

# Ein S/MIME-Plugin für Outlook zur Verwendung alternativer kryptographischer Algorithmen

Markus Tak  
Technical Report  
Technische Universität Darmstadt

14. Mai 2002

## Zusammenfassung

Das Projekt FlexiPKI am Lehrstuhl Buchmann hat es sich zur Aufgabe gemacht, die Entwicklung und Integration von alternativen kryptographischen Algorithmen voranzutreiben. Das derzeit wohl am meisten genutzte kryptographische Protokoll ist S/MIME zur Absicherung von Emails durch Verschlüsselung und Digitale Signaturen. In diesem Paper wird das in der AG Buchmann entwickelte Outlook- Plugin vorgestellt, das die Verwendung von alternativen kryptographischen Algorithmen innerhalb des S/MIME Standards ermöglicht. Durch eine einfach zu bedienende Oberfläche ist es so jedem interessierten Benutzer möglich, Einblick in die praktische Anwendbarkeit solcher alternativen Verfahren zu bekommen.

## 1 Einleitung

Die Notwendigkeit, sich mit der Entwicklung von alternativen kryptographische Algorithmen zu befassen und diese in real existierende Anwendungen zu integrieren, ergibt sich aus der Tatsache, daß heutige Standard-Anwendungen, die bereits Public-Key-Kryptographie benutzen, auf wenige kryptographische Algorithmen wie RSA und DSA beschränkt sind. Da diese Algorithmen darüber hinaus auf den gleichen mathematischen Grundprinzipien beruhen, würden sämtliche darauf aufbauenden Sicherheits-Infrastrukturen zusammenbrechen, wenn es einen erfolgreichen Angriff auf das zugrunde liegende Prinzip gäbe ([BRT00], [BUMA00]).

In der Arbeitsgruppe Buchmann sind eine Reihe von Implementierungen kryptographischer Algorithmen entstanden und werden laufend weiterentwickelt. Dabei wurden Standard- Algorithmen wie RSA und DSA ebenso berücksichtigt wie alternative Algorithmen wie z.B. Elliptische Kurven oder Zahlkörper-Kryptographie [NFC]. Als Basis für diese Implementierungen hat man sich für Java als Programmierplattform wegen der offenen Krypto-Architektur "JCA" (Java Cryptographic Architecture) entschieden ([BRT00]).

Es reicht jedoch nicht aus, nur kryptographische Algorithmen zu implementieren. Selbst die beste Funktionsbibliothek kann nur von Experten benutzt werden. Daher entschied man sich dazu, die kryptographischen Algorithmen auch in Anwendungen zu integrieren. Aus diesem Grund entstehen parallel Pakete, wie z.B. die Trustcenter-Software "FlexiTrust" oder das hier beschriebene Outlook-Plugin "FlexiSMIME". Alle Anwendungen setzen konsequent auf die JCA auf, um kryptographische Algorithmen zu nutzen, und bilden diese auf die relevanten internationalen Standards wie z.B. X.509, PKCS, LDAP etc. ab.

Die Entscheidung, ein Outlook S/MIME-Plugin zu entwickeln, fiel aus mehreren Gründen:

- S/MIME ist eines der meistgenutzten kryptographischen Protokolle
- Der S/MIME Standard sieht bereits alternative kryptographische Algorithmen vor, da auf die offenen PKI-Standards PKIX und X.509 zurückgegriffen wird
- Alle uns bekannten S/MIME Implementierungen nutzen lediglich

Standard-Algorithmen wie RSA oder DSA, welche die oben erwähnten Schwächen aufweisen<sup>1</sup>.

- Outlook ist ein weit verbreiteter Email-Client, der gut erweiterbar ist

## 2 Design

Ein wesentliches Ziel bei der Entwicklung des Outlook-Plugins war die Benutzerfreundlichkeit. Es sollte ein Werkzeug entstehen, mit dem interessierte Benutzer auf einfache Weise mit alternativen kryptographischen Algorithmen experimentieren können.

### 2.1 Outlook-Integration

Outlook sieht von Hause aus keine Schnittstelle zu Java vor, so daß hier zunächst eine prinzipielle Anbindung geschaffen werden mußte. Diese besteht aus zwei Elementen:

- Die Outlook Exchange Extension [EXCHEXT]
- Das Java Native Interface [JNI]

Die *Outlook Exchange Extension* bildet eine C-Schnittstelle, über die Erweiterungen der Outlook-Funktionalität über eine externe Bibliothek<sup>2</sup> eingebunden werden können. Dabei kann die Exchange Extension insbesondere auf den Inhalt von Emails vor dem Versand bzw. vor der Darstellung nach dem Empfang zugreifen und diese verändern. Diese Eingriffsmöglichkeiten sind notwendig, um eine Email ver- bzw. entschlüsseln zu können und um eine Digitale Signatur anzubringen bzw. zu prüfen, wenn alle dazu notwendigen Funktionen extern bereitgestellt werden sollen. Kommerzielle S/MIME Plugins wie z.B. Secude AuthenteMail [SAM] oder SSE Trusted-Mime [SSE] setzen ebenfalls an dieser Schnittstelle an.

Da es sich bei der Outlook-Extension um eine in C implementierte Funktions-Bibliothek handelt, ist noch ein weiteres Brückenelement notwendig, um die gewünschte Anbindung an die JCA zu erzielen. Genau für diese Zwecke gibt es in Java das JNI-Interface, welches Funktionen für Java-Aufrufe aus C heraus (und umgekehrt) zur Verfügung stellt.

Details zur Outlook-Anbindung finden sich in [DS01].

### 2.2 S/MIME Verarbeitung

Nachdem es nun gelungen ist, Java-Funktionen aus Outlook heraus aufzurufen, ist noch die zentrale Aufgabe zu klären: die S/MIME Verarbeitung.

---

<sup>1</sup>Es entstehen derzeit jedoch erste Plugin's, die auch ECDSA unterstützen

<sup>2</sup>Windows-DLL, Dynamic Linkable Library

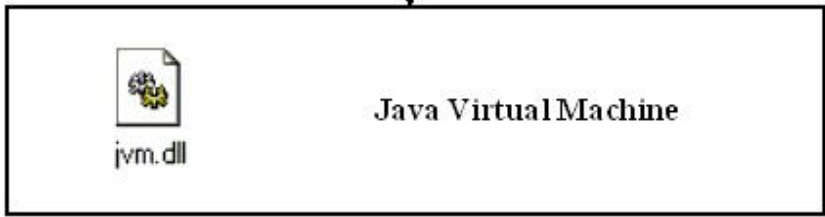
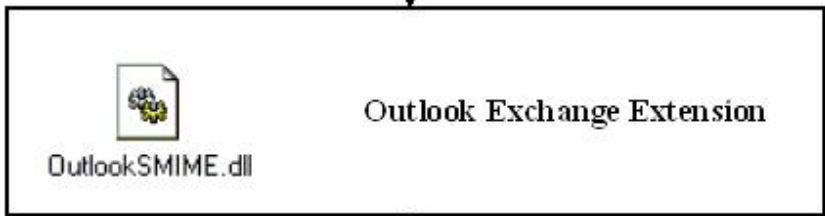
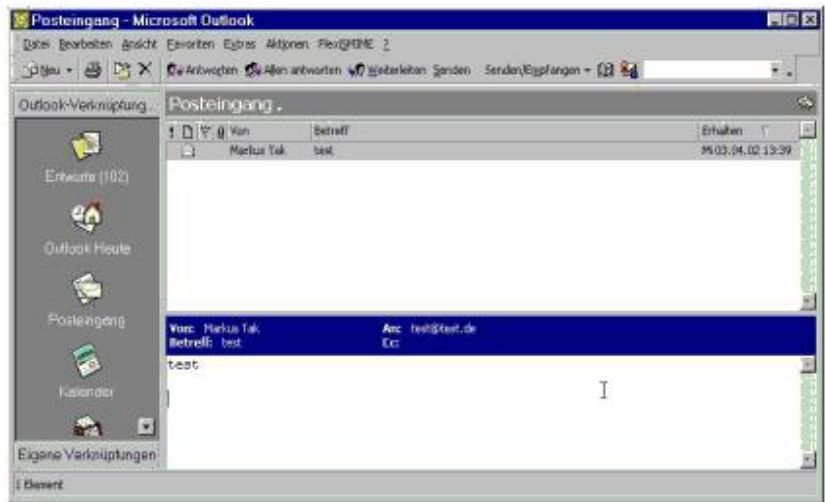


Abbildung 1: Outlook-Anbindung an Java

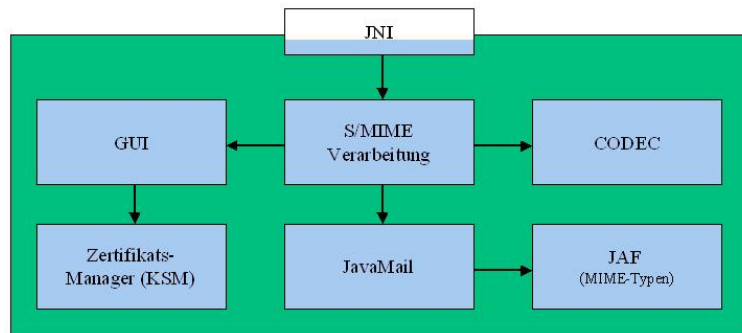


Abbildung 2: S/MIME Verarbeitung

Der Java Sprachumfang gibt von Hause aus keine Schnittstellen oder fertigen Module für die Verarbeitung von S/MIME Nachrichten her. Allerdings gibt es mit JavaMail ([JAM]) und dem Java Activation Framework ([JAF]) ein Paket zur Verarbeitung von Emails inkl. beliebigen MIME-Attachments. Da S/MIME lediglich eine Erweiterung des MIME-Konzepts darstellt, wurde in [DS01] eine S/MIME Bibliothek erstellt, die auf JavaMail aufsetzt.

Da man mit der Implementierung von S/MIME nun auch die Welt der Kryptographie-Standards betritt, stellt sich die Frage nach einer Lösung für die Umsetzung der komplexen Kodierung von kryptographischen Nachrichten und Elementen (wie z.B. Zertifikaten oder Schlüsseln). Hier wird seit Beginn der Arbeiten in der AG Buchmann das im Rahmen des SeMoA-Projektes [SEMOA] entstandene CODEC-Paket eingesetzt. Es ermöglicht den einfachen Umgang mit ASN.1 Datenobjekten, ohne sich um die umfangreichen Details der DER/BER-Kodierung kümmern zu müssen. CODEC ist ein ausgereiftes Paket, das ebenfalls auf die JCA aufsetzt und unabhängig vom eingesetzten JCA-Provider arbeitet.

## 2.3 Krypto-Provider

Die S/MIME Verarbeitung benötigt ihrerseits natürlich die Bereitstellung von kryptographischen Algorithmen in Form einer Bibliothek. An der AG Buchmann sind eine Reihe von JCA-Providern entstanden, die außer den Standard-Algorithmen wie RSA oder DSA auch alternative Algorithmen zur Verfügung stellen, wie z.B. ECDSA (Elliptische Kurven über GF(p) und GF(2n)) oder IQ-DSA, IQ-RDSA und IQ-GQ (Zahlkörper-Kryptographie, siehe [NFC])

Die JCA-Provider können unter [FP] auch einzeln heruntergeladen werden.

Ein wichtiges Kriterium beim Design des Outlook-Plugins war es, keine Annahmen über die eingesetzten JCA-Provider bzw. über die eingesetzten kryptographischen Verfahren zu machen, so daß das System jederzeit für weitere Provider und Verfahren offen ist. Das CODEC-Paket war aufgrund der gleichen Ausrichtung dabei eine große Hilfe.

## 3 Implementierung

### 3.1 Zertifikatsverwaltung

Um das Outlook-Plugin sinnvoll benutzen zu können, wurde ein Modul zur Verwaltung von Zertifikaten benötigt. Die Schnittstelle zur Zertifikatsverwaltung wurde in [DS01] definiert und in [VS01] und [DL02] implementiert. Entstanden ist der Keystore-Manager, ein Werkzeug, das die einfache Verwaltung von Zertifikaten in Java-Keystores ermöglicht<sup>3</sup>. Neben den Funktionen innerhalb des Outlook-Plugins ist der KeyStore-Manager auch einzeln nutzbar, um Keystores zu einzusehen und zu verwalten.

Folgende Funktionen wurden im Keystore Manager implementiert:

- Ansicht von Zertifikaten und Schlüsseln innerhalb von Java-Keystores
- Erzeugung von Schlüsselpaaren auf Basis der installierten JCA-Provider
- Zertifizierungs-Anfragen für öffentliche Schlüssel an ein beliebiges Trustcenter im PKCS#10 Format und Import des entsprechenden X.509 Zertifikats
- Im- und Export von Schlüsseln inklusive dazugehöriger Zertifikate im PKCS#12 Format
- Im- und Export von Zertifikaten im X.509 Format (als Datei und in Form von Verzeichnis-Zugriffen über LDAP)
- Import von bereits vorhandenen Keystores
- Schnittstelle zum Outlook-Plugin zum automatischen Auffinden von Zertifikaten über Suchkriterien (z.B. über die Email-Adresse)

### 3.2 Integration von JCA-Providern

Das Outlook-Plugin und das zur ASN.1-Kodierung verwendete CODEC-Paket sind so aufgebaut, daß sie keinerlei Annahmen über die Implementierung der einzusetzenden JCA-Provider machen. Dadurch sind auch externe JCA-Provider mit dem Outlook-Plugin nutzbar. Das setzt jedoch voraus, daß sich die JCA-Provider streng an die von SUN vorgegebenen Richtlinien für die Implementierung von Providern [SUN] halten müssen.

Da bei nahezu allen getesteten externen JCA-Providern hier mehr oder weniger unvollständig vorgegangen wird, seien die Richtlinien hier nochmals zusammengefaßt:

- Für jeden Algorithmus müssen auch die passenden OID's (Object Identifier) als *Alias* registriert werden. Das gilt für alle JCA-Objekte:
  - KeyFactory, SecretKeyFactory
  - AlgorithmParameters

---

<sup>3</sup>Keystores sind das Abstraktions-Konzept in der JCA, um Schlüssel und Zertifikate zu speichern

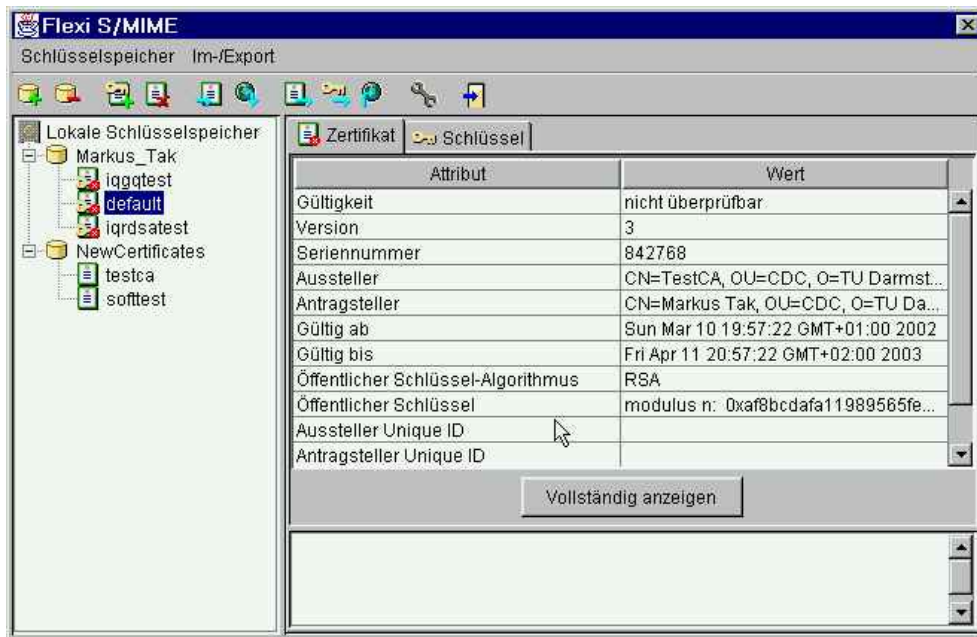


Abbildung 3: Der KeyStore Manager

- Signature
- Cipher
- MessageDigest
- Mac

Beispiel (Anmeldung der Algorithmen in der Provider-Klasse):

*"Alg.Cipher.RSA", "de.flexiprovider.core.rsa.RSA"*

*"Alg.Alias.1.2.840.113549.1.1.1", "RSA"*

- Für jede OID muß auch das "Back-Mapping" auf einen JCA-Namen angegeben werden (insbesondere wenn es mehrere OIDs für ein Verfahren gibt), und zwar in der Form *"Alg.Alias.komponente.OID.oid"*, also beispielsweise *"Alg.Alias.Cipher.OID.1.2.840.113549.1.1.1", "RSA"*.
- Bei symmetrischen Verfahren muß die OID-Zuordnung auch den *Mode* und das *Padding-Schema* umfassen, also beispielsweise:  
*"Cipher.DESede", "de.flexiprovider.core.desede.DESede"*  
*"Alg.Alias.Cipher.1.2.840.113549.3.7",*  
*"DESede/CBC/PKCS5Padding"*  
*"Alg.Alias.Cipher.OID.1.2.840.113549.3.7",*  
*"DESede/CBC/PKCS5Padding"*
- Der korrekte Umgang mit Modes und Paddings bei Cipher-Objekten muß gewährleistet sein (Darstellung in "/"-Schreibweise, also beispielsweise *"DESede/CBC/PKCS5Padding"*)
- Signatur-Verfahren müssen auch (als Alias) in einer "Slashed-Form" angemeldet werden, aus welcher der verwendete Hash-Algorithmus

hervorgeht, also beispielsweise “SHA1/RSA” zusätzlich zu “SHA1withRSA”

Dabei muß der Hash-Algorithmus über diesen Teil-Namen (im obigen Beispiel “SHA1”) aufrufbar sein. Das Roh-Signaturverfahren (im obigen Beispiel “RSA”) muß zumindest über ein OID-Mapping verfügen, in dem der Algorithmus-Name vorkommt, z.B. über die KeyFactory. Dies ist für einen Provider-unabhängigen Zugriff auf PKCS#7 unbedingt notwendig, da hier Hashverfahren und Roh-Signaturverfahren getrennt kodiert werden und die JCA nur die Kombination aus beiden kennt!

Beispiel für die korrekte Anmeldung von SHA1/RSA Signature:

```
put(“Signature.SHA1withRSA”,
    “de.flexiprovider.core.rsa.SHA1withRSA”);
put(“Alg.Alias.Signature.SHA/RSA”, “SHA1withRSA”);
put(“Alg.Alias.Signature.SHA-1/RSA”, “SHA1withRSA”);
put(“Alg.Alias.Signature.SHA1/RSA”, “SHA1withRSA”);
put(“Alg.Alias.Signature.1.2.840.113549.1.1.5”, “SHA1withRSA”);
put(“Alg.Alias.Signature.1.3.14.3.2.29”, “SHA1withRSA”);
put(“Alg.Alias.Signature.OID.1.3.14.3.2.29”, “SHA1withRSA”);
```

Dies gilt auch für Signatur-Verfahren, die nur für eine einzige Hashfunktion definiert sind, z.B. ECDSA:

```
put(“KeyPairGenerator.ECDSA”,
    “de.flexiprovider.ec.ecdsa.ECDSAKeyPairGenerator”);

put(“Signature.SHA1withECDSA”,
    “de.flexiprovider.ec.ecdsa.ECDSASignature”);
put(“Alg.Alias.Signature.ECDSA”, “SHA1withECDSA”);
put(“Alg.Alias.Signature.SHA1/ECDSA”, “SHA1withECDSA”);
put(“Alg.Alias.Signature.1.2.840.10045.4.1”, “SHA1withECDSA”);
put(“Alg.Alias.Signature.OID.1.2.840.10045.4.1”, “SHA1withECDSA”);

put(“KeyFactory.ECDSA”,
    “de.flexiprovider.ec.ecdsa.ECDSAKeyFactory”);
put(“Alg.Alias.KeyFactory.1.2.840.10045.2.1”, “ECDSA”);
put(“Alg.Alias.KeyFactory.OID.1.2.840.10045.2.1”, “ECDSA”);

put(“AlgorithmParameters.ECDSA”,
    “de.flexiprovider.ec.ecparameters.ECParameters”);
put(“Alg.Alias.AlgorithmParameters.1.2.840.10045.4.1”,
    “ECDSA”);
put(“Alg.Alias.AlgorithmParameters.OID.1.2.840.10045.4.1”,
    “SHA1withECDSA”);

put(“AlgorithmParameterGenerator.ECDSA”,
    “de.flexiprovider.ec.ecparameters.ECParameterGenerator”);
put(“Alg.Alias.AlgorithmParameterGenerator.1.2.840.10045.4.1”,
    “ECDSA”);
put(“Alg.Alias.AlgorithmParameterGenerator.OID.1.2.840.10045.4.1”,
    “ECDSA”);
```

### 3.3 Installation

Um eine einfache Benutzung zu gewährleisten, wurde mit Hilfe des Microsoft Visual Studio Installers ein Setup erstellt, das alle zur Installation notwendigen Schritte automatisch vornimmt:

- Installation des Java Runtime Environment (JRE), um eine einheitliche und unabhängige Java-Installation zu haben
- Installation und Integration der Outlook Exchange Extension DLL (OutlookSMIME.DLL)
- Installation der notwendigen Java-Pakete:
  - JavaMAIL
  - Java Activation Framework (JAF)
  - CODEC
  - FlexiSMIME (Java-Klassen des Outlook-Plugins)
  - CertificateManager (enthält den KeyStore-Manager)
  - JCE (Java-Erweiterung für starke Kryptographie, entnommen aus dem *Cryptix*-Paket)
  - FlexiProvider (Enthält alle unterstützten kryptographischen Algorithmen)
  - Netscape LDAP-Klassen (benötigt für den LDAP-Zugriff)
- Installation eines Default-Benutzerprofils mit leerem KeyStore

Das Setup wurde unter Windows 98, Windows NT und Windows 2000 unter Outlook 98 und Outlook 2000 SR1 erfolgreich getestet.

## 4 Bedienung

Abbildung 4 zeigt die Integration von FlexiSMIME in die Outlook-Menüleiste. Über den Menüpunkt *FlexiSMIME* können alle Funktionen des Outlook Plugins gesteuert werden.

Nach der Installation enthält das Benutzerprofil zunächst nur einen Default-KeyStore mit dem Wurzelzertifikat der TestCA (siehe Abbildung 5).

Um sinnvoll arbeiten zu können, sollte man sich zunächst ein Schlüssel-paar generieren. Dazu wählt man den Menüpunkt *Schlüsselspeicher - neues Schlüsselpaar erzeugen*. Abbildung 6 zeigt den Dialog, in dem man für den neuen Schlüssel den Namen, den Algorithmus, die Schlüssellänge und ein Paßwort festlegen kann. Außerdem muß der Signatur-Algorithmus für das selbst-signierte Zertifikat ausgewählt werden. Nach der Schlüsselgenerierung erscheint ein Datei-Fenster, über das man den öffentlichen Schlüssel in einer PKCS#10 Datei abspeichern kann, um ihn bei einem Trustcenter zur Zertifizierung einzureichen. Das fertige Zertifikat kann über die Import-Funktion wieder importiert werden und wird dann automatisch dem entsprechenden Schlüssel wieder zugeordnet.

Wenn man nun einen Blick in die Zertifikatsverwaltung wirft, kann man sich den neuen Schlüssel in allen Details ansehen (siehe Abbildung 7).

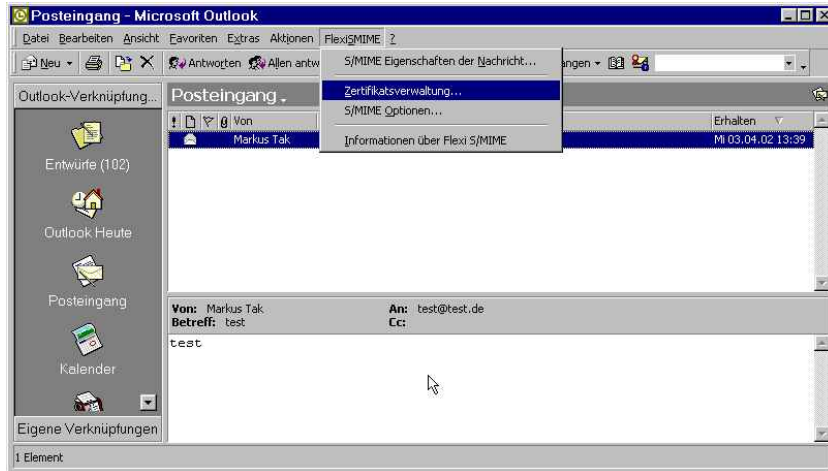


Abbildung 4: Integration in die Outlook Menüleiste

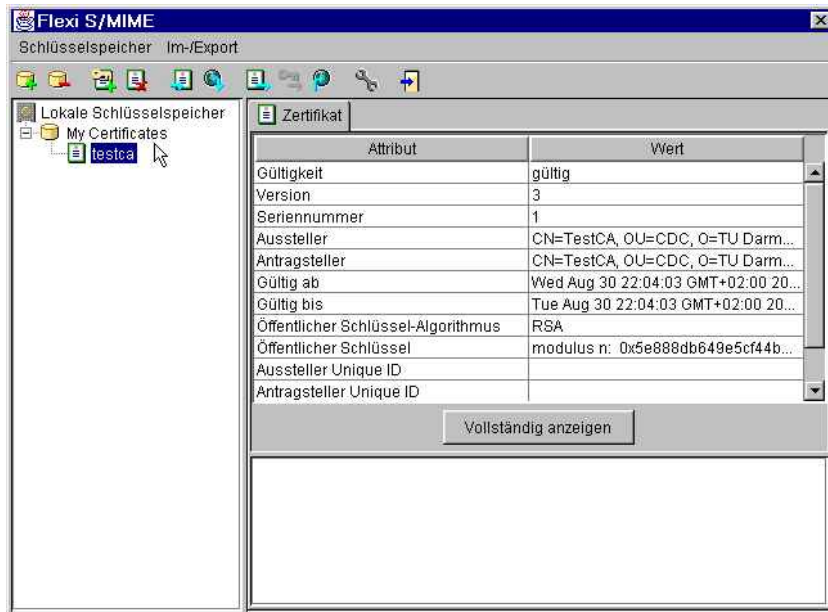


Abbildung 5: Mitgelieferter leerer KeyStore



Abbildung 6: Generierung eines Schlüssels

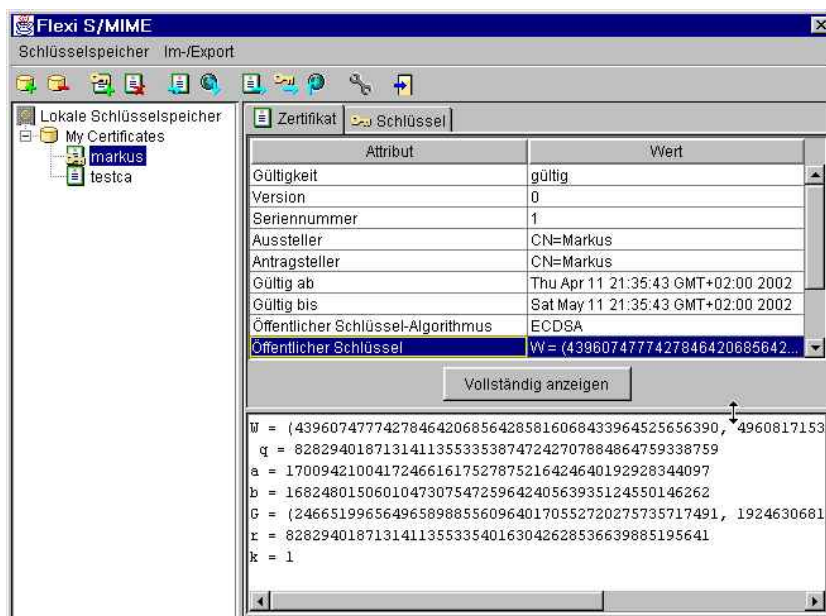


Abbildung 7: KeyStore mit selbstsigniertem Zertifikat

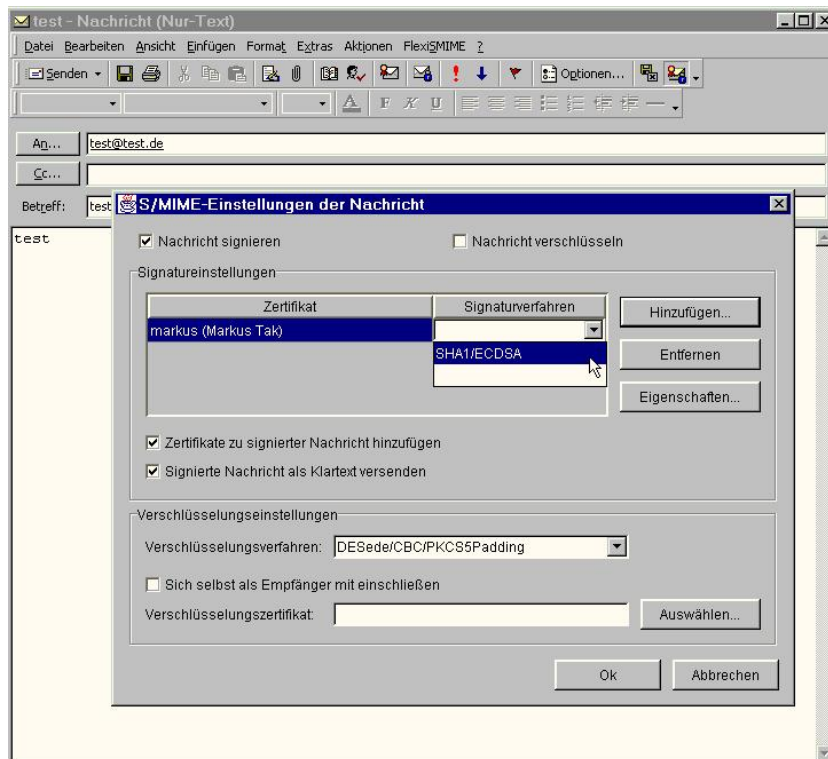


Abbildung 8: Signatur einer Email

Nun kann man den neuen Schlüssel etwa zum signieren einer Email benutzen. Dazu schreibt man wie gewohnt mit Outlook eine Mail und wählt vor dem Absenden im Menü *FlexiSMIME - SMIME-Eigenschaften der Nachricht* den Punkt *Nachricht signieren* aus, wie Abbildung 8 zeigt. Dabei muß sowohl das Signatur-Zertifikat als auch der Signatur-Algorithmus festgelegt werden. Es kann zu einem Signatur-Zertifikat ggf. mehrere Signatur-Algorithmen geben, etwa wenn mehrere Hashfunktionen zum Einsatz kommen. Falls Sie mehrere Signatur-Schlüssel haben, können Sie die Mail auch mehrfach signieren, auch mit verschiedenen Signatur-Verfahren.

Abbildung 9 zeigt den Empfang einer signierten Mail. Nach Auswahl des Menü-Punktes *FlexiSMIME - SMIME Eigenschaften der Nachricht* wird eine Zusammenfassung der Signatur-Prüfung angezeigt, wie in Abbildung 9 zu sehen.

Abbildung 10 zeigt schließlich die Verschlüsselung einer Mail. Nach der Erstellung der Mail und Auswahl des Menü-Punktes *FlexiSMIME - SMIME Eigenschaften der Nachricht* kann man den Verschlüsselungs-Algorithmus auswählen und ggf. ein Zertifikat zusätzlich angeben, an das zusätzlich zum Empfänger verschlüsselt wird, damit man die Mail später noch lesen kann.

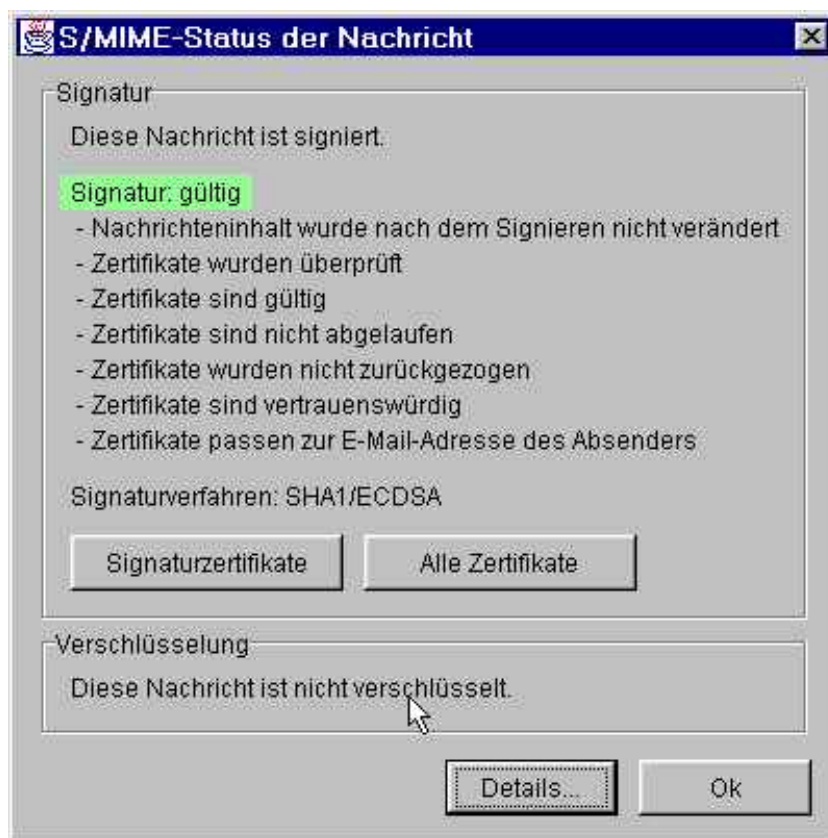


Abbildung 9: Ansicht einer signierten Email

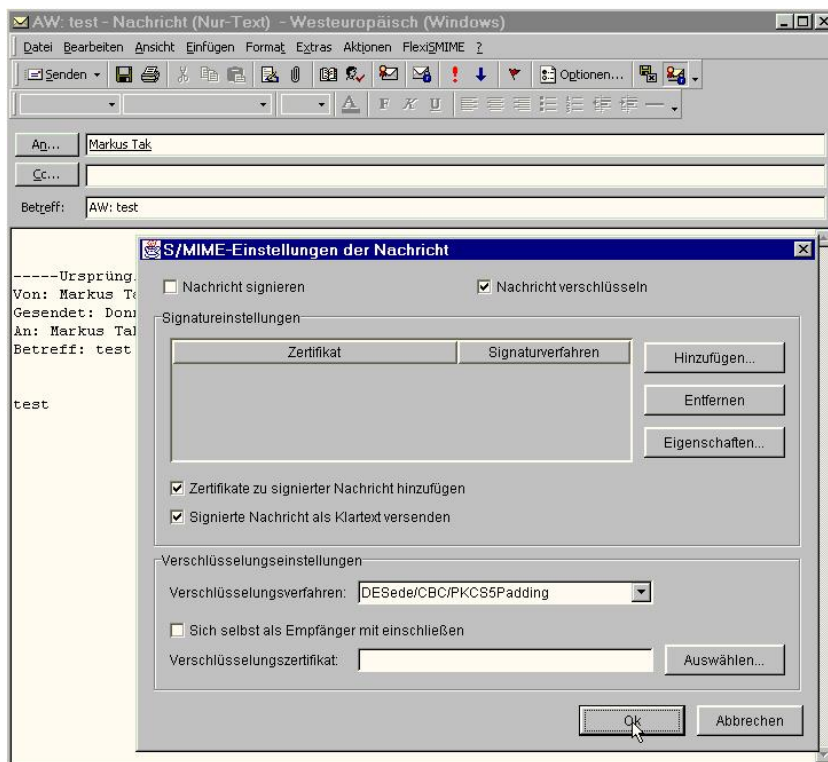


Abbildung 10: Verschlüsselung einer Email

## 5 Tests

Mit FlexiSMIME können signierte und verschlüsselte Nachrichten gesendet und empfangen werden. Die Tests wurden sowohl gegen die eigene Implementierung (FlexiSMIME) als auch gegen fremde S/MIME fähige Anwendungen gefahren. Fremde S/MIME Anwendungen unterstützen - außer im Falle von Secude AuthenteMail - keine alternativen kryptographischen Verfahren wie ECDSA oder IQDSA. Daher beschränken sich diese Tests nur auf die FlexiSMIME Implementierung.

Die folgenden Tests wurden mit anderen S/MIME Anwendungen erfolgreich durchgeführt:

- Verschlüsselte Emails zwischen FlexiSMIME und Netscape Messenger (RSA, DESede/CBC/PKCS5Padding, 168 Bit)
- Verschlüsselte Emails zwischen FlexiSMIME und Outlook Express (RSA, DESede/CBC/PKCS5Padding, 168 Bit)
- Signierte Emails zwischen FlexiSMIME und Netscape Messenger (SHA1/RSA und M5/RSA)
- Signierte Emails zwischen FlexiSMIME und Outlook Express (SHA1/RSA und M5/RSA)
- Signierte Emails zwischen FlexiSMIME und Netscape Messenger (SHA1/DSA<sup>4</sup>)
- Signierte Emails zwischen FlexiSMIME und Outlook Express (SHA1/DSA<sup>5</sup>)
- Signierte Emails zwischen FlexiSMIME und Secude AuthenteMail<sup>6</sup> (SHA1/ECDSA über GF(p))

Diese Tests wurden mit der FlexiSMIME Implementierung als Sender und Empfänger erfolgreich durchgeführt:

- Verschlüsselte Emails mit RSA (1024 Bit) und DESede/CBC/PKCS-5Padding (168 Bit)
- Verschlüsselte Emails mit RSA (1024 Bit) und IDEA/CBC/PKCS-5Padding (128 Bit)
- Signierte Emails mit RipeMD160/RSA, SHA1/RSA und M5/RSA (jeweils 1024 Bit)
- Signierte Emails mit SHA1/IQGQ und SHA1/IQRDSA (jeweils 800 Bit)
- Signierte Emails mit SHA1/DSA<sup>7</sup> (512 Bit)
- Signierte Emails mit SHA1/ECDSA über GF(p) (160 Bit)

---

<sup>4</sup>DSA-Implementierung des SUN-Providers

<sup>5</sup>DSA-Implementierung des SUN-Providers

<sup>6</sup>Es handelt sich um eine in Entwicklung befindliche Zwischenversion, basierend auf AuthenteMail 2.4.1

<sup>7</sup>DSA-Implementierung des SUN-Providers

Bei allen Tests wurde der FlexiProvider [FP] und der mit Java ausgelieferte SUN-Provider eingesetzt. Fremde JCA-Provider wurden bisher noch nicht ausgiebig getestet, da viele JCA-Provider die Anforderungen aus Abschnitt 3.2 nicht erfüllen, wie in [VR00] beschrieben.

## 6 Ausblick

Die folgenden Arbeiten stellen sinnvolle Erweiterung des Outlook Plugins dar:

- Erweiterung des KeyStoreManagers:
  - Automatischer Aufbau von Zertifikatsketten
  - CA-Zertifikate automatisch erkennen und ggf. in einem extra KeyStore verwalten (Vertrauensstatus)
  - Modell für eine einheitliche Zertifikatsprüfung unter Verwendung von OCSP und CRL's
- Geschwindigkeitsoptimierung bei Aufrufen der Zertifikatsverwaltung
- Verbesserung des Dialogs zur Schlüsselgenerierung:
  - Nicht alle Public Key Verfahren eignen sich für Signaturen, daher kann nicht immer ein selbst-signiertes Zertifikat erstellt werden (z.B. ElGamal-Varianten)
  - Verbesserung der Integration von Zertifizierungsprozessen, z.B. über das Certificate Management Protocol

## 7 Eingeflossene Arbeiten

Die folgenden Arbeiten sind in dieses Paper eingeflossen:

- [DS01]: Diplomarbeit von Dirk Schramm, in der das Outlook-Plugin weitestgehend entstanden ist. Ebenso die Schnittstelle *CertificateManager*, an die sich der KeyStore-Manager andockt.
- [VS01]: Praktikumsarbeit von Vladislav Satanovsky, die Urversion des KeyStore-Managers
- [DL02]: Praktikumsarbeit von Christof Leng und Thomas Dexheimer, Weiterentwicklung des KeyStore-Managers um Im- und Exportfunktionen und bessere Integration in das Outlook-Plugin
- *Der FlexiProvider [FP]*: Der FlexiProvider bildet das kryptographische Rückgrat des Outlook-Plugins. Er wird von der FlexiProvider-Gruppe gepflegt und steht unter [www.flexiprovider.de](http://www.flexiprovider.de) zum Download bereit.
- *CODEC*: Die von Volker Roth im Rahmen des SeMoA-Projekts entwickelte Bibliothek zur Kodierung von ASN.1 Datenstrukturen

Als verwandte Arbeit sei noch [KAL00] erwähnt. In der Diplomarbeit von Igor Kalenderian wird erstmals der reale Austausch von Krypto-Algorithmen in Form von JCA-Providern implementiert. Als Basis hierfür diente das in diesem Paper vorgestellte Outlook-Plugin.

Als externe Pakete wurden verwendet:

- Die *Cryptix-JCE*: Die Java Cryptography Extension<sup>8</sup> (JCE) ist bei der Verwendung eines JDK 1.2.x und 1.3.x notwendig, um starke Kryptographie innerhalb von Java überhaupt nutzen zu können. Derzeit entsteht im Rahmen des SeMoA-Projektes eine unabhängige Implementierung, die jedoch derzeit noch Probleme aufweist, so daß hier auf die bewährte Cryptix-JCE zurückgegriffen wurde.
- Die Netscape LDAP-Klasse (ldap40.jar). Dieses Paket wird mit dem frei erhältlichen Netscape Communicator installiert und enthält Klassen zur Implementierung des LDAP-Protokolls.

---

<sup>8</sup>Die JCE enthält die Klassen *javax.crypto.\**

## Literatur

- [BRT00] *FlexiPKI - Realisierung einer flexiblen Public-Key-Infrastruktur*, Johannes Buchmann, Markus Ruppert und Markus Tak, Tagungsband der Konferenz Systemsicherheit: Grundlagen, Konzepte, Realisierungen, Anwendungen, Vieweg Verlag, ISBN 3-528-05745-9, 2000
- [BUMA00] *Wie sicher ist die Public-Key Kryptographie?*, Johannes Buchmann und Markus Maurer, Tagungsband der Konferenz Systemsicherheit: Grundlagen, Konzepte, Realisierungen, Anwendungen, Vieweg Verlag, ISBN 3-528-05745-9, 2000
- [COR00] *Entwurf und Implementierung eines PKCS#11-Providers für die Java Crypto Architecture*, Diplomarbeit von Vlad Coroama, Technische Universität Darmstadt, März 2000
- [DL02] *Überarbeitung des KeyStoreManagers*, Praktikumsarbeit von Christof Leng und Thomas Dexheimer, Technische Universität Darmstadt, Sommersemester 2001
- [DS01] *Entwicklung einer flexiblen Java-basierten S/MIME-Erweiterung für Microsoft Outlook*, Diplomarbeit von Dirk Schramm, Technische Universität Darmstadt, 2000
- [EXCHEXT] *Extending the Microsoft Exchange Client*, Platform SDK Documentation, Microsoft Developer Network Library, July 2000
- [FP] *Der FlexiProvider*, [www.flexiprovider.de](http://www.flexiprovider.de)
- [HMA01] *Fail-Safe-Konzept für Public-Key-Infrastrukturen*, Technical Report, Michael Hartmann und Sönke Maseberg, TU Darmstadt, Januar 2002
- [NFC] *Number Field Cryptography*, <http://www.informatik.tu-darmstadt.de/TI/Forschung/nfc.html>
- [JAF] *JavaBeans Activation Framework Documentation*, <http://java.sun.com/products/javabeans/glasgow/jaf.html>
- [JAM] *JavaMail API Documentation*, <http://java.sun.com/products/javamail>
- [JNI] *The Java Native Interface (JNI)*, <http://java.sun.com/products/jdk/1.2/docs/guide/jni>
- [JNU] *Java in a Nutshell*, David Flanagan, O'Reilly 1998
- [KAL00] *Implementierung des Online Certificate Status Protocol*, Praktikumsarbeit von Igor Kalenderian, Technische Universität Darmstadt, 2000
- [KAL01] *Implementierung des Austausches kryptographischer Komponenten in FlexiPKI mittels Update Management Protocol*, Diplomarbeit von Igor Kalenderian, Technische Universität Darmstadt, November 2001
- [PK7] *Public Key Cryptography Standard #7 (PKCS#7)*, Rsa-Labs, [www.rsasecurity.com/rsalabs/pkcs](http://www.rsasecurity.com/rsalabs/pkcs)
- [RT01] *Sicherheitsmanagement durch generische objektorientierte Modellierung einer Trustcenter-Software*, Markus Ruppert und Markus Tak, Tagungsband der Konferenz Kommunikationssicherheit: Grundlagen, Strategien, Realisierungen, Anwendungen, Vieweg Verlag, ISBN 3-528-05763-7, 2001

- [SEMOA] *SeMoA - Secure Mobile Agents*, [www.semoa.org](http://www.semoa.org)
- [SUN] *Java Online-Dokumentation: How to implement a JCA provider*, [java.sun.com/products/jdk/1.2/docs/guide/security/HowToImplA-Provider.html](http://java.sun.com/products/jdk/1.2/docs/guide/security/HowToImplA-Provider.html)
- [SAM] *Secude AuthenteMail - Office Security Suite*, Secude Sicherheitssysteme GmbH, [www.secude.com](http://www.secude.com)
- [SSE] *SSE TrustedMime*, <http://www.sse.ie/trustedmime.html>
- [TR01] *JAVA-basierte Kryptographie wird interoperabel*, Technical Report, Vlad Coroama, Markus Ruppert, Michael Seipel und Markus Tak, Tagungsband der Konferenz Kommunikationssicherheit: Grundlagen, Strategien, Realisierungen, Anwendungen, Vieweg Verlag, ISBN 3-528-05763-7, 2001
- [VS01] *KeyStore-Manager: Ein Werkzeug zur Verwaltung von Java KeyStores*, Praktikumsarbeit von Vladislav Satanovski, 2001, Technische Universität Darmstadt
- [X509] *ITU-T RECOMMENDATION X.509, INTERNATIONAL STANDARD 9594-8*, Information Technology - OPEN SYSTEMS INTERCONNECTION - Directory Authentication Framework, August 1997
- [VR00] *Practical Aspects of the Java Security Architecture and Extension*, Volker Roth, Fraunhofer Institut für Graphische Datenverarbeitung, 2000