wurzelinstanz „zugeordnet“zu deren PKI sie gehört. Dies ist wichtig um Entitäten verschiedener PKIen in der Simulationsumgebung auseinander halten zu können.

5.1.1 Benutzer

werden durch Instanzen der Klasse `User` dargestellt. Da sie die Oberklasse der anderen Entitäten darstellen, gilt die folgende Auflistung von Funktionen auch für diese. Neben Funktionen mit organisatorischem Character z.B. zum Anlegen und Löschen von Benutzern verfügen sie über im Kontext relevanten Funktionen zum Beispiel zum

- Abspeichern von Schlüsselpaaren im KeyStore: `addKeyPairToKS()`
- Exportieren von Schlüsselpaaren oder Zertifikaten in eine Datei: `exportCert2File()` und `exportKeyPair2File()`
- Erzeugen einer Zertifikatsanforderung: `generateCSR()`
- Durchsuchen des KeyStores nach dem Ausstellerzertifikat zu einem vorhandenen Zertifikat: `getIssuerCert()`

- Zur Ermittlung und Veränderung der Vertrauenswürdigkeit von Zertifikaten: `setTrustCertificate()` und `isTrustCertificate`

Zum *Empfang* von Dateien von anderen Entitäten aus der Simulationsumgebung ist die Methode `storeFile()` vorgesehen. Sie liefert dem Sender ein File-Objekt zurück, unter der er die Datei in der *inbox* des Benutzers speichern kann. Diese Funktionalität ist erforderlich für den Transfer von Zertifikatsanforderungen zur Zertifizierungsinstanz und von ausgestellten Zertifikaten zurück zum Antragsteller.

5.1.2 Zertifizierungsinstanzen

werden durch die Klassen `BridgeCA`, `CA`, und `RootCA` realisiert. Zwischen den Klassen besteht eine Hierarchie in der genannten Reihenfolge. Sie verfügen zusätzlich zu den genannten Methoden aus der Klasse `User` über Funktionalität zum

- Ausstellen von Zertifikaten: `createCertificate()` und `issueCertificate()`
- Verwalten der Registrierungsinstanzen mit denen die Zertifizierungsinstanz zusammen arbeitet: `setRAList()` und `getRAList()`
- Erzeugen von Wurzelzertifikaten (nur `RootCA`): `createRootCert`

5.1.3 Registrierungsinstanzen

sind Instanzen der Klasse `RA`. Sie verfügen neben den von `User` geerbten Methoden über keine weiteren Methoden, da ihre Funktionalität in Wizards abgebiL̈det wird – das heisst der Anwender übernimmt alle Funktionen der Registrierungsinstanz.

5.1.4 Verzeichnisdienste

sind Instanzen der Klasse `Dir`. Neben den von `User` geerbten Methoden (v.a. die zur Verwaltung des KeyStores) enthalten sie Methoden zum:

- Eintragen von Zertifikaten in das Verzeichnis: `storeCertificate()`

- Suchen nach Zertifikaten (über Distinguished Names oder Teile davon): `searchCertificates()`
- Rückgabe aller Zertifikate, die im Verzeichnis enthalten sind: `getAllCerts()`
- Rückgabe aller im Verzeichnis enthaltener Zertifikate, zu denen ein als Parameter übergebenen Ausstellerzertifikat gehört: `getChildCerts()`

Die Methoden des Verzeichnisdienstes wurden auf die Erfordernisse der Simulationsumgebung abgestimmt. Vorgesehene Methoden für Zertifikatsperrlisten wurden nicht implementiert, da diese in der Simulationsumgebung noch nicht realisiert sind.

5.2 Benutzeroberfläche

Die Interaktion des Anwenders mit der Simulationsumgebung erfolgt über eine graphische Benutzeroberfläche mit den Registerkarten *Übersicht*, *Benutzer*, *Registrierungsinstanz* und *Zertifizierungsinstanz*. Die Anordnung in dieser Reihenfolge ist bewusst gewähl̈t und soll die Rolle der Registrierungsinstanz zwischen Benutzer und Zertifzierungsinstanz verdeutlichen.

Fußzeile Unterhalb der Registerkarten befindet sich die Fußzeile mit Zugriff auf von den einzelnen Entitäten unabhängigen Funktionen:

- Hinzufügen, Löschen und Auswählen der aktiven Simulationsumgebung
- Verändern und Zurücksetzen des Datums im Modell
- Aufruf eines separaten Dialog zur Visualisierung von Gültigkeitsmodellen

5.2.1 Registerkarte *Benutzer*

Abbildung 5.1 stellt das Plugin mit geöffneter Registerkarte „Benutzer“dar. Sie entspricht der Sicht des Benutzers auf die Simulationsumgebung. Im linken Bereich finden sich die Optionen zur Auswahl des aktiven Benutzers, sowie zum Hinzufügen und Löschen von Benutzern.

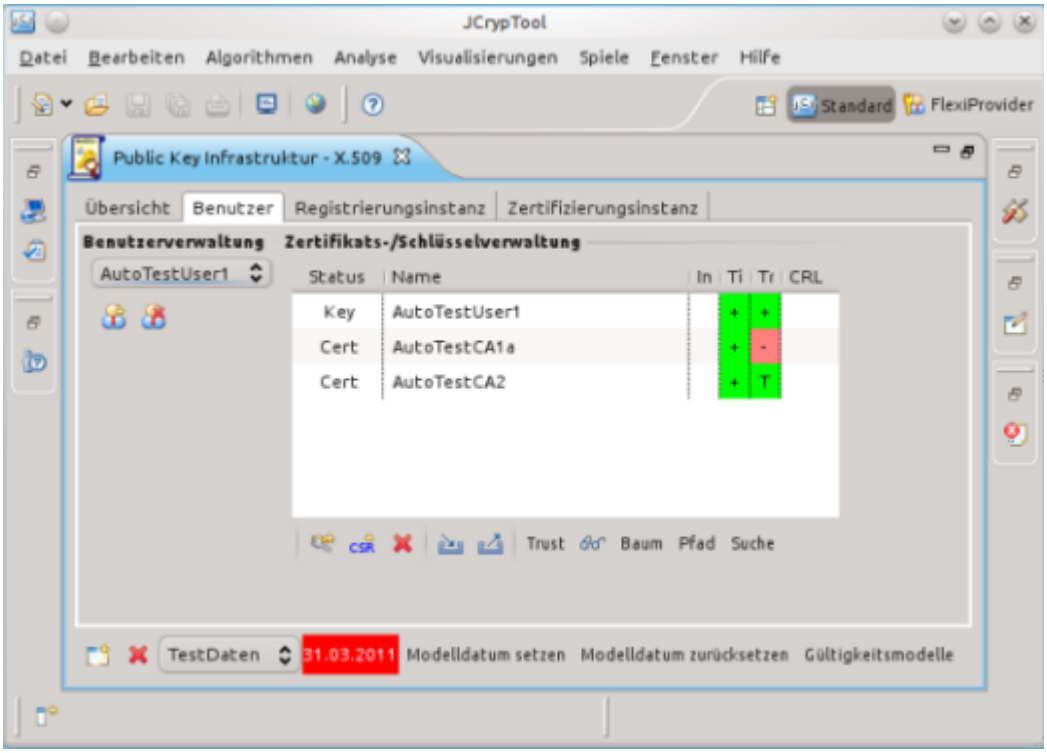

Abbildung 5.1: Benutzersicht

Zertifikats- und Schlüsselliste Zentrales Element der Benutzersicht ist die Zertifikats- und Schlüsselliste. Im ihr werden alle im KeyStore des Benutzers abgelegten Schlüssel und Zertifikate angezeigt. In der zweiten Spalte von rechts wird die Vertrauenswürdigkeit des Zertifikats angegeben. Die dritte Spalte von rechts zeigt die aktuelle Gültigkeit an. Die rechte Spalte ist vorgesehen zur Anzeige, ob das Zertifikat gesperrt wurde.
Mit der Toolbar unterhalb der Liste kann der Benutzer verschiedene Aktionen starten. Von links nach rechts stehen ihm folgende Möglichkeiten zur Verfügung:

- Generierung eines neuen Schlüsselpaares
- Erstellen einer Zertifikatregistrierungsanforderung
- Löschen eines Schlüsselpaares/Zertifikats
- Import eines Zertifikats
- Export eines Schlüsselpaares/Zertifikats

- Festlegen der Vertrauenswürdigkeit eines Zertifikats
- Ansicht eines Zertifikats
- Darstellung des Zertifizierungsbaums der PKI des Benutzers
- Validierung des Zertifizierungspfads für das ausgewählte Zertifikat
- Suche eines Zertifikats im Verzeichnisdienst

5.2.2 Registerkarte *Registrierungsinstanz*

Die Registrierungsinstanz hat in der Simulationsumgebung nur sehr wenige Aktionsmöglichkeiten. Hauptaufgabe ist das Prüfen von Zertifikatsanforderungen inklusive Überprüfung der Identität des Antragstellers (*Zertifizierungsanfrage bearbeiten*). Als zweite Aufgabe war das Weiterleiten von ausgestellten Zertifikaten an den Antragsteller vorgesehen[1](*CMP Nachricht verarbeiten*)

5.2.3 Registerkarte *Zertifizierungsinstanz*

Die Sicht der Zertifizierungsinstanz (siehe Abbildung 5.2) ähnelt stark der Benutzersicht. Im linken Bereich sind zu den organisatorischen Funktionen noch eine Punkt zur Bearbeitung von CMP-Nachrichten und ein weiterer zur Verwaltung der Registrierungsinstanzen, die mit der Zertifizierungsinstanz zusammenarbeiten. In der Sicht einer Wurzelinstanz gibt es hier noch zusätzlich die Option ein Wurzelzertifikat zu erzeugen. Die Zertifikats- und Schlüsselliste ist identisch mit der Benutzersicht, die Toolbar hat zusätzlich noch folgende Funktionen:

- Ausstellen eines Zertifikats
- Initiieren einer Crosszertifizierung

5.3 Aktionen

Zur anschauliche Visualisierung von Abäufen sind Wizards gut geeignet. Der Anwender wird Schritt für Schritt durch die Aufgabe geleitet – darüber hinaus besteht die Möglichkeit auf den Wizardseiten Erläuterungen anzugeben.

[1]Ist momentan nicht aktiviert, da Zertifikate direkt von der Zertifizierungsinstanz an den Antragsteller versandt werden

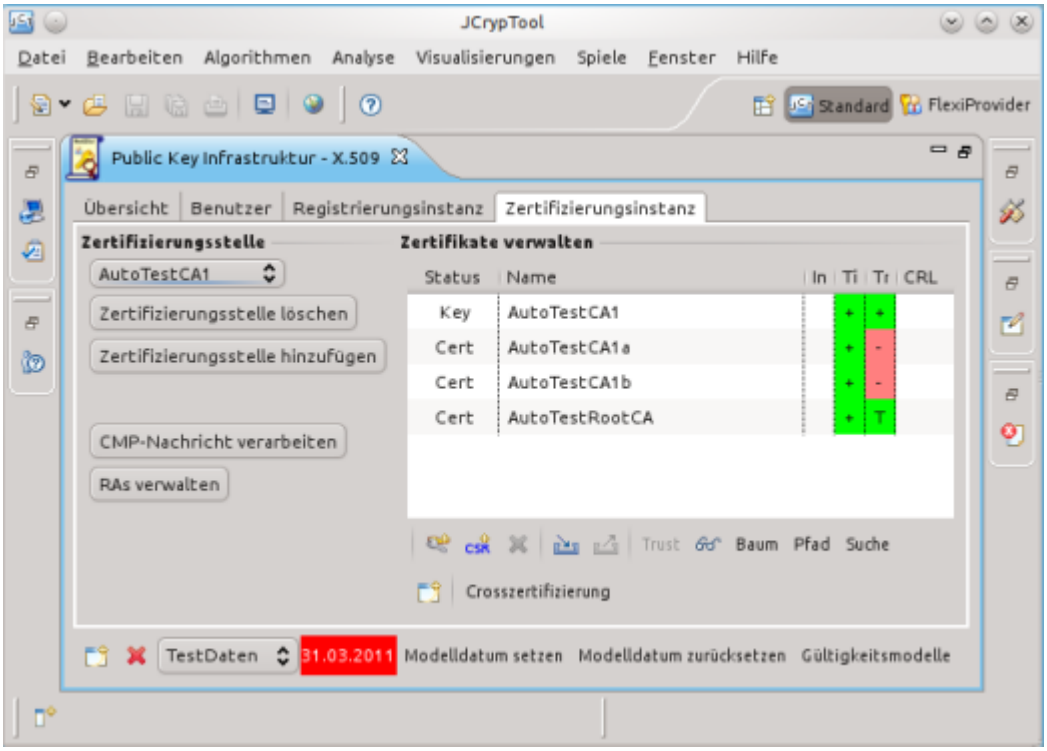

Abbildung 5.2: Benutzersicht

Fast alle in der Simulationsumgebung möglichen Aktionen sind mit JFace-Wizards realisiert, die im Paket `org.jcrypool.visual.pkiplugin.ui.wiz` enthalten. In der Regel wird der Wizard zu Beginn der Aktion mit bestimmten Werten vorbelegt. Im Anschluss erfolgt die Benutzerinteraktion über verschiedene Wizardseiten und schliesslich – abhängig davon, ob neue Daten erzeugt wurden, die gespeichert werden müssen – werden die Berechnungen durchgeführt und die Daten im Modell abgelegt. Im folgenden Abschnitt werden einige der realisierten Aktionen in der Simulationsumgebung dargestellt.

5.3.1 Zertifikatsanforderung

Zertifikatsanforderungen können von den Benutzeroberflächen *Benutzer* und *Zertifizierungsinstanz* ausgelöst werden. Der Wizard `W_ReqCert` führt den Anwender durch die Erstellung einer Zertifikatsanforderung. Nach einer Beschreibung auf der einleitenden Wizardseite können die folgenden Einstellungen vorgenommen werden:

1. Eingabe bzw. Abänderung des Distinguished Name.
2. Auswahl eines Schlüsselpaares. Hier bestehen die Alternativen ein Schlüsselpaar neu zu erstellen, ein Schlüsselpaar aus dem KeyStore auszuwählen oder ein Schlüsselpaar aus einer Datei zu importieren.
3. Eingabe von Zertifikatserweiterungen, die beantragt werden sollen. Zur Zeit wird aber nur die Erweiterung *Basic Constraints* unterstützt.
4. Auswahl eines Signieralgorithmus zum Signieren der Zertifikatsanforderung. Hierbei ist zu beachten, dass ein zur Art des ausgewählten Schlüsselpaares geeigneter Algorithmus gewählt wird.
5. Auswahl des Empfängers der Anforderung. In einer ersten Auswahlliste kann die gewünschte Zertifizierungsinstanz ausgewählt werden, in einer zweiten Liste eine entsprechende Registrierungsinstanz (oder nochmals de Zertifizierungsinstanz, falls sie gleichzeitig als Registrierungsinstanz fungiert). Alternativ besteht die Möglichkeit die Anforderung in einer Datei abzuspeichern.

Nach Abschluss der Eingaben ruft der Wizard die Funktion `createCSR()` der Klasse `User` auf, die mit den übergebenen Parametern die Zertifkatsanforderung generiert.

5.3.2 Ausstellen eines Zertifikats

Die Ausstellung von Zertifikaten kann – entsprechend Modell und Realität – nur von der Benutzeroberfläche *Zertifizierungsinstanz* initiiert werden. Hierbei muss unterschieden werden zwischen einer Zertifikatsanforderung, die direkt vom Antragsteller kommt und einer Anforderung, die bereits durch eine Registrierungsinstanz geprüft wurde. Der zweite Fall wird später beschrieben. Im ersten Fall wird über einen Dateidialog die entsprechende Datei aus der *Inbox* der Zertifizierungsinstanz ausgewählt und anschliessend durch den Wizard `W_IssueCert` bearbeitet. Die folgenden Schritte werden bei der Ausstellung eines Zertifikats durchlaufen:

1. Signaturverifizierung: Es wird überprüft, ob die Signatur der Zertifikatsanforderung tatsächlich mit dem entsprechenden privaten Schlüssel ange-

fertigt wurde. Intern wird hierzu die entsprechende Funktion `verify()` verwendet. Nur nach positiver Prüfung kann fortgefahren werden.

2. Identitätsprüfung: Die Zertifizierungsinstanz überprüft die Identität des Antragstellers mit geeigneten Mitteln.
3. Auswahl des Signierschlüssels der Zertifizierungsinstanz. Auf der Wizardseite werden zur Orientierung Aussteller und Gültigkeitsdaten für das zum Schlüssel gehörende Zertifikat angegeben. Letztere Information ist wichtig bei der Festlegung der Gültigkeit des ausgestellten Zertifikats.
4. Überpfügung des Distinguished Names
5. Angabe der Gültigkeit für das auszustellende Zertifikat
6. Überprüfung und ggf. Änderung der angeforderten Zertifikatserweiterungen (z.Zt. nur *Basic Constraints*)
7. Auswahl des Signieralgorithmus
8. Auswahl des Empfängers oder einer Datei zur Ausgabe des Zertifikats

Anschliessend wird mit Hilfe des entsprechenden CertificateGenerators das Zertifikat generiert und an den Empfänger übermittelt bzw. in die Datei geschrieben. Zusätzlich wird das erzeugte Zertifikat an den Verzeichnisdienst übertragen.

5.3.3 Validierung des Zertifizierungspfads

Der Wizard `W_CertPath` kann gestartet werden, wenn ein Zertifikat in der Zertifikatsliste auf der Benutzeroberfläche *Benutzer* oder *Zertifizierungsinstanz* ausgewählt ist. Worher sollte jedoch im Zertifikatsspeicher das Zertifikat der eigenen Wurzelinstanz – oder testweise auch ein anderes Zertifikat – als vertrauewürdig markiert sein. Ansonsten kann der Wizard nicht beendet werden. Zunächst kann auf der Startseite des Wizards ausgewählt werden, welche Zertifikatserweiterungen bei der Validierung bekannt sein sollen (zur Zeit noch nicht realisiert). Danach läuft die Validierung des Zertifizierungspfads in zwei Phasen ab:

Konstruktion des Zertifizierungspfads In der ersten Phase wird vom zu überprüfenden Zertifikat ausgehend der Pfad zu einem vertrauenswürdigen Zertifikat gesucht. Hierzu stehen verschiedene Möglichkeiten zur Verfügung:

Suche Zertifikatsspeicher : Es wird nach einem Ausstellerzertifikat für das momentan markierte zertifikat gesucht. Zunächst wird im Zertifikatsspeicher nach einem Zertifikat gesucht, dass den issuer name des markierten Zertifikats als subject name hat. Im Anschluss wird überprüft, das schon im Pfad vorhandene Zertifikat mit dem öffentlichen Schlüssel aus dem potentiellen Ausstellerzertitikat verifiziert werden kann. Ist dies der Fall, wird das Zertifikat in den Pfad übernommen und nach einem Ausstellerzertifikat für dieses gesucht.

Import aus Datei Import eines Zertifikats aus einer Datei. Weitere Vorgehensweise wie bei der Suche im Zertifikatsspeicher.

Verzeichnisdienst-Anfrage Ist ein benötigtes Zertifikat weder im KeyStore des Benutzers enthalten, noch als Datei vorhanden, bietet sich die Verzeichnisdienstanfrage als weiter Möglichkeit an das Zertifikat zu erhalten. Im folgenden Dialog kann der Anwender entweder nach dem Distinguished Name des Ausstellerzertifikats direkt oder per Eingabe eines Suchstrings nach dem Zertifikat suchen. Alle relevanten Zertifikate werden in einer Liste angezeigt und können in den KeyStore übernommen werden. Hiernach ist es notwendig nochmals den Punkt *Suche Zertifikatsspeicher* aufzurufen

Validierung des Zertifizierungspfads Im Anschluss an die Konstruktion des Zertifizierungspfads würde dieser in der zweiten Phase validiert. Die Implementierung ist an dieser Stelle jedoch noch nicht weit fortgeschritten.

5.3.4 Visualisierung von Gültigkeitsmodellen

Unabhängig von der Simulationsumgebung kann über den Dialog `D_Visual_ValModel` die Funktionsweise von verschiedenen Gültigkeitsmodellen visualisiert werden. Um direkt Zertifikate mit „passenden“Gültigkeitsdaten zu haben wird hier abstrahiert und es werden lediglich die gewünschten Daten in eine Tabelle eingetragen. Nach Auswahl des entsprechenden Gültigkeitsmodells werden die Gültigkeitszeiträume der einzelnen Zertifikate als Zeitstrahlen dargestellt. Die Zeitpunkte der Signaturerstellung und der Verifikation sowie das Ergebnis der Gültigkeitsprüfung werden ebenfalls in der Visualisierung dargestellt.
Die Hinterlegung von Testdaten ermöglichst es den Dialog ohne die langwierige

Eingabe von Gültigkeitsdaten sofort zu nutzen.
Anmerkung: Dieser Dialog nutzt fast keine Funktionalitäten des Plugins und ist nach minimalen Änderung auch separat nutzbar.

Anmerkung Um das Plugin ohne größeren Konfigurationsaufwand (z.B. Anlegen von Benutzern und Zertifizierungsstellen, Erzeugen von Schlüsseln und Zertifikaten) nutzen zu können, können über den Button *Testdaten generieren* eine Reihen von Entitäten erstellt werden.

Kapitel 6

Fazit und Ausblick

Zum Schluss kommt der Zeitpunkt um Resümee zu ziehen über die Entwickung des Plugins und die Anfertigung der Arbeit. Im Nachhinein betrachtet war die Einarbeitung in die Entwicklung von RCP-Plugins sehr zeitraubend aufwendig – häufig traten Fehler auf, die erst nach längerer Suche zugeordnet werden konnten. Trotz allem war dies jedoch eine lehrreiche Erfahrung. Ein zweiter Punkt, der sich anders entwickelt hat als zunächst geplant, ist die Entwicklung der graphischen Benutzeroberfläche. Auch hier hat es einige Zeit gedauert, bis eine brauchbare Oberfläche entstanden ist. Die grundlegende Arbeit, die Entwicklung der Simulation, war lehrreich und sehr interessant. Ich denke an dieser Stelle kann ich die Aussagen aus der Einleitung aus eigener Erfahrung bestätigen.

Ausstehendes

Unter anderem aus Zeitgründen sind zum Ende der Bearbeitung einige Funktionen noch nicht fertiggestellt. So gehören die Internationalisierung sowie die Erweiterung des Anwenderhandbuchs auf die To-Do-Liste. Da bis zuletzt einzelne Fehler in der Anwendung korrigiert wurden, kann nicht ausgeschlossen werden, dass sich hierdurch neue Probleme ergeben haben. Eine weitere nicht ganz abgeschlossene Baustelle ist die graphische Benutzeroberfläche. Hier gibt es noch Verbesserungsbedarf in gestalterischer Hinsicht.

Ideen

Im Rahmen der Entwicklung sind einige Ideen entstanden, deren Realisierung zum Teil schon begonnen wurde bzw. in den Entwurf des Modells einbezogen wurde. Dies sind im einzelnen:

Verschiedene Simulationsumgebungen Unter umständen ist es sinnvoll mehrere Simulationsumgebungen parallel zur Verfügung zu haben. Aus diesem grund wurde die Möglichkeit geschaffen den *Kontext* der Simulationsumgebung zu wechseln – genaugenommen ist dies der Wechsel der Simulationsumgebung. Standardmäßig verwendet das Plugin die Umgebung „default“, die hinterlegten Testdaten werden im Kontext „Testdaten“erstellt.

Im- und Export von Simulationsumgebung biete den Vorteil, dass eine auf einem Rechner aufgebaute Simulationsumgebung an andere Anwender verteilt werden kann. Grundsätzlich ist dies auch jetzt schon möglich, indem das Verzeichnis der entsprechenden simulationsumgebung kopiert und versandt wird. Der Im- und Export von der Benutzeroberfläche aus, z.B. als ZIP-Archiv, wäre natürlich eine anwenderfreundliche Lösung.

Vernetzung von Simulationsumgebungen bieten den Anreiz mit mehreren Personen simultan in verschiedenen Rollen die Funktionsweise eine PKI zu simulieren. Ansatzweise ist eine „Vernetzung“bereits jetzt möglich: Zertifikatsanforderungen können als Datei exportiert und anschliessend z.B. per Mail versandt werden. Der Empfänger importiert die Anforderung in seine Anwendung, stellt das Zertifikat aus und sendet es wieder zurück an den Antragsteller. Funktionen wie Verzeichnisdienstanfrage funktionieren af diesem Weg natürlich nicht.

Aufgaben für Anwender Um eine intensivere Beschäftigung des Anwenders mit dem thema zu erreichen, könnten verschiedene Aufgaben definiert werden, die der Anwender durchspielen soll. Diese könnten beispielsweise als *Spcikzettel* in das Plugin integriert werden.

Literaturverzeichnis

[C+08] D. Cooper et al. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, 05 2008.

[Dau06] Berthold Daum. *Java-Entwicklung mit Eclipse 3.2*. dpunkt.verlag GmbH, 2006.

[Ham01] Dr. Volker Hammer. Cross-zertifikate verbinden. *Datenschutz und Datensicherheit*, (25):65–70, 2001.

[Hoo05] David Hook. *Beginning Cryptography with Java*. Wiley Publishing, Inc., 2005.

[ITU05] Information technology – open systems interconnection – the directory: Public-key and attribute certificate frameworks, 08 2005.

[JCT10] Getting started with jcryptool, 01 2010.

[LFB+00] Peter Lipp, Johannes Farmer, Dieter Bratko, Wolfgang Platzer, and Andreas Sterbenz. *Sicherheit und Kryptographie in Java*. Addison-Wesley, 2000.

[Rap09] Georgios Raptis. Zertifikatsprofile für x.509 attributzertifikate v2.3.1, 2009.

[Sch01] Klaus Schmeh. *Kryptografie und Public-Key-Infrastrukturen im Internet*. dpunkt.verlag GmbH, 2001.

[Sch07] Klaus Schmeh. *Kryptografie*. dpunkt.verlag GmbH, 2007.