Abbildung 1: Eingesetzte Chipkarten

UNDINE - Abschlußbericht

Projekt "Universitäre Dienste im Internet"

Matthias Honka, 16.11.2002, matthias.honka@asknet.de

ask|net www.asknet.de
Vincenz-Prießnitz-Str. 3
76131 Karlsruhe
Fon: 0721-96458-0
Fax: 0721-96458-99

Einführung

Motivation

Software bietet sich wie kein zweites Produkt für den Handel im Internet an. Alle Prozeßschritte, von der Produktinformation, -auswahl, ggf. Genehmigungsworkflow, Bestellung, Lieferung und Bezahlung bis zum nachgelagerten Support durch Updates etc. können rein digital abgewickelt werden. In realen Internet-Shops oder Beschaffungsystemen werden diese Möglichkeiten jedoch noch relativ zögerlich eingesetzt. Hintergrund dafür sind bisher wenig verbreitete Sicherungstechnologien für das kopierbare Gut Software.

Unter diesen Voraussetzungen wurde das Projekt UNDINE entworfen. Sein Ziel ist alle Prozeßstufen der Softwarebeschaffung ohne Medienbruch über das Internet abwickeln zu können. Als zentrale Lösung für die vielfältigen Anforderungen bietet sich die Nutzung von Public Key Infrastrukturen (PKI) und Smartcards an. Zentrale Ziele des Projekts sind die prototypische Integration von PKI in eine Web-Applikation und das Sammeln von Erfahrungen im Einsatz beim Nutzer.

Der Handel mit Software unterscheidet sich in einem wesentlichen Punkt vom Handel mit anderen Waren: Das Produkt ist selbst digitale Information und damit elektronisch austauschbares Gut. Daher ist es möglich, die Softwarebeschaffung für Forschungseinrichtungen, Unternehmen oder Einzelkunden rein digital über das Internet abzuwickeln.

Um diese Ziele zu erreichen, sind verschiedene Hürden zu überwinden, die sich gerade aus der leichten elektronischen Verteilbarkeit von Software und aus den spezifischen Eigenarten des Onlinehandels ergeben. Hier sind zu nennen:

1. Betrugsmöglichkeit durch illegale Kopie und Nutzung von Softwareprodukten
2. Die direkte digitale Lieferung während des Einkaufs erfordert die direkt verbundene Bezahlung oder Rechnungsstellung.
3. Betrugsmöglichkeit im Bereich der verbreiteten elektronischen Onlinebezahlmethoden insbesondere bei Kreditkarten.
4. Verschiedene Lizenzbedingungen der Software-Hersteller, deren Einhaltung soweit als möglich durch die Händler gewährleistet werden muß, bspw. Studentenlizenzen, Campuslizenzen.
5. (Halb-)automatische Updatemöglichkeit von Produkten oder Produktteilen, bspw. Virusscanner.

Der aktuell noch am häufigsten auftretende Beschaffungsarbeitsablauf von Software für Einzelkunden oder Mitarbeiter in Institutionen ist in [Abb. 1] aufgezeigt. Die Graphik stellt nicht den Stand der Technik dar, sondern, wie gesagt, den Standardfall.

Bei ask|net werden bereits seit mehreren Jahren Softwaredownloads und verschiedene Online-Bezahltechnologien, die allerdings nur im Endkundenbereich eine Rolle spielen, angeboten.

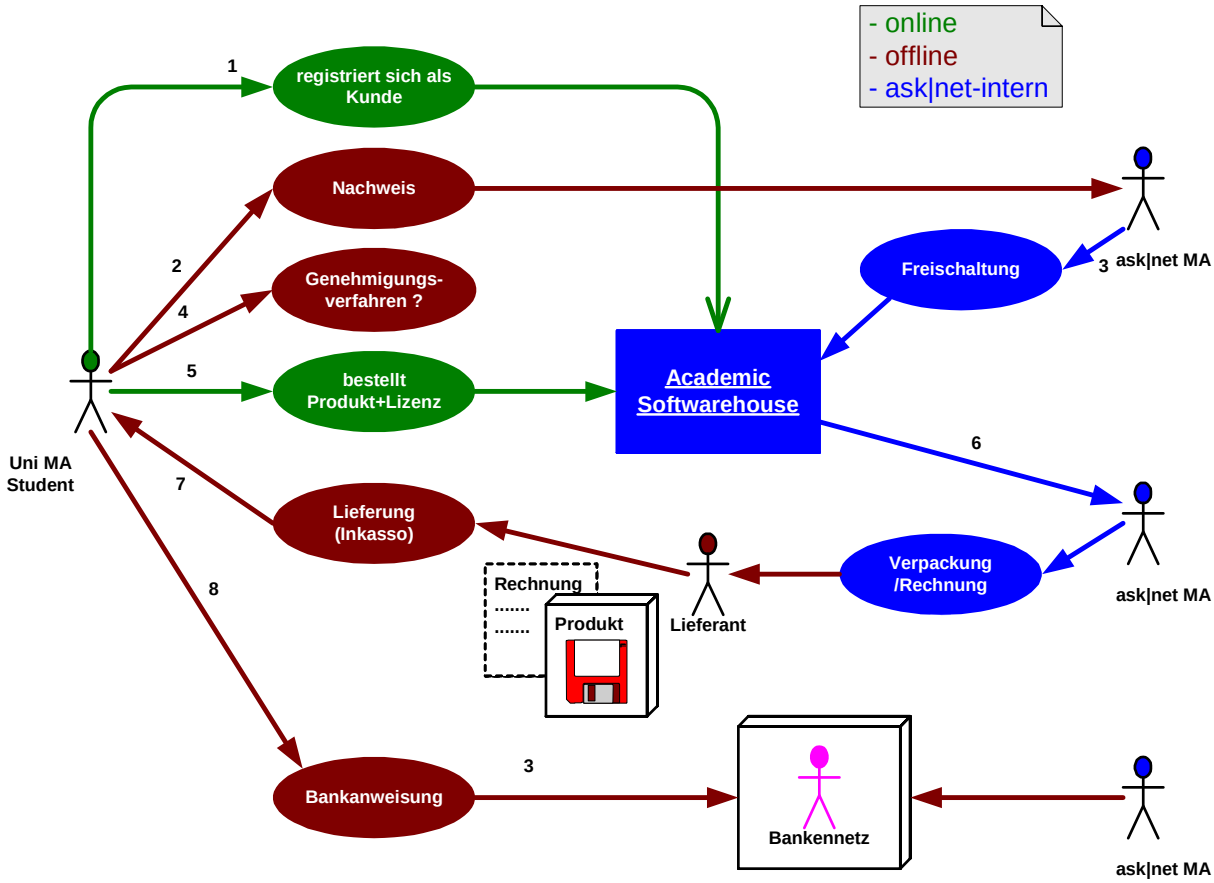


Abb. 2: Software-Einkaufs-Workflow

Die sich ergebenden Medienbrüche sind zu finden bei:

1. Authentifizierung und Autorisierung: Die in einem Webformular eingegeben Daten bei der Nutzerregistrierung müssen verifiziert werden, bevor die Freischaltung zum Kauf von Lizenzen aus einem Campus-Vertrag oder Studentenlizenzen möglich ist.
2. vorgelagerten Beschaffungsprozessen in Institutionen: Hier sind häufig noch Papier-Laufzettel zu finden, da Genehmigungschritte signiert werden müssen.
3. Lieferung von Produkt und Lizenz: Bei Paketlieferung ist ein Dienstleister in Anspruch zu nehmen, Adressierung und Verpackung geschehen z.T. von Hand. Der Lieferant kann ggf. auch das Inkasso übernehmen (Nachnahme).
4. Bezahlung per Rechnung: Die Rechnung liegt der Produktlieferung bei oder wird separat an eine Rechnungsadresse versandt. Die Bezahlung selbst erfolgt wieder durch Überweisung.

Digitale Zertifikate und Smartcards

Ohne auf den theoretischen und organisatorischen Hintergrund von Public-Key-Infrastrukturen und Hardwaretoken einzugehen, läßt sich der Nutzen für die Abwicklung von Internet-Geschäften mit Hilfe von publicKey-Kryptographie und Zertifikaten zusammenfassen:

Private und public Key bilden ein Schlüsselpaar, mit dem komplementäre Aufgaben wahrgenommen werden können.

- Zur Versendung vertraulicher Nachrichten wird der - möglichst zertifizierte - öffentliche Schlüssel des Adressaten zur Kodierung verwendet. Damit wird eine Entschlüsselung nur durch ihn möglich.

- Zum Nachweis der eigenen Identität oder Vertrauenswürdigkeit, bzw. zur Signatur von Dokumenten wird der eigene private Schlüssel verwendet.

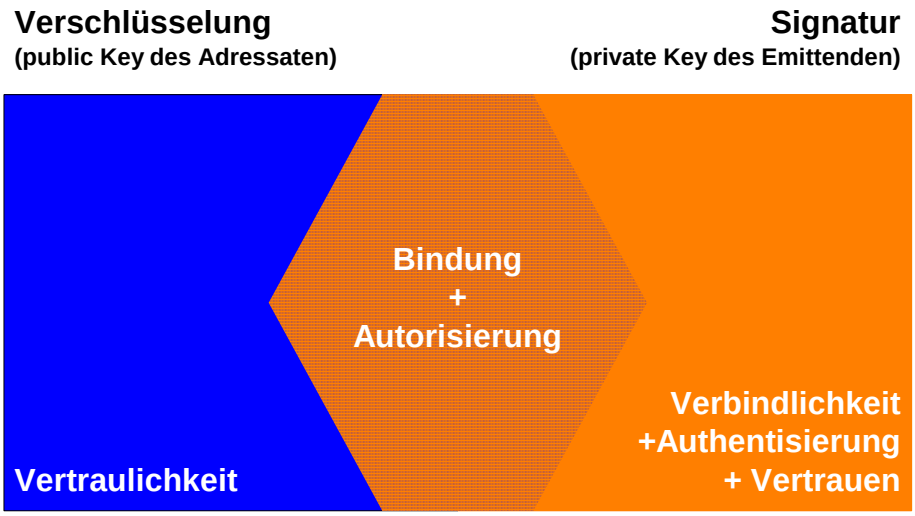


Abb. 2: Komplementäre Nutzung von public und private Key

- Beide Kodierungsverfahren zusammen ermöglichen die Autorisierung von Nutzern für exklusive digitale Dienste. Die automatische Authentifizierung läßt eine differenzierte Zugriffskontrolle zu und Verschlüsselung von Datenströmen verhindert die Ausspähung durch Dritte. Zudem können Datenlieferungen an den Adressaten gebunden werden, so daß Lieferung und Entpackung zertifikatsbasiert überprüft werden. Im extremsten Fall wäre sogar die Nutzung der Daten oder einer Software, wie etwa bei Digital-Rights-Managment- Systemen (DRMS), kontrollierbar.

PKI in der Softwarebeschaffung

Es wurden prototypisch Teillösungen implementiert und die Einsetzbarkeit und Integrierbarkeit vorhandener PKI-Lösungen in ein vorhandenes System, hier eine Web-Shop-Applikation mit gegebenem Workflow, erprobt und realisiert.

Die verschiedenen Sachverhalte bei der digitalen Softwarebeschaffung und den dortigen Einsatzmöglichkeiten von PKI lassen sich grob in folgende Segmente mit absteigender Bedeutung einteilen:

Segment	Beschreibung und Anforderungen
Authentifizierung Autorisierung der Kunden	Kunden müssen für verschiedene Produkte oder Dienstleistungen ausreichend und vertrauenswürdig identifiziert sein. Lieferung von bestimmten Produkten ist nur an bestimmte Nutzerkreise möglich (Studentenlizenz, Updaterechte). • Nachweis der Identität und Zugehörigkeit (Immatrikulationbescheinigung)
Lieferung von Produkten, Lizenzen	Sofortige Lieferung von digitalen Produkten und Lizenzen nach der Bestellung oder Bezahlung über das Internet. • Sicherung des Lieferkanals, ggf. Bindung der Entpackung und Installation an den Käufer oder Hardware des Käufers durch Verschlüsselung von Produkten oder Archiven.

Abrechnung Bezahlung	Online-Bezahl-Technologien sollen die Bezahlung vor der Lieferung gewährleisten oder digitale Abrechnungsmethoden sollen anwendbar sein. • Vertrauenswürdige Sofortbezahlung (digitales Bargeld ?) • Versendung von digitalen Rechnungen
Zeichnung von Vertragsbedingungen	Kunden müssen ggf. vor dem Kauf über best. Rahmenbedingungen für die Produktnutzung informiert werden und sie akzeptieren. • Signierung von Vertrags/Lizenzbedingungen.
Beschaffungswesen	Bisherige Offline-Beschaffungsprozesse im Vorfeld des Kaufs per Laufzettel sollen als Dienstleistung online ermöglicht werden • Digitale Signierung von Genehmigungen
Andere Nutzungsmodelle für Software	Kontrolle der Software über die Lieferung bis hin zur Nutzung • Pay-per-use, Try-before-buy mit Sandboxtechnologie (geringe Akzeptanz, Durchdringung) • Application-Service-Providing

Erprobte und umgesetzte Lösungen

Um alle Varianten des Zertfikatseinsatzes austesten zu können hat sich die ask|net beim Bau der Prototypen auf vier Sektoren [Abb. 3] konzentriert:

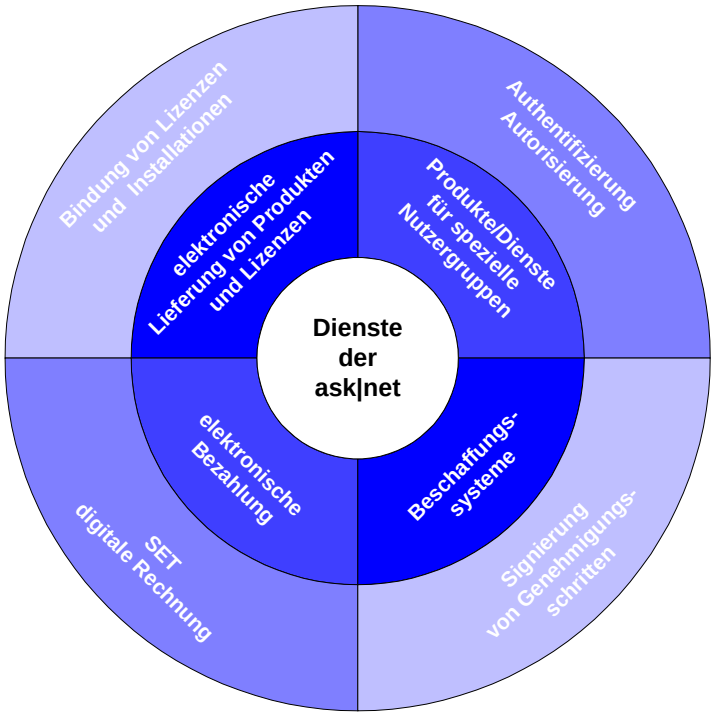


Abb. 3: Dienste die zertifikatsbasiert implementiert wurden

Somit sind alle Fälle von Diensten, die Verschlüsselung, Zertifikate der Kunden oder vom Dienstleister ask|net erfordern, abgebildet.

Kunden-Authentifizierung

Die Authentifizierung von Kunden bei der Registrierung oder beim Anmelden erfolgt direkt unter Ausnutzung der Möglichkeiten für Clientauthentifizierung von Browsern. Für spezielle Webseiten wird im Webserver eine Clientauthentifizierung vorgeschrieben.
Der Nutzer wird beim Anwählen der Seite auf die notwendige Authentifizierung aufmerksam gemacht und kann frei zwischen software- oder hardwarebasierten digitalen Zertifikaten wählen.

Voraussetzung für Hardwarezertifikate ist die Installation eines Smartcardreader mit Treibersoftware und entsprechendem Browserplugin. Bei Netscape-Browsern sind dies pkcs11-Module, bei Microsoft muß der Smartcardhersteller für seine Karten sog. Crypto-Service-Provider für die MS-Crypto-API des Betriebsystems bereitstellen.

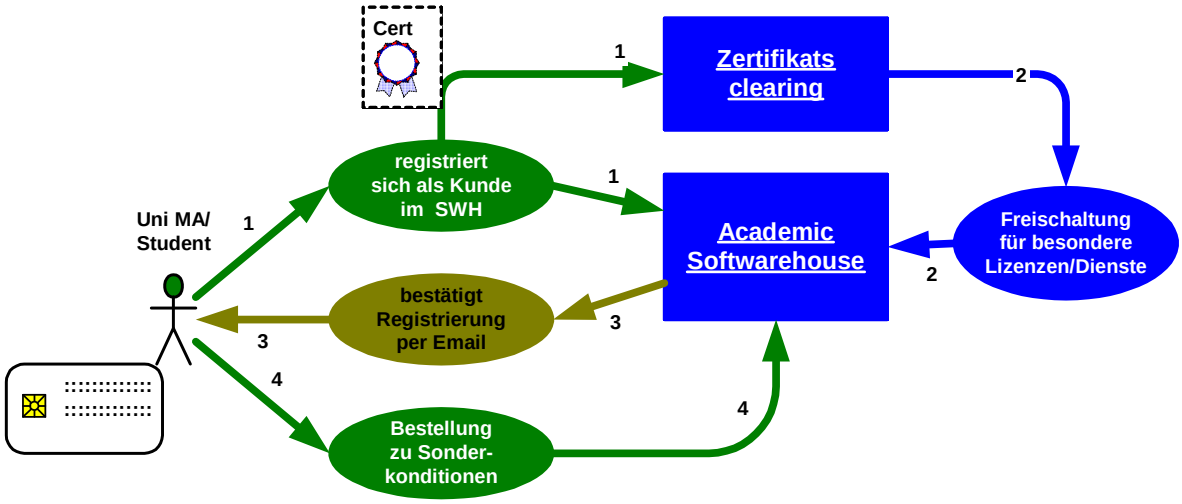


Abb. 4: Ablauf der automatischen Autorisierung für Dienste

Ist aus Zertifikatsaussteller oder Zertfikatsattributen die Zugehörigkeit des Nutzers zu bestimmten Gruppen feststellbar, so kann er automatisch für Dienste oder für den Kauf besonderer Produktlizenzen - wie die o.g. Studentenlizenzen freigeschaltet [Abb. 4] werden.

Abb. 5: Erfolgtes SSL-Login mit digitalem Zertifikat

Digitale Bezahlung und Abrechnung

Digitale Bezahltechnologien werden von ask|net bereits seit langem angeboten. Zur Zeit stehen folgende Methoden zur Bezahlung in den ask|net-Shops zur Verfügung:

1. per Rechnung für vorab autorisierte Kunden
2. Zahlung per Kreditkarte (Visa/Mastercard)
3. SET (mit Kundenzertifikat)
4. Bezahlung per Handy (Paybox)

Die einzige Technologie die auf digitale Zertifikate zurückgreifen könnte, ist SET. Die derzeitige Nutzung ist aber sehr gering. Außerdem sind SET-Zertifikate bisher nur für diesen einzigen Anwendungszweck für Endanwender einsetzbar. Die anderen angebotenen Methoden, werden von den Nutzern ibs. den Einzelkunden durch ihre weitaus einfachere Handhabung bevorzugt.

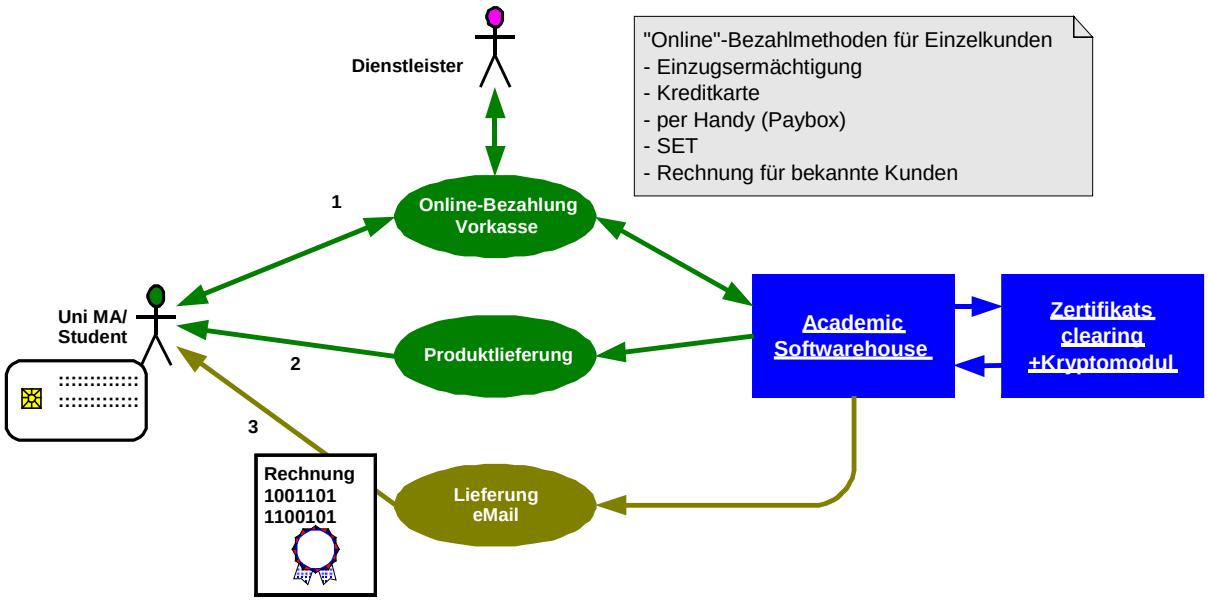


Abb. 6: Ablauf digitaler Bezahlung

Für Großkunden aus dem Bereich Forschung und Lehre kommt aus organisatorischen Gründen nur die Bezahlung per Rechnung in Frage. Hier ist eine Lösung, durch automatischen Versand von digital signierten Rechnungsemails [Abb. 6] denkbar.

Das Signaturgesetz in der aktuellen Fassung fordert hier allerdings die digitale Signatur für Rechnungen nach den höchsten Sicherheitsstandards, d.h. eine Signatur durch eine **natürliche Person**. Dies steht dem ursprünglichen Automatisierungsgedanken entgegen. Ggf. kann durch Zusatzvereinbarungen mit Großkunden auch auf Signaturen juristischer Personen umgestellt werden.

Digitale Produkt Lieferungen

Die sichere digitale Lieferung ist ein sensitiver Aspekt, der bei den Software-Herstellern eine große Rolle für die Vergabe von digitalen Distributionsrechten spielt. Von manchen wird sogar eine technische Lösung für der Kontrolle der Softwarenutzung erwartet. Solche Lösungen wurden im Projekt ebenfalls diskutiert und verworfen:

1. Sandboxverfahren: Vor der Installation der eigentlichen Software muß ein Sandboxsystem auf dem Kundenrechner installiert sein. Dies erfordert tiefe Eingriffe in das Betriebsystem des Kunden, die größtenteils abgelehnt werden.
2. Application-Service-Providing (ASP): Die Software wird als entfernter Service dem Kunden über Internet bereitgestellt und nur die Bedienoberfläche auf Mietvertragsbasis zum Nutzer übertragen. Dies scheitert z.T. an den erforderlichen Bandbreiten und an sicherheitstechnischen Überlegungen, da alle Kundendaten auf den Servern des ASP-Anbieters liegen.

Durchsetzbar und umgesetzt sind Technologien die soweit wie möglich die Kopie von Installationssoftware verhindert oder nutzlos macht. Bei ask|net ist eine zweischichtige Schutzhülle für Installationspakete entwickelt worden [Abb. 7].

1. Produkte werden per HTTP im SSL-Kanal übertragen.
2. Produkte können zusätzlich als verschlüsseltes selbstentpackendes Archiv (Box of Bits) übertragen werden.

Standardmäßig wird in den ask|net-Shops ein Kundenschlüssel bei der Registrierung des Kunden erzeugt und diesem per Email zugesandt.

In UNDINE ist es zusätzlich möglich, für jeden Download Schlüssel zu generieren und diese Kunden, die über ein digitales Zertifikat verfügen, mit ihrem öffentlichen Schlüssel kodiert per Email zuzusenden.

Als weitere Sicherheitsstufe ist es möglich, die Entpackung des Archives direkt an das Zertifikat zu koppeln, so daß ein Nutzer nur mit seiner Smartcard oder einem installierten Softwarezertifikat das Paket öffnen kann.

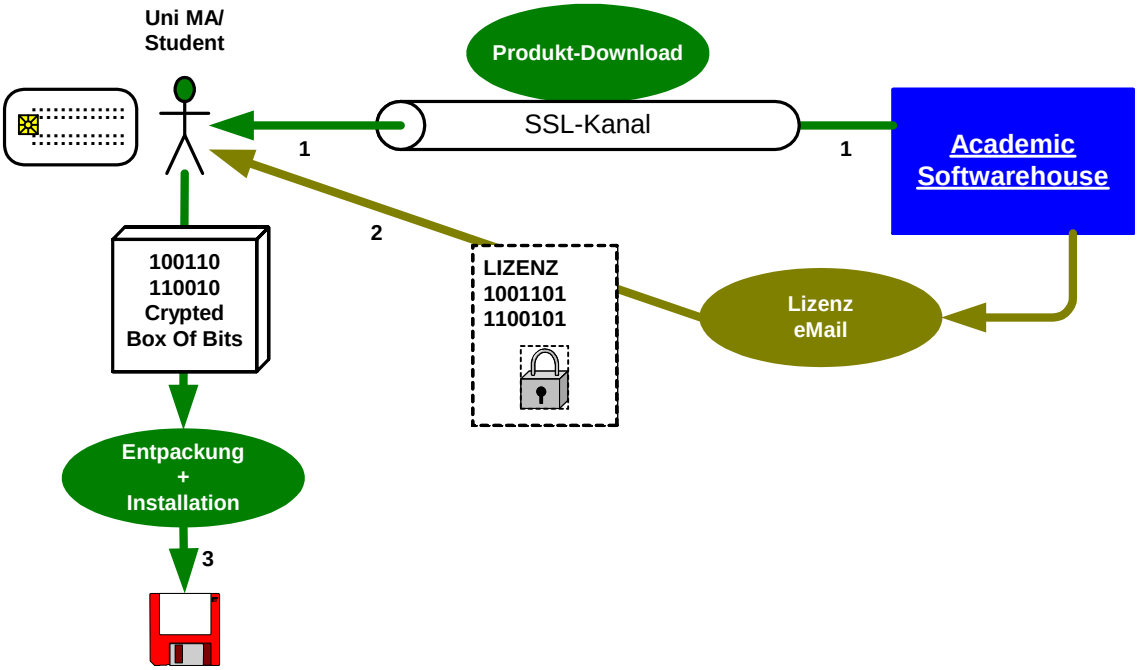


Abb. 7: Sichere Signierte Produktlieferung

So werden Lieferpakete direkt an einen Kunden gebunden, bei dem die Weitergabe eines persönlichen Schlüssels riskant oder mit der gleichzeitigen Weitergabe eigener besonderer Privilegien verbunden sein kann.

Diese Art der Lieferung sicherer Archive verhindert nicht die Kopie vorhandener Installationen. Sie dienen u.a.

1. als Hilfe für System-Administratoren, die die Verteilung von gekaufter Software in einem Unternehmen kontrollieren und sich so um den Schutz der Original-Installationspakete nicht selbst kümmern müssen.
2. Als Kopierschutz gegenüber technisch weniger versierten Nutzern, die keine Kopie von Software ohne Installationspaket durchführen können.

Beschaffungssystem

Digitale Signaturen bieten sich hier besonders an. Moderne Beschaffungsysteme (eProcurementsysteme) erlauben schon länger solche Arbeitsabläufe in einer Firma online per Knopfdruck abzuwickeln. Ein Nutzer, der Software anfordert, muß sich nicht mehr selbst um die Genehmigungen kümmern und kann jederzeit Einblick in den Fortgang des Genehmingungsprozesses nehmen. Ask|net bietet ein eProcurementsystem für Großkunden als Web-Applikation an, welches direkt in vorhandene Beschaffungsportale integriert werden kann.

In der Abbildung 8 ist der schematische Ablauf einer Genehmigung dargestellt. Ein Nutzer stellt seine gewünschte Bestellung ins System, Genehmigungsinstanzen können automatisch per Email informiert werden und Ihre Entscheidung in einem HTML-Formular eintragen.

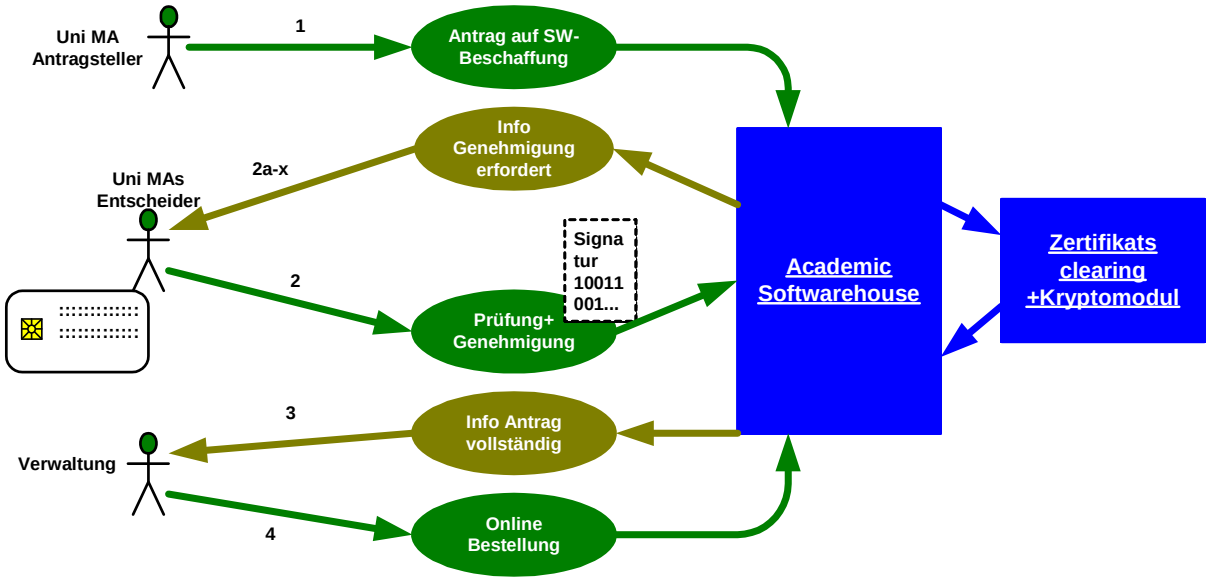


Abb. 8: Ablauf eines Genehmigungsprozesses

Erhält ein Genehmiger per Email eine Aufforderung zur Genehmigung, so sieht er bei Aufruf der entsprechenden Seite [Abb. 9] im ask|net-eProcurementsystem den Warenkorb des Anforderers und die bisher durch andere erteilten Genehmigungen. Die Integration einer Signierkomponente erfolgt in Form eines Java-Applets, welches zur Signierung mit Smartcards genutzt werden kann.

Die Signatur seiner Entscheidung führt er mit dem ask|net-Signier-Applet [Abb. 10] durch. Dieses greift über eine pkcs11-Schnittstelle auf die installierten Smartcardreader und Smartcards zu. Es benötigt dementsprechend hohe Systemrechte.

Abb. 9: Ausschnitt aus einem virtuellen Laufzettel

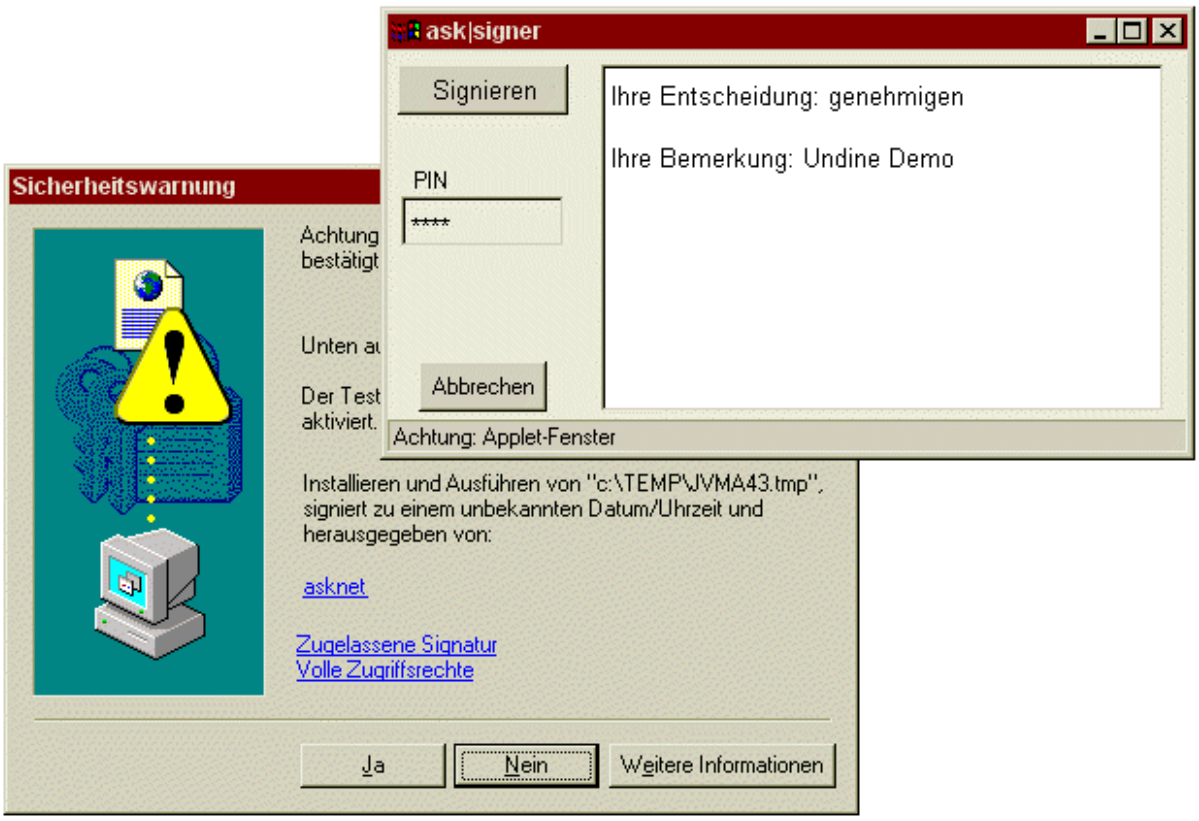


Abb. 10: Signierung eines Genehmigungschrittes per Signierapplet und Smartcard

Technologie

Server Architektur

Der Umbau der serverseitigen Applikationen erforderte den Einbau einer Zwischenschicht, den sog. "Request Broker", der die mitgelieferten Informationen zu Nutzerzertifikaten auswertet und je nach Anforderung des Nutzers die Autorisierung bzw. die Signatur prüft bzw. bei den ausgelieferten Daten zusätzliche Ver-/Entschlüsselung oder Signierung durch ask|net vornimmt [Abb. 11].

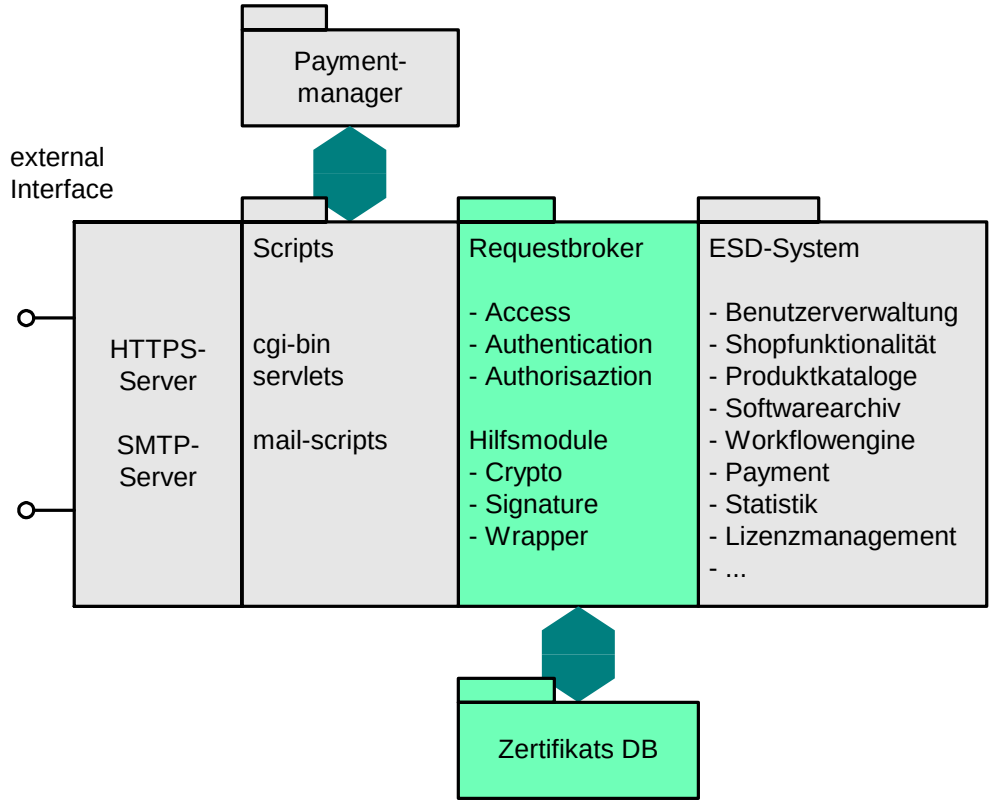


Abb. 11: Erweiterte Architektur der Digital Logistik Engine

Diese so entworfene Architektur sollte die notwendigen Änderungen am System so gering wie möglich halten und vor- oder nachgelagerte Prozesse der digitalen Logistik weitgehend unbeeinflußt lassen. Die Realisierung prototypischer Prozesse erfolgte in perl, C und Java. Da die Serverarchitektur inklusive Quellcode vollkommen unter der Kontrolle der ask|net steht, waren die notwendigen Anpassungen relativ leicht im System zu implementieren.

Client Architektur

Die Clients erforderten wesentlich mehr Aufmerksamkeit als zu Projektbeginn angenommen. Die ursprüngliche Annahme, daß gängige Browser die Online-Signierung von Dokumenten wie für Beschaffungswesen oder Zeichnung von Lizenzverträgen direkt unterstützen, mußte verworfen werden. Standardmäßig wird von Browsern nur SSL-basierte Clientauthentifizierung unterstützt. Dabei verwenden die Browser Internet Explorer von Microsoft und Communicator von Netscape bzw. in einer eigenen Browserdatenbank abgelegte Software-Zertifikate.

Zusätzlich besteht über die Microsoft-Crytpto-API bzw. über pkcs11-Plugins von Netscape die Möglichkeit, Nutzerzertifikate auf Smartcards anzusprechen und sich damit zu

authentifizieren. Keine aktuelle Version der beiden Browser ermöglicht direkt die Signatur von Dokumenten, die von einem Server heruntergeladen wurden.

Einige Überlegungen für praktische Lösungen zielten auf die Nutzung von Infrastruktur, die von den Karten- und Kartenleser-Herstellern bei der Installation bereitgestellt werden. Insbesondere die Nutzung von "sicheren Betrachtungskomponenten", die die komplette Kommunikation mit Signaturkarten selbst abwickeln, wurde eruiert.

Die Darstellungssoftware, die von Smartcard-Herausgebern mitgeliefert wird, ist jedoch oft nur mit deren Gesamt-Lösungen für Großkunden kompatibel oder bestenfalls in **Mail**programme, wie Outlook oder Lotus Notes, als Pluginmodul einzubinden [Abb. 12].

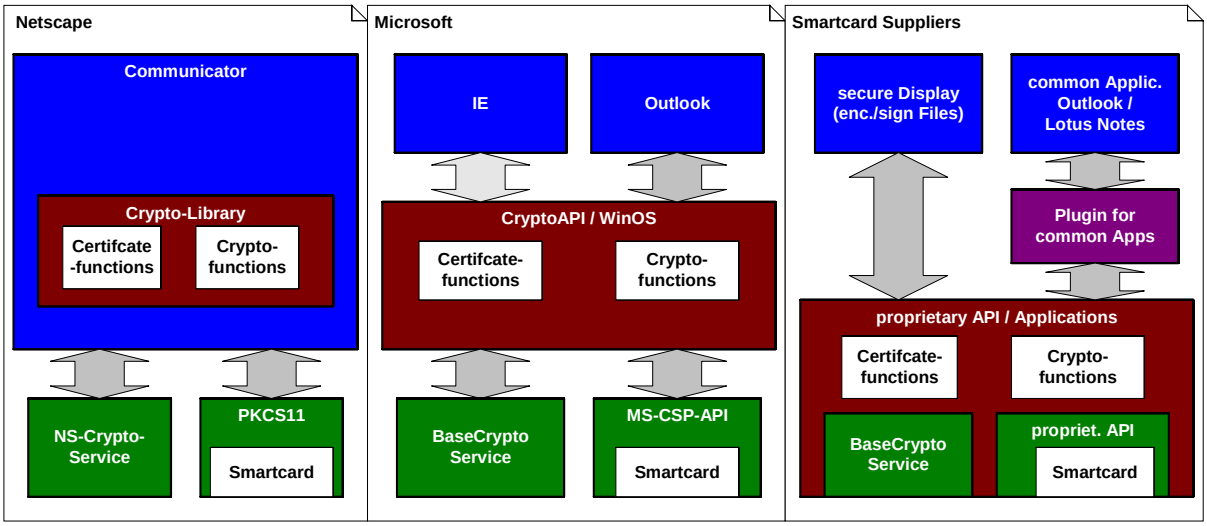


Abbildung 12: Verschiedene Architekturen für Smartcardzugriff

Damit stellte sich das Problem des Einsatzes einer eigenen Darstellungskomponente, die zum einen in den gängigen Browsern funktioniert und zum anderen leicht in verschiedene Abläufe integrierbar sein sollte. Daher wurde eine eigene Darstellungskomponente (siehe Zusatzpaket B), zur Ermöglichung von elektronischer Signierung im Browser entwickelt.

Um die Browser signaturfähig zu machen, sind zwei verschiedene Ansätze möglich:

Native Browsererweiterung (Plugin/ActiveX)	**Java Applet**
• Download + Installation erforderlich • Implementierungsaufwand: Varianten für IE und Netscape	• VM meist vorhanden • Sandbox: Zugriff auf Systemresourcen (Kartentreiber) muß vom Nutzer freigegeben werden

Die verschiedenen Standards und Bibliotheken, die verfügbar sind um die beiden Konzepte umzusetzen, sowie ihre Vor und Nachteile sind kurz aufgelistet:

Interface	Beschreibung	Nachteile	Vorteile
Native Library	Jeweilige API des Karten-treibers der versch. Karten-hersteller	Großer Aufwand durch breite Kartenvielfalt Software-Installation	Verläßlichkeit, Gut dokumentiert
PC/SC	Standard von Microsoft Auch Linux durch MUSCLE-Projekt. Framework zur Integration von Kryptotoken		Verläßlichkeit Dokumentaion Sicherheit
ActiveX + CryptoAPI	Entwicklung einer ActiveX-Komponente, die Javascript gesteuert die Signierung von HTML-Form-Inhalten ermöglicht	Nur für Windows/IE Installation Sicherheitsfragen	
JavaApplet + OpenCard-Framework	JavaApplet zur Anzeige und Signierung, Interface zur Karte über OCF	Keine ausreichende Unterstützung für OCF von versch. Kartenherstellern	Pure Java=> Portierbarkeit. OCF für versch. Kartenleser verfügbar.
JavaApplet + Pkcs11	JavaApplet zur Anzeige und Signierung, Interface zur Karte über JNI auf native pkcs11-Library.	PKCS11-API nicht für alle Karten implementiert Sicherheitsfragen	Java-VM-Plugin für IE und NS vorhanden, Standardisiert

Für die Entscheidung ein JavaApplet mit JNI (Java Native Interface) mit einem Pkcs11-Modul zu verbinden, wurden folgende Anforderungen berücksichtigt:

- Implementierungsaufwand
- Nutzerfreundlichkeit
- Leichte Integration in Webapplikation

Sicherheit

Der Aspekt Sicherheit wurde im Rahmen der Aufgabenstellung von UNDINE, die sich primär mit dem Sammeln von Erfahrungen bei der Integration von elektronischer Signatur in einen vorhandenen Workflow befasst, bewußt mit geringer Priorität behandelt. Der prototypische Charakter des Projektes und nicht zuletzt die Frage der Sinnhaftigkeit selbst hoch sichere Darstellungskomponenten auf den derzeitig verbreiteten Plattformen zu erstellen, spielte ebenso eine Rolle.

Trotzdem haben gerade die Überlegungen zum Thema Sicherheit für den Nutzer einige interessante Aspekte vorhandener Systeme zu Tage gebracht. Zur Veranschaulichung sind eine ideale und eine reale Architektur auf einem PC mit Signaturkomponente gegenübergestellt [Abb 13].

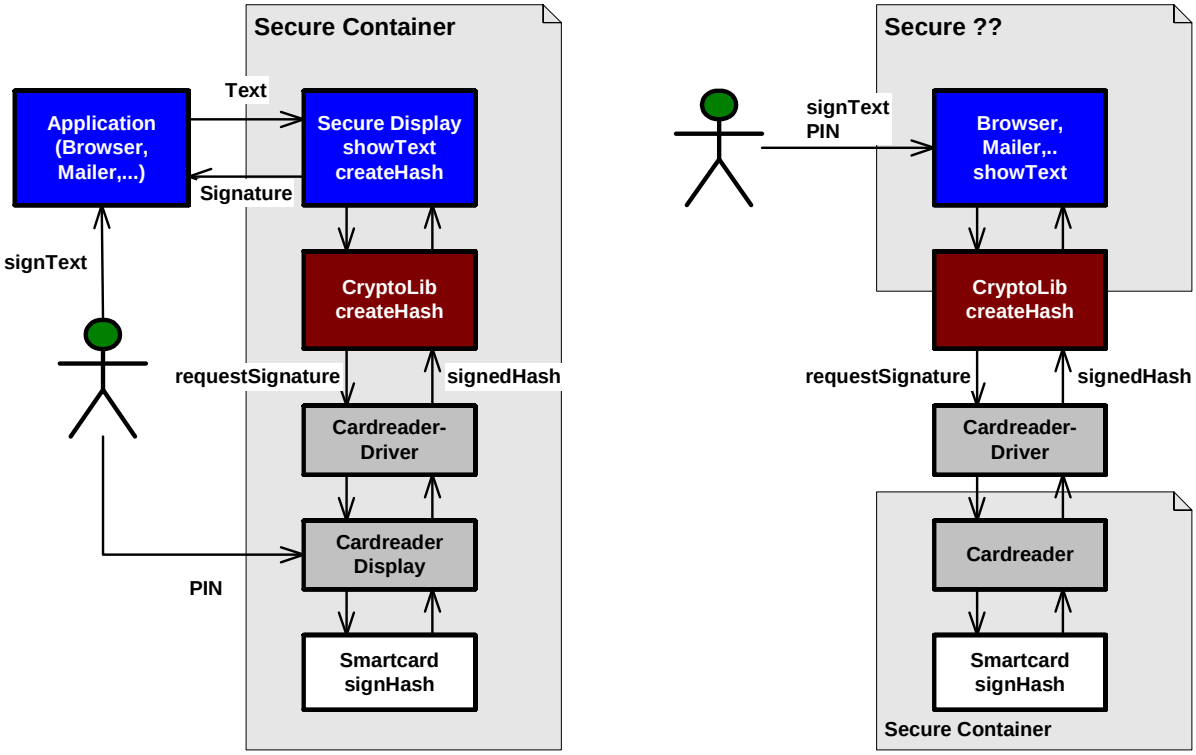


Abb. 12: Ideale und reale Infrastruktur auf dem Client

Um die höchste Sicherheit für den Nutzer zu gewährleisten ist die Implementierung der links dargestellten Struktur erforderlich. Der Nutzer sollte in die Lage versetzt werden aus beliebigen Applikationen heraus beliebige Dokumente zu signieren. Dies könnte durch Übergabe eines Dokumentes an eine "sichere Darstellungskomponente" geschehen. Damit die Kommunikation zwischen Darstellungskomponente und Signaturkarte nicht durch Trojaner auf dem PC kompromittierbar wird, muß die Darstellungskomponente zusammen mit Kartentreiber, Kartenleser und Karte einen sicheren Kontainer bilden. Hierzu sind signierte, also durch den Nutzer überprüfbar unveränderbare Softwarekomponenten sowie verschlüsselte Kommunikation zwischen den Soft- und Hardwarekomponenten erforderlich.

In derzeitigen Praxis sind die Applikationen selbst die Darstellungskomponenten, die häufig leicht kompromittierbar sind. Wie auch durch z.B. verbreitete Macroviren o.a. am Beispiel Outlook leicht deutlich wird. Diese Applikationen kommunizieren wiederum über die nicht kryptografisch gesicherte Betriebsystemebene mit den Harwaretoken. Außerdem kommt hinzu, daß Kartenleser mit eigener PIN-Eingabemöglichkeit aus Kostengründen noch wenig Verbreitung gefunden haben.

Damit sind verschiedene Angriffe möglich:
1. Der Nutzer unterzeichnet ein anderes Dokument als ihm angezeigt wird
2. Die PIN-Eingabe über die PC-Tastatur des Nutzers wird durch Trojanersoftware abhörbar und nutzbar
3. Beides zusammen ermöglicht die unbemerkte Signatur beliebiger Dokumente mit dem privaten Schlüssel des Nutzers.

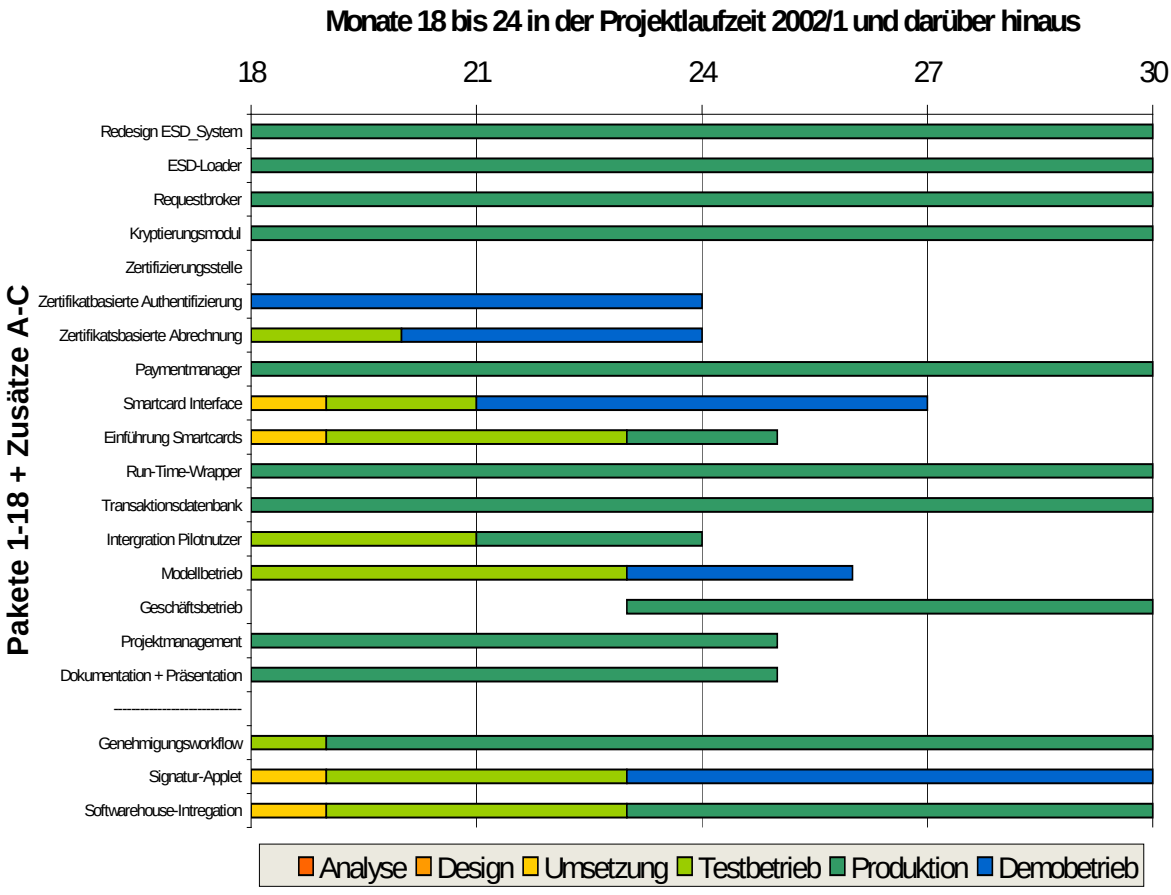
Aktivitäten 2002

Die Arbeiten konzentrierten sich im letzten Halbjahr auf folgende Punkte:

1. Produkt-/Module-Integration in vorhandene produktive Online-Shops, soweit möglich und sinnvoll.
2. Sammeln von Erfahrung bei der Einrichtung und Nutzung von signaturfähigen Clients im praktischen Einsatz mit ausgewählten Testnutzern.
3. Analyse von alternativen Möglichkeiten des Smartcardeinsatzes in Webapplikationen
4. Konzeption eines leicht integrierbaren Signatur-Applet-Protoptyps.
5. Entwicklung, Test und prototypischer Einsatz eines Smartcardinterface für Java-Applets im produktiven Umfeld.

Bei der Entscheidung über den Grad der produktiven Integration verschiedener Komponenten gaben, die bisher geringe Verbreitung von Kartenlesern an den Client-PCs bzw. die schwach ausgeprägte Ausstattung der Kunden mit Smardcards, den Ausschlag.

Stand Projektplan



Erreichte Projektziele

Generell läßt sich sagen, daß alle Ziele technisch und organisatorisch erreicht wurden. Fast alle geplanten Module konnten in vollem Umfang oder darüber hinaus umgesetzt, getestet und schließlich produktiv eingesetzt werden.

Beim Paket 13 "Integration Pilotnutzer", welches das Ziel eines Feldtestes einer Applikation unter realen Einsatzbedingungen beinhaltet, mußte eine Schwerpunktverlagerung zu Testnutzern aus dem Mitarbeiterkreis der ask|net erfolgen.

Bei der Projektumsetzung ist deutlich geworden, welche Diskrepanz noch herrscht zwischen der zum Zeitpunkt der Projektentwurf erwarteten und der tatsächlich eingetretenen Verbreitung von Smartcards mit Kryptoprozessor. Viele Universitäten führen z.Z. Smartcards für Mitarbeiter als auch für Studierende ein - jedoch enthalten diese aus Kostengründen keine Kryptofunktion.

Damit entfällt die Möglichkeit in einem Feldversuch mit größerer Nutzeranzahl die Praktikabilität von massenweise eingesetzter Online-Signatur oder an Kundenzertifikate gebundene Produktlieferung genau zu prüfen. Diese Aspekte konnten nur exemplarisch untersucht werden.

Zum anderen ist während des Projektes aufgefallen, wie wenig ausgereift einige Produkte bzw. wie wenig interoperabel kommerzielle Produkte verschiedener Hersteller sind. Dies wurde durch ursprünglich nicht vorgesehene Eigenentwicklungen wettgemacht. (Pakete A,B,C).

Status und Nutzungsgrad der entwickelten Pakete

Projektpakete laut Projektentwurf

Paket 1: Redesign ESD-System (produktiv)

Ziel: Anpassung der Shop-Infrastruktur der ask|net die zusätzlichen und geänderten Anforderung zur technischen Integration der folgenden Komponenten in eine Test- und Demosystem.

Die im Quartal 2000/2 entwickelte und im Bericht 1 aufgezeigte Zwischenschicht (Paket 3) in der Architektur der Online-Portale ist voll integriert. Die enthaltenen Komponenten können einzeln je nach Bedarf aktiviert werden. Die Nutzung der Zwischenschicht ist vollkommen transparent, d.h. ihre Nicht-/Nutzung führt nicht zu Ablaufänderungen in den Produktionsmodulen der Online-Portale.

Paket 2: ESD-Loader (produktiv)

Ziel: Entwicklung eines Plattform übergreifend nutzbaren Hilfsprogramm zum Download von elektronischen Waren und zur wiederaufnahme Abgebrochener Downloads.

Der ESD-Loader oder Downloadmanager ist als Java-Applet realsiert und in verschieden Portalen produktiv und unterstützt Nutzer beim herunterladen von gekaufter Software von unseren Servern. Abgebrochene Downloads können an der Abbruchstelle wieder aufgegriffen werden auch in verschiedenen Sitzungen.

Paket 3: Requestbroker (produktiv)

Ziel: Entwicklung eine konfigurierbaren Zwischenschicht zwischen Webserver und Business-Modulen, die vorhandene Funktionalität weitgehend unberührt läßt und die bei Bedarf vor- oder nachgelagerte Krypto- oder Signatur-Aufgaben wahrnimmt.

Siehe Paket 1.

Paket 4: Kryptierungsmodul (produktiv)

Ziel: Entwicklung eines Verschlüsselungs und Signatur-Moduls, welches halb-automatisch vom Requestbroker aktivert wird zur Verschlüsselung von Kommunikation, Produktlieferungen oder serverseitiger Signatur von Dokumenten.

Das Kryptierungsmodul basierend auf Openssl (www.openssl.org) wird an verschieden Stellen produktiv eingesetzt:

1. Verschlüsselung der Kommunikation mit dem Endkunden (HTTPS) (nur Serverauthentifizierung).
2. Verschlüsselung der Kommunikation mit Partnern (HTTPS) mit Clientauthentifizierung.
3. Ver- und Entschlüsselung von Digitalen Lieferungen im serverseitigen Download-Modul und dem heruntergeladenen Entpacker (siehe ESD-Loader und Runtimewrapper).
4. automatische Signierung von eMails

Paket 5: Zertifizierungsstelle (Testbetrieb abgeschlossen)

Ziel: Aufbau einer eigenen Zertfizierungsstelle oder CA (**C**ertificate **A**uthority) bei ask|net zur Herausgabe von Zertifikaten an Kunden oder Partner.

Nach einer kurzen Probezeit wurde die Zertifizierungsstelle bei ask|net wieder außer Betrieb gestellt. Der Betrieb einer eigenen Zertifizierungsstelle ist organisatorisch und sicherheitstechnisch zu aufwändig, da bei der Zertifikats-Ausstellung eine persönliche Nutzerauthentifizierung mit Personal- oder Universitätsmitarbeiter- oder Studentenausweis notwendig wird.
Für die Endnutzer selbst ist es ebenfalls mit geringeren Aufwänden verbunden, wenn sie Ihre Zertifikate von einer Instanz erhalten, die alle Partner kennen. Die Verwendung von Zertifikaten von öffentlichen CA's reduziert die Zahl der Zertifikate, ggf. auch die Zahl der Smartcards, die ein Nutzer mitführen müßte.

Paket 6: Zertifikatsbasierte Authentifizierung (Demobetrieb erfolgreich abgeschlossen)

Ziel: Realisierung, der sicheren entfernten Nutzerauthentifizierung mit digitalem Zertifikat basierend auf SSL.

Zertfikatsbasierte Authentifizierung mit Nutzerzertifikaten in der Betriebsystem- oder Browserdatenbank des Endnutzers oder externen Zertifikaten auf einer Smartcard sind in Entwicklungs- und Demonstrationsprojekten realisiert und getestet worden. Zum einen wurde auf die Möglichkeiten bereits breit eingesetzter Technologien, z.B. Clientauthentifizierung via HTTPS ausgelotet, zum anderen wurden eigene Komponenten entwickelt die am Ende auch eine digitalen Unterschrift im Browser ermöglichen (Ergänzungspaket B).

Paket 7: Zertifkatsbasierte Abrechnung (produktive Nutzung möglich)

Ziel: Implementierung des automatischen Versands von digital signierten Rechnungen.

Obwohl technisch einer der weniger aufwändigen Module, da Verschlüsselung und Signierung durch Plugins direkt von Mailprogrammen unterstützt werden, ist dieses Modul nach der Fertigstellung nicht zu Einsatz gekommen.
Hier fiel besonders ins Gewicht, daß das Signaturgesetz für digital versandte Rechnungen die höchste Sicherheitsstufe vorschreibt. Diese impliziert die manuelle Freigabe der Signierung unter Verwendung von Sicheren Darstellungskomponenten durch "*natürliche Personen*". Automatische Erzeugung und Versand von Rechnungen, wie bei Großversandhäusern im Schriftverkehr üblich, wird dadurch unmöglich.
Hinzu kommt, daß rein digitale Arbeitsabläufe zum Teil nicht bei unseren Partnern nicht vorgesehen oder gar nicht erlaubt sind. Verschiedene Universitätsverwaltungen waren und

sind noch dabei ihre rechtlichen Rahmenbedingungen zu ändern um wenigstens interne digitale Abläufe verbindlich abwickeln zu können.

Paket 8: Paymentmanager (produktiv)

Ziel: Analyse und Implementierung verschiedener Online-Bezahlmethoden, ibs. solcher die durch Anwendung digitaler Kundenzertifikate eine hohe Sicherheit versprechen, sowie eines zentralen Payment-Requestbrokers zur Verteilung von Bezahl-Anfragen auf die Module.
Der Paymentmanager besteht aus einer Gruppe von Modulen. Die ask|net war und ist in der Lage folgende Online-Payment-Methoden anzubieten:

- Kreditkarten (Visa, Amercan. Express, Mastercard,...)
- SET (Kreditkarte und Clientzertifikat)
- Paybox (Bezahlung per Handy)
- Elektronisch autorisierter Bankeinzug
- Paybyte (Micropaymentsystem)
- Cybercash (außer Betrieb)
- Elektronische Rechnung (siehe oben Paket 7)

Diese Module werden zu jeder Zeit weiterentwickelt um aktuelle Bezahlungsmethoden anbieten zu können.

Paket 9: Smartcardinterface (Demo verfügbar -> Paket B)

Ziel: Integration von Krypto- und Signaturfunktionalität von Smartcards über geeignete Plugin-Module für Webbrowser und Mailprogramme zur einfachen Anwendung durch Endnutzer von Webangeboten.

Diese Paket beschreibt ursprünglich die Integration von Komponenten verschiedenen Typs von Smartcard über Smartcardreader, Treibersoftware, Plugin-Module bis hin zu Applikationssoftware ibs. Web-Browser in einer clientseitigen Konfiguration, die die nahtlose Nutzung von Online-Diensten, Verschlüsslung von Nachrichten und die Online-Signatur von Dokumenten ermöglicht.
Dies ist durch verschiedene technische Hürden erschwert, zum Teil unmöglich gemacht (siehe Kapitel "Client Architektur", Seite 13). Daher wurde das Modul komplett ersetzt durch das neue Zusatzmodul **B**.

Paket 10: Einführung Smartcards (Testkontigent)

Ziel: Einführung von Smartcards bei Mitarbeitern, Partnern oder Kunden der ask|net.

Die ist aus heutiger Sicht aus Kosten- und aus organisatorischen Gründen nicht mehr sinnvoll. Vielmehr kann dies den Partnern selbst überlassen werden. Für Endkunden bietet sich die Möglichkeit sich ein digitales Zertifikat auf einer Chipkarte von einem, öffentlichen Trustcenter ausstellen zu lassen. Mitarbeiter oder Studenten von Forschungseinrichtungen bekommen in naher Zukunft eventuell ohnehin eine Chipkarte mit Signaturfähigkeit ausgestellt. Die zweite Variante würde auch die Authentifikation und Autorisierung von Neukunden für besondere Lizenzprogramme in ask|net-Portalen automatisieren, da Studenten oder Uni-Mitarbeiter bei der Neuregistrierung im Shop schon erkennbar sind.
Ask|net hat für Entwicklungs-, Test- und Demonstrationszwecke ein Kontingent von Karten für eigene Mitarbeiter und einige Partner angeschafft.

Paket 11: Runtimewrapper (produktiv)

Ziel: Entwicklung eines sicheren Auslieferungsmoduls für Softwarepakete, welches die illegale Weitergabe von Software soweit als möglich verhindert. Digitale Produkte werden zusätzlich verschlüsselt um über den Transport über SSL hinaus eine technische Bindung von Lieferungen an Kunden zu erreichen.

Der Runtimewrapper, auch als BOB (crypted **B**ox **o**f **B**its) bezeichnet ist in verschiedenen Versionen in verschiedenen Portalen im Einsatz. Es handelt sich um ein selbstentpackendes verschlüsseltes Archiv. Der Schlüssel ist wahlweise an Nutzer oder an Transaktionen gebunden. Die produktiven Varianten arbeiten mit symetrischen Schlüsseln, die Nutzern vorher per Mail zugesandt wurden.
Versionen, die gelieferte digitale Güter mit einem privaten Schlüssel eines zertifizierten Schlüsselpaares entpacken sind getestet, aber wegen der noch geringen Verbreitung von digitalen Zertifikaten nicht im produktiven Einsatz.
Der produktive Einsatz erfolgt nur für Produkte, für die eine doppelte Sicherung bei der digitalen Auslieferung, von Software vom Hersteller vorgeschrieben ist, da die Verschlüsselung vor der Auslieferung zur Laufzeit eine hohe Last auf dem Server erzeugt.

Paket 12: Transaktionsdatenbank (produktiv)

Ziel: Schaffung einer Verwaltungseinheit für die Zertifikats-Daten.

Die implementierten Erweiterungen der Transaktionsdatenbank der ask|net lassen sich in 3 Klassen teilen:
1. Zuordungseinheit von dig. Zertifikaten zu Nutzerkonten.
2. Sperrmöglichkeit von Konten und Zertifikaten.
3. Ablage von vom Kunden signierten Dokumenten.
4. Zuordnung von Sig. Dokumenten zur Geschäftstransaktionen

Diese Komponenten konnten ohne Probleme an die vorhandene Infrastruktur zur Verwaltung von Benutzern und deren Transaktionen angegliedert werden.

Paket 13: Integration Pilotnutzer (exemplarisch durchgeführt)

Ziel: Ausgewählte Smartcardeigentümer, ggf. ganze Personengruppen, führen alle Schritte einer kompletten Online-Software-Beschaffung durch.

Die Schritte in der Übersicht: Registrierung als Kunde, Anforderung, ggf. Online-Genehmigungsworkflow, Bestellung, Online-Bezahlung, digitale Lieferung und ggf. Abrechnung. Echte Transaktionen sowie Tests sind soweit möglich von Kunden oder von ask|net-Mitarbeitern durchgeführt worden.

Paket 14: Modellbetrieb (abgeschlossen)

Ziel: Prototypischer Einsatz und Durchführung echten Geschäftstransaktionen mit der entwickelten Technologie.

Der Modellbetrieb wurde als "überwachter" Produktivbetrieb für verschiedene Module durchgeführt. D.H. es wurden die Module nach Implementierung und Test direkt in produktive Portal aufgenommen und ihre Funktion in der realen Anwendung protokolliert. Einige Analysen des Nutzerverhaltens konnten Aufschluß über Akzeptanz oder Verbesserungsmöglichkeiten der Nutzerführung geben. Serverseitige Komponenten konnten so auch sofort unter Last geprüft werden.

Pakte 15: Geschäftsbetrieb (produktiv -> Paket C)

Ziel: Überführung der Komponenten in produktive Portale, Bereitstellung von Diensten für Forschungseinrichtungen.

Module, die inzwischen voll in Produktivbetrieb integriert sind:
- Requestbroker & ESD-System Anpassung
- Kryptomodul
- ESD-Loader/Downloadmanager
- Runtimewrapper (serverseitige und clientseitige Komponenten)
- Elektronische Rechnung ohne Signatur - ausdruckbar als pdf
- Kundenauthentifizierung per (Software-)Zertifikat

Paket 16: Projektmanagement

Das Projekt wurde integriert in der Entwicklungsabteilung der ask|net mit wechselnden Mitarbeitern durchgeführt. Dies ermöglichte die optimale Nutzung des unterschiedlichen vorhandenen Know-Hows und der Erfahrungen der Mitarbeiter zu verschiedenen betroffenen Themenschwerpunkten dig. Zertifikaten, elektronischer Produktlieferung, Smartcard-Technologie und rechtlicher Fragen. Neben der Projektleitung waren zu jeder Zeit 1-3 Mitarbeiter an dem Projekt beteiligt.

Hauptbeiträge:

Michael Kupperberg:	Smartcard-Technik auf dem Client (Studienarbeit)
Michael Weber:	Downloadmanager, Runtimewrapper
Klaus Peter Boden:	Bezahlsysteme
Matthias Honka:	Organisation, Öffentlichkeitsarbeit, digitaleZertifikate

Paket 17: Dokumentation

Die Projektentwicklung wurde auf der Website http://undine.asknet.de in den Projektberichten und verschiedenen Veröffentlichung dokumentiert.
Die Liste aller Veröffentlichungen ist im Anhang zu finden.

Zusätzlich entwickelte Pakete

Zusatzpaket A: Workflowmodul mit Signaturmöglichkeit

Ziel: Abbildung von Software-Beschaffungsprozessen an Forschungseinrichtungen als Web-Applikation in der Entscheider ihre Entscheidung per Mausklick zusätzlich digital signieren können.

Es ist ein konfigurierbares Beschaffungsystem entstanden, welches verschieden Beschaffungsprozesse an Forschungseinrichtungen elektronisch unterstützt (Bericht 2001/2). Es ist bereits seit Anfang dieses Jahres produktiv im Portal für die DLR. Die Siganturkomponente wird nicht genutzt, da hierzu alle Mitarbeiter mit Smartcards oder mindesten mit digitalen Zertifikaten versorgt werden müßten.

Zusatzpaket B: Online-Signatur im Webbrowser

Ziel: Realisierung einer plattformunahängigen Darstellungs- und Signatur-Komponente für Dokumente, die möglichst ohne hohe Vorleistungen des Endnutzers, wie etwa zusätzliche Softwareinstallation, im Browser eingesetzt werden kann.

Hierzu wurde u.a. eine Studienarbeit an der Universität Karlsruhe unter Betreuung von Prof. Thurm durchgeführt. Entstanden ist ein Java-Applet welches über eine pkcs11-Schnittstelle auf ggf. vorhandene Smartcardreader zugreifen kann. (siehe Anhang).

Zusatzpaket C: Softwareintegration

Ziel: Analyse und Optimierung der Datenübertragung zwischen Applet und Webapplikation.

Verschiedene Modelle wurden auf ihre Wirkung auf den Nutzerworkflow und technische Probleme analysiert:

- Applet interagiert lokal in der aufrufenden HTML-Seite und schreibt seine Ergebnisse in ein HTML-Formular zurück.
- Applet öffnet eigene SSL-Verbindung zum Server um Dokumente abzuholen und zu senden.
- Kombinationen

Fazit

Die Integration von PKI in eigene Webapplikationen gestaltet sich serverseitig durch Anwendung von Open-Source-Software, wie Apache, OpenSSL und ModSSL relativ einfach. Wesentliche Module der Applikations-Logik können unberührt bleiben.

Dagegen sind auf den Clients viele Voraussetzungen für eine systemübergreifende und leichte Integration von auf Hardwaretoken basierten Krypto- und Signiertechnologie noch nicht gegeben. Die Anbieter von Lösungen sind gezwungen, eigene Komponenten zu entwickeln und ggf. verschiedene proprietäre APIs der Smartcardhersteller unterstützen.

Standards von Microsoft werden zwar häufig von Herstellern unterstützt, allerdings auf der anderen Seite fehlt die Unterstützung von MS-Produkten insbes. des Internet-Explorer für die direkte Nutzung der Signierfunktionalität der MS-CryptoAPI. Zudem kann der IE nicht als "sichere Darstellungskomponente" im Sinne des Signaturgesetzes bezeichnet werden.

Die Verankerung der Crypto-API im Betriebsystem macht eine Integration von Krypto- und Signierfunktionalität in eigene Applikationen auf der einen Seite einfach, gleichzeitig untergräbt man damit die Unabhängigkeit der Sicherheit von digital signierten Transaktionen von der Sicherheit des Betriebsystems. Der Aufwand für Applikationsanbieter wird dadurch höher, je höher die Sicherheitsanforderungen sind. Ein wünschenswerte Delegation von Verschlüsselung und Signierung durch beliebige Applikationen an sichere Komponenten ist bisher nicht realisiert. [Abb. 13]

Insbesondere die Schnittstelle zwischen beliebigen Applikationen und Signaturkomponenten sollte aus unserer Sicht als Applikationsentwickler standardisiert werden. Speziell für Webapplikationen wäre es z.B. interessant dem Nutzer nach Eingabe einer URL einen neuen Dokumententyp zuzusenden als "***Document Signing Request***" analog zum schon existieren ***Certificate Signing Request*** (CSR) von Nutzern an Zertifikatsherausgeber (CAs). Entsprechend konfigurierte Browser können dann wie beim Empfang von pdf- oder anderen Dokumenten das entsprechende Plugin starten. In diesem Fall wäre das eben die sichere Darstellungskomponente, die von den Chipkarten oder Zertifikatsherausgebern mitgeliefert werden könnte.

Die einzige bisher "standardisierte" Übergabemöglichkeit von Daten an ein Verschlüsselungs oder Signierprogramm ist das Dateisystem. Nutzer speichern bisher unterschiedlichste Dokumente (Word, pdf, etc.) in einem Ordner ab und starten die Signierkomponente, die die proprietären Formate meist nicht darstellen kann, und signieren die Dateien dann blind. Die Manipulationsmöglichkeiten durch ggf. auf dem System vorhandene Trojaner oder Viren sind evident.

Ausblick

Die Anschaffung von Kartenleser, einer Signaturkarte mit Zertifikat kosten einen Endanwender z.Z. min. 50 €. In der nahen Zukunft werden daher aus Kostengründen im Endnutzerbereich Softwarezertifikate die vorrangige Rolle spielen. Die geringere Sicherheit gegenüber einer "qualifizierten Signatur" mit Karte spielt hierbei eine untergeordnete Rolle, da die Rechtsgültigkeit vieler durch "Mausklick" ausgelösten Online-Transaktionen heute schon gegeben ist.

Trotz der noch offenen Sicherheitsfragen bilden digitale Zertifikate jetzt schon eine interessante Möglichkeit Probleme der einheitlichen Authentifizierung auf mehreren verteilten Systemen, die auch z.T. offline arbeiten können, zu lösen. Certficate Authorities oder Trustcenter dienen als zentrales Authentifizierungsmodul.

In der derzeitigen Praxis zeigt sich die Bereitschaft zum Einsatz von Zertifikaten hauptsächlich bei:

- Der Absicherung von Server-Server-Kommunikation
- Nutzerauthentifizierung zur Erreichung höherer Vertrauenswürdigkeit als bei klassischen passwortbasierten Verfahren durch Software-Zertifikate.
- Verschlüsselung von persönlicher Emailkommunikation (bspw. PGP)

Der Einsatz von Zertifikatstechnologie wird bei der ask|net sicherlich weiter zunehmen. Die aufgezeigten und bisher noch prototypisch implementierten Einsatzmöglichkeiten werden in naher Zukunft, wenn die Verbreitung von Softwarezertifikaten und/oder Signaturkarten zunimmt, ebenfalls produktiv eingesetzt werden. Verschiedene derzeitige Ansätze und Projekte sind z.Z. in der Diskussion und Planung.

Details zur Implementierung eines Smartcardinterfaces für den Einsatz in Webapplikationen werden separat veröffentlicht.

Anhang

A: Veröffentlichungen

Nr.	Veröffentlichung	Autor/Referent	Typ
1	Vortrag auf der DFN Tagung In Kassel Feb. 2001	Dr. Waudig (ask|net)	Vortrag
2	Intermediäres Software Management der ask|net	Ask|net	Präsentation
3	Artikel in den DFN-Nachrichten	M. Honka	Artikel
4	Vortrag auf der DFN-Arbeitstagung in Düsseldorf Mai 2002	M. Honka	Vortrag
5	Beitrag zum Tagungsband In GI-edition, Lecture Notes in Informatics, ISBN 3-88579-346-6	M. Honka	Artikel
6	Studienarbeit Entwicklung einer portablen Smartcardinterface für Webapplikationen	M. Kupperberg	Studienarbeit +Vortrag
7	4 Projektberichte halbjährlich 2000/2 - 2002/1	M. Honka	Berichte

Alle Vorträge und Artikel sind auf der Webseite von UNDINE (http://undine.asknet.de/) als PowerPoint (ppt) bzw. Adobe Portable Document-Format (pdf)-Dateien abgelegt.

B: Referenzen

Undine
Projektseite: http://www.undine.de
Ask|net: http://www.asknet.de

Digital Payment
Untersuchung IWW - Uni Karlsruhe: http://www.iww.uni-karlsruhe.de/IZV5/
SET: http://www.setco.org
Paybox: http://www.paybox.de
Stackbox: http://stackbox.de

PKI allgemein
Linksammlung: http://www.pki-page.org
BSI: http://www.bsi.de/literat/doc/index.htm

Karten-APIs und Standards
Opencard: http://www.opencard.org
PKCS11: http://developer.netscape.com
CryptoAPI: http://msdn.microsoft.com

Smartcardhersteller:
Utimaco: http://www.utimaco.com
Towitoko: http://www.towitoko.de
IBM: http://www.ibm.com
Basiccard: http://www.basiccard.com

Sonstige Technik
Apache Webserver: http://www.apache.org

Tomcat Servlet Engine: http://java.apache.org
OpenSLL: http://www.openssl.org
ModSSL: http://www.modssl.org
JSSE: http://java.sun.com/products/jsse/

UNDINE wird gefördert durch den Verein zur Förderung eines Deutschen Forschungsnetzes (DFN).