

# FlexiPKI - Realisierung einer flexiblen Public-Key-Infrastruktur

Johannes Buchmann<sup>1</sup> · Markus Ruppert<sup>1</sup> · Markus Tak<sup>1</sup>

Technische Universität Darmstadt  
Fachbereich Informatik

Fachgebiet Kryptographie, Computeralgebra, Verteilte Systeme  
<sup>1</sup>{buchmann, mruppert, tak}@cdc.informatik.tu-darmstadt.de

## Zusammenfassung

Wir beschreiben unser Projekt FlexiPKI, in dem eine Public Key Infrastruktur entwickelt wird, in der die kryptographischen Basistechniken schnell und mit minimaler Beeinträchtigung des laufenden Betriebs ausgetauscht werden können.

## 1. Einleitung

Mit der wachsenden Bedeutung der elektronischen Kommunikation im privaten und öffentlichen Bereich entsteht zunehmend die Notwendigkeit, Daten sicher, d.h. geheim, authentisch und vertraulich, zu speichern und zu übertragen. Für die Lösung dieser Aufgabe werden Public-Key Kryptosysteme verwendet. Die Verwendung solcher Systeme erfordert die Implementierung einer Public-Key Infrastruktur (PKI) [PKIX][SPKI]. In der PKI werden z.B. private Schlüssel sicher abgelegt, öffentliche Schlüssel in allgemein zugänglichen Verzeichnissen organisiert und Soft- und Hardwaremoduln bereitgestellt, mit denen man verschlüsseln und signieren kann [PKCS].

In dieser Arbeit beschreiben wir unser Projekt FlexiPKI, in dem wir eine flexible PKI entwerfen und prototypisch implementieren. Flexibel bedeutet, daß die verwendeten kryptographischen Basistechniken und Module leicht ersetzt werden können, wenn sie sich als unsicher herausstellen. Wir glauben, daß eine solche Flexibilität zentrale Voraussetzung für eine längerfristige sichere Verwendung von PKIs ist.

## 2. Warum Flexibilität?

Eine wichtige (aber nicht hinreichende) Voraussetzung für die Sicherheit einer PKI ist die Sicherheit der verwendeten kryptographischen Techniken [BuMa00]. Die meisten PKIs verwenden als Public-Key-Mechanismus ausschließlich das RSA-Verfahren. Manche verwenden auch ElGamal-Varianten in endlichen Körpern. Wenige benutzen ElGamal-Verfahren auf elliptischen Kurven. Die genannten Public-Key-Techniken beziehen ihre Sicherheit aus der Schwierigkeit gewisser mathematischer Probleme. Die Sicherheit des RSA-Verfahrens beruht auf der Schwierigkeit, große Primfaktoren einer zusammengesetzten natürlichen Zahl zu finden. Die Sicherheit des ElGamal-Verfahrens beruht auf dem Problem, diskrete Logarithmen

in endlichen Körpern oder auf elliptischen Kurven zu berechnen. Es ist aber nicht bekannt, ob eines dieser Probleme wirklich schwierig ist. Im Gegenteil, es ist bekannt, daß die Probleme leicht sind, wenn man sogenannte Quantencomputer verwendet [Shor94]. Aber noch weiß keiner, ob man Quantencomputer wirklich bauen kann. Es könnte aber schon morgen ein schneller Algorithmus gefunden werden, der auf einem herkömmlichen Computer läuft und natürliche Zahlen faktorisiert oder diskrete Logarithmen berechnet. Dann wären die entsprechenden PKIs unsicher.

Was macht man, wenn eine kryptographische Basistechnik einer PKI unsicher geworden ist? Man ersetzt sie durch eine andere, wenn das geht. Meist geht das aber nicht so einfach. Die Abhängigkeit der PKI von der Basistechnik ist zu komplex. Zum Beispiel: Die meisten Chipkarten können nur RSA-Algorithmen ausführen. Würde RSA unsicher, müßte man zuerst neue Karten entwickeln, die Software entsprechend anpassen und alle vorhandenen Chipkarten austauschen. Das dauert sehr lange und ist sehr teuer.

Man hätte ähnliche Abhängigkeiten, wenn Türschlösser Teil der Tür wären und alle Türen dasselbe Schloß hätten. Man könnte dann nicht einfach ein Schloß austauschen, dessen Schlüssel verloren gegangen wäre, sondern man müßte alle Türen komplett austauschen. Das Beispiel läßt sich noch weiterdenken: die Türen hängen nicht in Scharnieren, sondern sind fest mit der Wand verbunden. Um sie zu entfernen, müßten die Mauern aufgestemmt werden. Bei der Absicherung von Gebäuden geht man sehr viel modularer vor.

Bei der augenblicklichen Konstruktion der meisten PKIs wäre es ein Supergau, wenn sich die zugrundeliegende Basistechnik als unsicher erweisen würde. Darum meinen wir, daß man folgendes tun muß:

- a) Neue Basistechniken müssen erfunden werden.
- b) Die neuen Basistechniken müssen bis zur praktischen Verwendbarkeit entwickelt werden.
- c) PKIs müssen so entwickelt werden, daß Basistechniken schnell und ohne große Störung des laufenden Betriebs ausgetauscht werden können.

### **3. Eigenschaften einer PKI**

Die wichtigste Eigenschaft unserer FlexiPKI ist die Austauschbarkeit der kryptographischen Basistechniken. Weitere Eigenschaften, die eine PKI aus unserer Sicht haben soll, werden in diesem Abschnitt dargestellt.

Eine Public Key Infrastruktur sollte flexibel sein, d.h. sie sollte Schnittstellen zu allen wichtigen Komponenten bieten. Die Flexibilität soll sich nicht nur auf die verwendeten Basistechniken sondern auch auf die verwendeten Standards für Zertifikate und Schlüsselinformationen beziehen. Außerdem wäre es wünschenswert, die anzuwendenden Sicherheitsrichtlinien ("Policies") bei Bedarf ersetzen zu können. Durch diese Abstraktionen reduziert sich die PKI auf ein Skelett von Schnittstellen, das die Zusammenarbeit der einzelnen Komponenten sicherstellt.

Benutzerfreundlichkeit ist ein weiterer wesentlicher Faktor. Der Benutzer soll ein Werkzeug in die Hand bekommen, das die gewohnte Benutzungsoberfläche seiner Anwendungen kaum verändert. Er möchte sich nicht um Details von Verschlüsselungsalgorithmen oder um inkompatible Zertifikatsstandards kümmern. Solche Abläufe müssen im Hintergrund transparent

durchgeführt werden. Dabei soll die kryptographische Funktionalität aber nicht vollständig verborgen werden, denn natürlich muß sich der Benutzer beispielsweise über die möglichen Folgen einer ausgestellten Signatur im klaren sein.

Die eingesetzte Software sollte leicht zu evaluieren sein. Die Semantik der verwendeten Programme muß in ihrem Wirkungszusammenhang überschaubar und verständlich sein. So fördern einheitliche Schnittstellen die Durchschaubarkeit und Wiederverwendbarkeit vorhandener Applikationen im kryptographischen Umfeld. Es sollte für einen geschulten Betrachter möglich sein, sich von der Sicherheit des gesamten Systems überzeugen zu können.

Integration in vorhandene Soft- und Hardwaresysteme sowie die Erweiterbarkeit während des laufenden Betriebs erfordern objektorientierte und portable Strukturen. Diese müssen jedoch so gestaltet sein, daß damit auch Anwendungen realisiert werden können, die wenig zusätzliche Ressourcen benötigen (Thin Clients).

Da eine einmal eingerichtete Public Key Infrastruktur über einen längeren Zeitraum in Betrieb bleiben soll, ist es wichtig, auf ausreichende Skalierbarkeit zu achten. Gerade im industriellen Umfeld ist ein langfristiger Ausblick auf die Bedürfnisse oft schwierig, zumal die Kosten einer PKI erheblich von deren Größe abhängen können (Kartenleser müssen ausgegeben werden, Lizenzierung pro Zertifikat usw.)

Public Key Infrastrukturen bilden nur in den seltensten Fällen geschlossene Systeme. Meist müssen sie mit anderen Systemen kommunizieren, z.B. mit anderen PKI oder Anwendungs-komponenten. Daher muß auf ausreichende Interoperabilität geachtet werden. Dabei ist der Einsatz von internationalen Standards unabdingbar.

Der Administrationsaufwand einer laufenden PKI ist auch ein nicht zu unterschätzender Kostenfaktor. Ausfallzeiten können hohe Folgekosten nach sich ziehen.

## 4. Grundlegende Designentscheidungen

Bei der Entscheidung über die Programmierplattform für das Projekt fiel die Wahl auf JAVA. JAVA ist eine objektorientierte und plattformunabhängige Programmiersprache. Sie besitzt alle Attribute einer modernen Plattform wie Wiederverwendbarkeit, Modularität, automatische Speicherverwaltung, Robustheit durch ausgefeilte Fehlerbehandlung, ein einheitliches Ein-/Ausgabekonzept und eine einfache Anbindung an Datenbanken über standardisierte Schnittstellen.

Mit der Java Cryptographic Architecture (JCA) [Knud98][Oaks98], besitzt JAVA bereits ein Konzept für den transparenten Einsatz beliebiger Kryptoalgorithmen. Es stehen Signaturen, Verschlüsselung, Schlüsselerzeugung, Zertifikatsmanagement und Message Digests zur Verfügung. Teil des JCA-Konzeptes ist die Zusammenfassung von Implementierungen zu Bibliotheken, den sogenannten Providern. Es können mehrere dieser Provider unabhängig voneinander gleichzeitig verwendet werden. Dies ermöglicht die Verwendung von Algorithmen eines Herstellers des eigenen Vertrauens. Die JCA berücksichtigt gängige Standards wie X.509v3. Dadurch stellt JAVA ein Höchstmaß an Flexibilität zur Verfügung.

Die Integration von JAVA in moderne Betriebssysteme ist noch nicht abgeschlossen. Daher fügt JAVA sich noch nicht transparent in jede Oberfläche ein. Unterschiedliche Ansätze im Komponentenmodell tragen dazu bei, daß bestehende Applikationen nicht ohne weitere

Hilfsmittel auf die JAVA-Plattform zugreifen können. Darunter leidet natürlich die Benutzerfreundlichkeit. Hier besteht noch ein deutlicher Entwicklungsbedarf.

Der Quelltext eines JAVA-Programmes ist aufgrund der guten Strukturierung der Sprache leicht zu lesen. In Verbindung mit strikter objektorientierter Programmierung und einer hervorragenden Dokumentationsschnittstelle (javadoc) ist es einfach, übersichtliche und durchschaubare Software zu schreiben, die leicht evaluierbar ist.

Um den Sprachkern von JAVA herum entstehen ständig neue Erweiterungen, wie zum Beispiel "java spaces", die verteiltes Rechnen unterstützen soll. Außerdem wird die JAVA- Produktpalette in mehrere Basispakete unterteilt, so daß die JAVA-fähige SmartCard genauso wie der große Unternehmens-Server eine sinnvoll ausgestattete Infrastruktur vorfindet. Derartige Vorstöße garantieren gute Skalierbarkeit auf längere Sicht.

Einige wichtige Standards, wie z.B. X.509 Zertifikate, werden bereits direkt vom JAVA Basissprachumfang implementiert. Andere Standards können über Erweiterungen hinzugefügt werden, wobei Sun meist eine Programmierschnittstelle bereitgestellt, die von diversen Anbietern ausgefüllt wird - ähnlich wie bei der JCA. Auf diese Weise läßt sich JAVA leicht um diejenigen Standards erweitern, die Interoperabilität mit anderen Systemen gewährleisten.

Die bislang einzige Programmiersprache, die im Hinblick auf Sicherheit konzipiert wurde, ist JAVA. Der Policymanager von JAVA ermöglicht eine fein granulいたe Vergabe von Rechten an Applikationen und bedingtes Ausführen von Klassen abhängig von ihrer Herkunft und ihrer Signatur. Durch diese Maßnahmen wird ein hohes Maß an Administrierbarkeit und Sicherheit gewährleistet.

## **5. Flexible PKI - Projektstatus**

Wir beschreiben die Komponenten von FlexiPKI.

### **5.1 Zertifizierungsinstanz LiDIA-CA**

Mit LiDIA-CA wurde eine vollständig in JAVA realisierte Zertifizierungsinstanz geschaffen. Die Software wurde bereits auf drei Plattformen erfolgreich eingesetzt (SUN Solaris, Linux, Windows NT 4.0). Zur Zeit wird LiDIA-CA im Rahmen des Nordrheinwestfälischen Projektes "Digitale Bibliothek" an der Universität Bielefeld verwendet.

Bei der Entwicklung dieses Prototyps war das Hauptziel die Flexibilität. So sind durch das konsequent modulare Konzept die Zugfallszahlen- und Schlüsselgeneratoren ebenso wie die Personalisierungsmodule beliebig austauschbar und passen sich den verwendeten Chipkarten an.

LiDIA-CA besteht aus den Basiskomponenten

- Registrierung (RA) zur Annahme von Zertifikatsanträgen
- Zertifizierung (CA) zum Austellen von Zertifikaten
- Revokation (RS) zum Widerrufen von Zertifikaten
- Personalisierung (PS) zum austellen von Soft- und Hardware-PSE

Durch Schnittstellen zu Verzeichnisdiensten und SQL-fähigen Datenbanken wird das Zertifikats- und Schlüsselmanagement ermöglicht.

## 5.2 PSE

Ein Personal Security Environment (PSE) bewahrt die geheimen Schlüssel der Benutzer sicher auf.

Eine flexible PKI erfordert eine flexible, transparente Integration der Hardware-PSE-Komponenten (Chipkarten). Grundlegende PKI-Funktionen wie das Signieren werden normalerweise auf den Chipkarten ausgeführt. Die PKI muß sich daher der Verwendung neuer Chipkarten anpassen können. Um dies zu ermöglichen, wird eine Erweiterung der JCA entwickelt die eine einheitliche Verwendung der von den Herstellern bereitgestellten Software gestattet. Diese Software beruht meist auf einem der Standards PC/SC, OpenCard oder PKCS#11 [PKCS].

Die zusammen mit LiDIA-CA bislang verwendeten Chipkarten und Chipkartenleser können über eine in unserer Arbeitsgruppe entwickelte Bibliothek benutzt werden. Diese Bibliothek wurde aber vor der Implementierung der JCA geschrieben und ist daher dazu inkompatibel. Wir sind dabei, die JCA so zu erweitern, daß sie auch Hardware-PSE-Komponenten ansprechen kann.

Geplant ist der Einsatz von JAVA-Karten, wodurch ein entscheidender weiterer Schritt in Richtung Flexibilität möglich wird. Erst Java-Karten erlauben es, die verwendeten kryptographischen Techniken auch in der PSE auszutauschen.

## 5.3 Anwendungen (Clientsoftware)

Im Rahmen des Projekts FlexiPKI werden Anwendungen typischerweise als Erweiterungen (PlugIns) zu bestehenden Applikationen konzipiert. In dem momentan entstehenden Realisierungskonzept wird versucht, diese Erweiterungen als JAVA-PlugIns zu verwirklichen. Die Vorteile sind offensichtlich. Es können alle bei der Entwicklung von LiDIA-CA entstandenen Schnittstellen wiederverwendet werden. Die Anwendungen erhalten dadurch die Basis für die angestrebte Flexibilität.

Derzeit wird an einer ersten prototypischen Realisierung in Form einer S/MIME<sup>1</sup>-Erweiterung für das E-mail-Programm "pine" gearbeitet. Sie soll zugleich Ausgangspunkt für eine entsprechende Erweiterung der E-Mail-Programme Outlook, Netscape und Eudora dienen.

## 5.4 Zeitstempeldienst

In enger Zusammenarbeit mit Nippon Telegraph and Telephone (NTT) wird derzeit ein verteilter Zeitstempeldienst entwickelt, der vollständig in JAVA programmiert ist. Dabei entstehenden Algorithmen zur verteilten Schlüsselgenerierung und zum verteilten Signieren. Die Implementierung erfolgt JCA-konform und bietet so die Basis für eine sichere Zertifizierungsinstanz, die keine besonderen baulichen Maßnahmen zum Schutz des privaten Schlüssels der CA benötigt.

---

<sup>1</sup> S/MIME ist ein Standardformat für signierte und verschlüsselte E-mail.

## 5.5 Verfügbarkeit alternativer Kryptoalgorithmen

Eine flexible PKI soll mehr Sicherheit in der Anwendung durch die Verwendung alternativer Kryptoalgorithmen bieten. Da derzeit, mit Ausnahme von RSA, nur wenige Verfahren als JAVA-Implementierung verfügbar sind, werden parallel zur Weiterentwicklung von FlexiPKI JCA-konforme JAVA-Provider für alternativer Kryptoalgorithmen geschrieben.

Folgende Verfahren sind bereits realisiert:

- symmetrische Verschlüsselung mit den AES Kandidaten RC6, MARS und den früheren Kandidaten E2 und Safer+
- ElGamal und DSA für quadratische Formen

An weiteren Verfahren wird momentan gearbeitet:

- NICE und Rabin mit quadratischen Formen
- Verschlüsselung mit elliptischen Kurven über  $GF(p)$ ,  $p > 2$
- Signaturen mit elliptischen Kurven über  $GF(p)$ ,  $p > 2$
- Signaturen mit elliptischen Kurven über  $GF(2^n)$
- Verteilte Erzeugung von RSA-Schlüsselpaaren
- Verteiltes Signieren mit RSA

### Literatur

- [BuMa00] Johannes Buchmann, Markus Mauerer, Wie sicher ist die Public-Key Kryptographie, eingereicht bei der Arbeitskonferenz Systemsicherheit, März 2000
- [Knud98] J. Knudsen, Java Cryptography, O'Reilly & Associates Inc, 1998
- [Oaks98] S. Oaks, Java Security, O'Reilly & Associates Inc, 1998
- [PKCS] Public-Key Cryptography Standards, <http://www.rsa.com>
- [PKIX] IETF Working Group, IETF-WorkingPublic Key Infrastructure (X.509) (pkix), (<http://www.ietf.cnri.reston.va.us/html.charters/pkix-charter.html>)
- [Shor94] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, Proceedings of the IEEE 35th Annual Symposium on Foundations of Computer Science, 124-134, 1994
- [SPKI] IETF working group, Simple Public Key Infrastructure (SPKI), (<http://www.clark.net/pub/cme/html/spki.html>)