Generische Sicherheitsdienste für mobile Anwendungen

Jürgen Bohn, Günter Karjoth

Mobile Agenten ermöglichen eine flexible Ausführung von verteilten Anwendungen und die Realisierung mobiler, dynamischer Dienste. Dies macht sie attraktiv für den Einsatz in Bereichen wie z.B. des elektronischen Handels, des Netzwerkmanagements oder der Groupware. Ein erfolgreicher Einsatz hängt aber auch davon ab, in wieweit ihr Schutz gewährleistet ist. Dieser Beitrag beschreibt die Realisierung einer mobile Agentenanwendung zur sicheren Produktrecherche, die mit Hilfe von generischen Sicherheitsdiensten entwickelt wurde. Generische Basisdienste wie auch komplexere, anwendungsorientierte Sicherheitsdienste verringerten den Entwicklungsaufwand und die Konfigurierbarkeit des Software-Rahmens schaffte Spielraum für die Befriedigung individueller Sicherheitsbedürfnisse der mobilen Agentenanwendung.

Inhaltsübersicht

1 Einleitung

Die Attraktivität der mobilen Agententechnologie [Mattern 1998] beruht insbesondere auf möglichen Anwendungsszenarien aus den Bereichen elektronischer Handel, Netzwerkmanagement und Groupware. Eine Stärke des Paradigmas mobiler Agenten liegt darin, dass es den Entwurf, die Implementierung und die Wartung verteilter Anwendungen wesentlich einfacher und intuitiver gestaltet. In verteilten Systemen lassen sich die autonomen, miteinander kooperierenden Parteien in natürlicher und verständlicher Weise als mobile Agenten modellieren. Während für viele Einzelfälle herkömmliche Lösungsansätze ohne den Gebrauch mobiler Agenten existieren, so stellt das Paradigma mobiler Agenten ein Rahmenwerk zu Verfügung, welches all diese Fälle gleichzeitig abdeckt und behandelt [Chess *et al.* 1997].

Trotz der anfänglichen Euphorie hat sich die Technologie nur zögerlich verbreitet. Dies liegt nicht zuletzt daran, dass ein Grundproblem der mobilen Agenten, die Frage nach der Sicherheit, noch nicht zufriedenstellend gelöst ist. Während der Schutz des Wirtssystems vor böswilligen Agenten bereits relativ gut verstanden ist, besteht noch Forschungsbedarf beim *Schutz des Agenten* vor zufälliger oder böswilliger Manipulation [Vigna 1998]. Für den Nutzer von mobilen Agentenanwendungen ist dieser Sicherheitsaspekt von besonderer Bedeutung. Mag der Ausfall einer Plattform im Gesamtsystem geduldet werden, die fehlende Zuverlässigkeit und der mangelnde Integritätsschutz der eigenen mobilen Agenten wird jedoch kaum toleriert. Die Sicherheitseigenschaften mobiler Agenten sind deshalb für die Akzeptanz und Verbreitung von Agentenanwendungen ein kritischer Faktor.

Interoperable und sichere mobile Agentenanwendungen erfordern offene Agentensysteme, welche einen Rahmen von Sicherheitsdiensten besitzen, deren Schnittstellen bekannt sind und allen Teilnehmern zur Verfügung stehen. Neben den kryptographischen Dienstprimitiven, wie z.B. Methoden zur Verschlüsselung, digitale Signaturverfahren oder kryptographische Hashfunktionen, und den darauf aufbauenden bereits vorgestellten elementaren Sicherheitsdiensten, treten mit zunehmender Nähe zur Anwendung weitergehende Sicherheitsbedürfnisse und Forderungen auf.

Dieser Beitrag beschreibt die Realisierung einer mobile Agentenanwendung zur sicheren preisvergleichenden Produktrecherche (*comparison shopping*): Der mobile Agent eines Kunden besucht eine Reihe von elektronischen Marktplätzen (Agentenplattformen), auf denen er von den jeweils vorhandenen Händleragenten ein Produktangebot einholt. Nachdem der mobile Agent eine gewisse Anzahl solcher Angebote gesammelt hat, kehrt er zum Ursprungsort zurück und übergibt seinem Auftraggeber das beste gefundene Angebot. Die Anwendung benützt generische Sicherheitsdienste aus einem Sicherheitsrahmen [Bohn & Karjoth 2001]. Generische Basisdienste, wie etwa Containerklassen, wie auch komplexere, anwendungsorientierte Sicherheitsdienste verringern dabei den Entwicklungsaufwand. Die Konfigurierbarkeit des Software-Rahmens schafft Spielraum für die Befriedigung individueller Sicherheitsbedürfnisse von mobilen Agentenanwendungen.

2 Ansätze zum Schutz mobiler Agenten

Heutige Wirtssysteme schützten Agenten vor Manipulationsversuchen durch andere Agenten, in dem diese in isolierten Adressräumen ausgeführt werden. Dabei erlauben proxy-basierte Zugriffsmechanismen den Austausch von Agentenobjekten. Die Ausführungsplattform selber hat aber Zugriff auf alle unverschlüsselten Daten des Agenten und kann daher den internen Kontrollfluss ausspionieren und zum eigenen Vorteil manipulieren. Es stellt sich damit die Frage nach der Verlässlichkeit der ausführenden Plattform, und wie man sich gegen betrügerische Manipulationen schützen oder diese zumindest zuverlässig erkennen kann.

Von betrügerische Rechnerplattformen können eine Vielzahl möglicher Angriffe ausgehen, wie zum Beispiel das Auspionieren oder Manipulieren von Daten, Code oder Kontrollfluss. Existierende Lösungen

zum Schutz des Agenten vor unbefugten Dritten werden danach unterschieden, ob sie aktiven oder passiven Schutz gewähren. Während die Verwendung von Kryptographie es erlaubt, Daten innerhalb des Agenten zu verbergen und deren Integrität zu überprüfen, scheint heute nur die Verwendung spezieller, vertrauenswürdiger Hardware in der Lage zu sein, die Manipulation von Agenten zu verhindern. Es gibt zwar bereits mehrere Lösungsansätze zum Schutz mobiler Agenten, die nur auf Software beruhen, ihre praktische Einsatzfähigkeit ist jedoch begrenzt [Karjoth & Posegga 2000].

Sichere, vertrauenswürdige Geräte erscheinen bezüglich ihrer Funktionsweise nach außen hin als eine Blackbox. Kritische Operationen des Agenten werden so ausgelegt, dass sie nur innerhalb dieser sicheren Geräte ausführbar sind und sich somit dem Einfluss des zugehörigen Rechners entziehen. Die Bandbreite derartiger Geräte reicht von nur kreditkartengroßen Chipkarten über komplette geschützte Rechner bis hin zur teuren Spezialhardware.

Auf Grund dieser Sachlage muss aber ein Rahmen von Sicherheitsdiensten gewährleisten, dass der Schutz der Ausführung mobiler Agenten gegen Manipulation und Ausspionieren sowohl durch existierende Lösungen, wie die Verwendung von sicheren, vertrauenswürdigen Geräten, als auch durch noch zu entwickelnde Verfahren realisiert werden kann. Ein Rahmenwerk von generischen Sicherheitsdiensten abstrahiert von der konkreten Realisierung und kann daher verschiedene Implementierungen der gleichen Dienstschnittstelle, z.B. mit oder ohne Verwendung sicherer Hardware, bereitgestellen.

3 Sicherheitsdienste

Sicherheitsdienste gewähren die Integrität und/oder Vertraulichkeit des Zustandes eines Agenten und dessen Ausführung auf unsicheren Plattformen.

3.1 Klassifikation

Besitzt das auszuführende Programm eines Agenten zur Laufzeit einen statischen Charakter, kann die empfangende Plattform anhand von digitalen Signaturen bzw. Zertifikaten (einer bekannten vertrauenswürdigen Instanz) dessen Integrität überprüft werden. Programmcode, der während der Ausführung dynamisch nachgeladen wird, wie es zum Beispiel Java ermöglicht, muss zudem noch auf Kompatibilität überprüft werden. Diese Überprüfungen dienen vorrangig dem Schutz der Plattform vor bösartigen Agenten.

Auch wenn die Integrität des Programms gewährleistet ist, kann der Ausführungszustand des Agenten, definiert durch seinen Variablenzustand und dem Ausführungskontext (Befehlszähler, Aufrufverschachtelung, etc.), von der auszuführenden Plattform manipuliert werden.

Die anwendungsspezifischen Daten eines mobilen Agenten können im Hinblick auf deren Persistenz ebenfalls als statisch oder dynamisch klassifiziert werden. Weiter wird der Datenanteil nach semantischen Gesichtspunkten in Nutzdaten, Metadaten und Prüfdaten untergliedert. Nutzdaten sind jene Daten, die zur Erledigung der primären Aufgaben des Agenten gesammelt werden, z.B. die Angebote in der Produktrecherche. Metadaten tragen nur indirekt zur Lösung der durch den Agenten zu bearbeitenden Aufgabenstellung bei, etwa Auftragspara-

meter oder die Liste der zu besuchenden Händler. Prüfdaten dienen ausschließlich dem Schutz der Integrität und Vertraulichkeit der Nutzdaten, Metadaten oder des Codes selbst.

Die auf den verschiedenen Datenarten operierenden Parteien lassen sich in drei Gruppen einteilen: Der mobile Agent selbst, eine von allen teilnehmenden Parteien als vertrauenswürdig anerkannte Instanz (Trusted Third Party), sowie alle anderen am System beteiligten Parteien. Grundoperationen auf diesen Daten sind Lesen, Hinzufügen, Entfernen, und Ersetzen. Diese Operationen lassen sich weiter gemäss ihrer Nutzung verfeinern, z.B. die Verifizierung einer Hashkette oder die Aktualisierung einer Prüfsumme.

Dieser Gliederung erlaubt die Ableitung von grundlegende Datentypen und Sicherheitsdienste für mobile Agentenanwendungen. Dabei zeigt sich, dass viele elementare sichere Dienste sich durch verschiedene Ausprägungen von sicheren Containerklassen bereitstellen lassen. Ein Container ist – in der allgemeinsten Ausprägung – ein Behälter, dem Objekte beliebigen Typs hinzugefügt und entnommen werden können. Die Containerklasse stellt entsprechende Methoden bereit. Spezialisierungen sind beispielsweise möglich durch Einschränkung der zugelassenen Objektklassen (Typisierung) oder der Zugriffsmethoden (z.B. nur lesenden Zugriff gestatten).

Derartige Containerklassen sind für den Entwickler einfach zu benutzen und leicht in Anwendungen zu integrieren, weil sie sich in Bezug auf die Funktionalität nur unwesentlich von herkömmlichen Containerklassen unterscheiden. Ausserdem lässt sich so leicht die Trennung von Schnittstelle und Implementierung realisieren, so dass die tatsächlich verwendeten Sicherheitsmechanismen transparent zur Anwendungsentwicklung und zur Laufzeit gewählt werden können. Mit der gewählten Modellierung in Java wird diese Trennung analog zum SPI-Konzept aus *Java 2* gelöst, durch Definition von sogenannten Diensterbringer-Schnittstellen (*Service Provider Interfaces*, SPI).

3.2 Ein Softwarerahmen

Als Sicherheitsdienste wurden verschiedene Containerklassen zu einem Rahmen zusammengefasst und prototypisch als Java-Bibliothek implementiert. Die einzelnen Dienste können durch unterschiedliche Mechanismen realisiert werden, so dass mobile Agentenanwendungen flexibel gestaltbar sind.

Ein `ReadOnlyContainer` erlaubt beispielsweise nur das Lesen und ein `RemoveOnlyContainer` nur das Entnehmen von Daten bzw. Objekten. Der `LockableContainer` entspricht einem auf- und abschließbaren Behälter. Ein `AppendOnlyContainer` erlaubt nur das Anhängen von Daten an den bereits vorhandenen Datenbestand sowie das Lesen der vorhandenen Daten. Der davon abgeleitete `SecretAppendOnlyContainer` verschlüsselt zusätzlich die abgelegten Daten mit dem öffentlichen Schlüssel des Empfängers, um die Vertraulichkeit der Daten zu gewährleisten. Eine mögliche Realisierung erfolgt z.B. mit Hilfe des `PushOnlyIntegrityStack`-Dienstes, bei dem Objekte auf den Stack abgelegt, nicht aber entnommen werden dürfen. Die unerlaubte Entnahme oder Veränderung von Objekten wird je nach Implementierung entweder durch einen Software-Mechanismus erkannt oder durch entsprechende Prüfmaßnahmen innerhalb

einer sicheren Hardware unterbunden.

Der `PushOnlyIntegrityStack` ist eine Spezialisierung des integritätsgeschützten Stacks `IntegrityStack`, dessen Implementierung in zwei Abstraktionsschritten erfolgt. In der ersten Stufe fußen die abstrakten Stackklassen auf einer generischen Diensterbringer-Schnittstelle nach dem SPI-Konzept, dem Interface `StackCryptoServices`, das allgemeine Methoden zur Bereitstellung der Stackfunktionalität definiert. Die zweite Stufe stellen Implementierungen dieser SPI-Schnittstelle dar, wie z.B. die Klasse `IntegrityStack`. Hier werden die sicherheitskritischen Funktionen verborgen und realisiert. Dabei findet die Abbildung der allgemeinen Stackprimitiven auf konkrete kryptographische Softwarebibliotheken oder sichere Hardware statt. Die integritätsgeschützten Stackklassen sind auch Bestandteil des Softwarerahmens.

Eine prototypische Implementierung und Anwendung der beschriebenen Sicherheitsdienste erfolgte auf dem Aglets Agentensystem [Lange & Oshima 1998]. Der realisierte Rahmen mit den beschriebenen (und ggf. noch weitere zu identifizierende, applikationsspezifischere) Dienste lässt dem Anwender die freie Wahl der Implementierung.

4 Anwendungsszenario

Die preisvergleichenden Produktrecherche ist eine der am häufigsten zitierten Anwendungen von mobilen Agenten im elektronischem Handel. Ein *preisvergleichender Agent* macht im Auftrag seines Besitzers (dem Kunden) selbständig aus einer an sich unüberschaubaren Anzahl von Händlern und Angeboten die in Frage kommenden (hier: günstigsten) Angebote ausfindig.

Die Grundzüge einer agentenbasierten preisvergleichenden Produktrecherche umfassen dabei im Wesentlichen zwei Arten von Agenten: Den vom Kunden beauftragten Suchagenten (oder Einkaufsagenten, falls der Agent mit der entsprechenden Befugnis ausgestattet ist) sowie eine Anzahl von Verkaufsagenten, die auf Anfrage Auskünfte über aktuelle Händlerangebote geben. Der Ablauf der Produktrecherche ergibt sich wie folgt:

1. Der Kunde instruiert (auf der Heimatplattform) den mobilen Agenten mit der Beschreibung der zum Kauf gewünschten Ware und gibt mit Hilfe eines Adressbuches die Liste der zu besuchenden elektronischen Marktplätze an (Wegbeschreibung).

2. Anschließend migriert der Agent vom Rechner des Kunden zum ersten elektronischen Marktplatz. Dort angekommen stellt der Agent eine Produktanfrage an alle vorhandenen Händleragenten. Nach Erhalt der Preisangebote werden diese gespeichert, und der Agent setzt seine Reise fort zum nächsten Marktplatz, wo er diesselbe Anfrage wieder stellt.

3. Nachdem der Agent alle Marktplätze besucht und eine gewisse Menge von Angeboten eingeholt hat, kehrt er zurück zur Heimatplattform. Hier werden dem Auftraggeber alle gesammelten Angebote präsentiert und das preisgünstigste Angebot zum Kauf vorgeschlagen.

Die Beschreibung der Route des Agenten ist statisch, da sie in diesem Fall zu Beginn festgelegt wird, und kann deshalb krypto-

graphisch mit Hilfe der Dienstklasse `ReadOnlyContainer` gesichert werden. Die Routenplanung kann aber alternativ auch dynamisch erfolgen, etwa durch den Einsatz von Verzeichnisagenten, die dem mobilen Agenten unterwegs die Verfügbarkeit weiterer Marktplätze und Händleragenten mitteilen. Darüber hinaus können Marktplätze einen oder mehrere Händleragenten beherbergen. Dies hat den Vorteil, dass bei der Verwendung von sicherer Hardware diese nur einmal für jeden Marktplatz zur Verfügung gestellt werden muss, und nicht für jeden Händler einzeln.

Um die Integrität und Vertraulichkeit der gesammelten Daten zu gewährleisten, verwendet der mobile Agent den Dienst `SecretAppendOnlyContainer`. Dabei kann man unter zwei Varianten auswählen, um mit oder ohne Einbezug von Chipkarten die gesammelten Daten sicher zu verknüpfen. In der chipkartenlosen Variante werden die kryptographischen Operationen auf dem Wirtssystem ausgeführt werden. Die andere Variante setzt auf den händlerseitigen Einsatz von Chipkarten, die im folgenden näher beschrieben werden. Durch Verwendung der entsprechenden SPI-Implementierungen können beide Protokolle in einfacher Weise in die Anwendung integriert werden.

4.1 Protokoll

Die chipkartenbasierte Implementierung des `SecretAppendOnlyContainer` beruht auf einer Spezialisierung eines Protokolls von Devanbu und Stubblebine, in dem sichere, aber resourcenlimitierte Hardware-Stacks und -Queues auf unsicheren Rechnern speichern können, während nur eine konstante Speichermenge innerhalb der sicheren Hardware benötigt wird [Devanbu & Stubblebine 1998].

Der sparsame Umgang mit Speicher auf der Chipkarte ist für mobile Agenten sehr wichtig, da nur einzelne Elemente des Stacks und die Kontrollinformationen in der sicheren Hardware für Prüfzwecke bearbeitet werden müssen, während die unter Umständen sehr umfangreichen restlichen Daten im externen Speicher gehalten werden können. Die sicherheitskritische Datenstruktur enthält dabei nur den geheimen Anker und den zuletzt berechneten Wert der Hashkette, mit deren Hilfe sich die Integrität der externen Daten überprüfen lässt, wie in Abbildung 1 gezeigt.

Die Sicherheit der Hashkette beruht auf dem strikten Schutz der kritischen Protokolldaten, repräsentiert durch das Tupel $\langle h_0, h_{n+1} \rangle$, die immer nur innerhalb der Blackbox entschlüsselt, ausgewertet und aktualisiert werden. Der vorgehaltene Abschluss h_{n+1} verhindert die unentdeckte Manipulation (Entfernen, Hinzufügen, Verändern) von Datenobjekten: Die Neuberechnung der Hashkette ergibt im Falle eines verändernden Eingriffes aufgrund der Kollisionsfreiheit der kryptographischen Hashfunktion einen von h_{n+1} verschiedenen Endwert $\widehat{h_{n+1}}$, so dass die Veränderung erkannt wird. Als eindeutiger Identifikator der Hashkette dient der Hashwert $h_1 = \mathrm{Hash}(h_0)$, der öffentlich bekanntgemacht wird. Die nur dem Erzeuger der Dateninstanz bekannte geheime Verankerung h_0 verhindert, dass die Hashkette unbemerkt durch eine andere, in sich konsistente Hashkette ersetzt werden kann, die über andere Daten berechnet wurde.

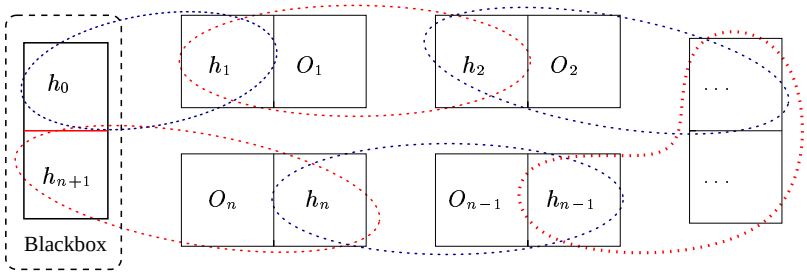

Abbildung 1: Aufbau der integritätsschützenden Hashkette: h_0 ist der geheime Startwert, h_{n+1} der (offengelegte) Abschluss, und O_i mit $1 \leq i \leq n$ sind die zu schützenden Objekte. Die rekursive Berechnung der Hashwerte h_i erfolgt über den durch die gestrichelten Ellipsen umfassten Daten. Das Datentupel $\langle h_0, h_{n+1} \rangle$ erlaubt die Überprüfung der Hashkette und wird nur innerhalb der sicheren Hardware ausgewertet und aktualisiert.

4.2 Implementierung

Unsere Implementierung der Containerklasse benutzt eine JavaCard [Baentsch *et al.* 1999], deren Leistungsfähigkeit und Speicherplatz groß genug ist, die Angebote der einzelnen Händler sicher in die Angebotsliste einzufügen. Immer wenn der mobile Agent auf einer unsicheren Plattform ein neues Angebot im `SecretAppendOnlyContainer` ablegt, wird innerhalb der JavaCard die integritätsschützende Kontrolldateneinheit entsprechend aktualisiert. Zum Zeitpunkt der Initialisierung des Container werden die Kontrolldaten mit dem öffentlichen Blackbox-Schlüssel versiegelt und können fortan nur noch innerhalb der JavaCards entschlüsselt und modifiziert werden; sie entziehen sich somit dem Wirkungskreis der Händlerplattformen. Manipulationen an der Datenstruktur unter Umgehung der Blackbox können stets nachgewiesen werden (mit Ausnahme von Replay-Attacken, die generell nicht verhindert werden können, wenn der Agent gänzlich autonom und unabhängig von anderen Plattformen operiert).

Die hier beschriebene Realisierung des `SecretAppendOnlyContainers` beruht auf einem Protokoll, in dem sich alle JavaCards ein Public-Key-Schlüsselpaar teilen. Es ist aber auch möglich, dieses Modul durch eine Implementierung eines Protokolls aus [Karjoth 2000] zu ersetzen, in dem jede JavaCard ein eigenes Public-Key-Schlüsselpaar zugeordnet ist.

4.3 Praxisbetrachtungen

In der Praxis stellt sich in Hinblick auf die Produktbeschreibung nicht nur die Frage nach einer eindeutigen Repräsentation der Angebotsinformation, z.B. unter Verwendung von XML [Siegemund *et al.* 2001], sondern auch nach einer gemeinsamen *Ontologie*: Agent und elektronische Läden sollen die gleiche Sprache sprechen bzw. die des jeweiligen Verhandlungspartners unmissverständlich interpretieren können. Jeder Agent wird in diesem Zuge mit einer komprimierten Produktbeschreibung ausgestattet, die von allen Teilnehmern des Produktrecherchesystems

eindeutig interpretiert werden kann. Diese Produktbeschreibungen werden z.B. in einer *Ontologie-Datenbank* verwaltet und durch einen *Ontologie-Server* zugänglich gemacht. Der Einsatz *umgangssprachlicher Produktbeschreibungen* im Modell ist wenig geeignet – einerseits laufen diese dem Bestreben zuwider, die durch den Agenten zu transportierende Datenmenge klein zu halten, andererseits bedingen sie einen erhöhten Verarbeitungsaufwand auf Seiten der elektronischen Läden, wobei zudem eine inkorrekte Interpretation nicht ausgeschlossen werden kann.

Im Idealfall sind der Geltungsbereich der verwendeten Ontologie global und die Produktbeschreibungen einheitlich für alle elektronischen Märkte. In der Realität ist aber zu erwarten, dass dies nur eingeschränkt für Untergruppen von Märkten der Fall sein wird: administrative und organisatorische Beschränkungen für regionale Systeme begünstigen lokale Marktverbunde und Händergemeinschaften (Händlerringe).

In kommerziellen mobile Agentenanwendungen lassen sich typischerweise zwei Fälle bezüglich des Erzeugungsortes und der Kontrolle der Agenten unterscheiden. Entweder kann der Kunde den Agenten auf seiner eigenen Plattform erzeugen und instruieren, wie in Abbildung 2 dargestellt, oder er stellt seine Anfrage über eine definierte Benutzerschnittstelle, die im folgenden als *Zugangsportal* bezeichnet wird. Ein solches Portal könnte, wie in Abbildung 3 gezeigt, eine HTML-Web-Schnittstelle im World Wide Web anbieten, mit deren Hilfe der Kunde über eine sichere HTTP-Verbindung kommuniziert. In diesem Falle hat der Kunde keine unmittelbare Kontrolle über seinen Agenten, sondern er beschreibt über die Zugangsschnittstelle das gewünschte Produkt, worauf vom System ein entsprechender mobiler Agent erzeugt und abgeschickt wird.

Der Einsatz eines Zugangsportals und die indirekte Beauftragung durch den Kunden bringt wesentliche Vorteile mit sich. Erstens kann dadurch ein lokaler Händlerverband (oder Händlerring) seine Anfragen auch auf eine lokale Ontologie-Datenbank stützen, d.h. es entfällt die Problematik einer globalen Ontologie, die offenen Systeme inhärent ist bzw. die Verwendung der passenden Ontologiedatenbank erfolgt implizit und ohne notwendiges Eingreifen des Benutzers. Außerdem bleibt dem Endbenutzer die Installation und der Betrieb einer eigenen Agentenplattform erspart, so dass der eigentliche Dienst, die Produktrecherche, für den Klienten transparent im Hintergrund durch die mobile Agententechnologie abgewickelt wird. Werden die Resultate nach getaner Arbeit des Agenten per verschlüsselter Email an den Kunden zurückgeschickt, so bleibt auch der asynchrone Charakter der Kunden-Agenten-Beziehung erhalten. Die einmal delegierte Aufgabe wird vom mobilen Agenten selbständig und zeitlich entkoppelt erfüllt.

Das vorgeschlagene Verfahren zum Schutz der preisvergleichenden Recherche verlangt das Vorhandensein von sicherer Hardware auf Seiten der Händler. Als low-cost Variante wurde hier dazu der Einsatz von *Smart-Cards* (JavaCards) vorgeschlagen. Zur Ausgabe, Verwaltung und Kontrolle der JavaCards wird daher eine *Blackbox–Behörde*, also eine vertrauenswürdige Dritte Instanz benötigt. Diese könnte sich im konkreten Szenario z.B. aus dem Zusammenschluss der beteiligten Händler zu einem Händlerring ergeben. Zu den denkbaren Aufgaben einer solchen Blackbox–Behörde zählen u.a. die Verwaltung und Verteilung der Blackbox-Schlüssel und deren Erneuerung

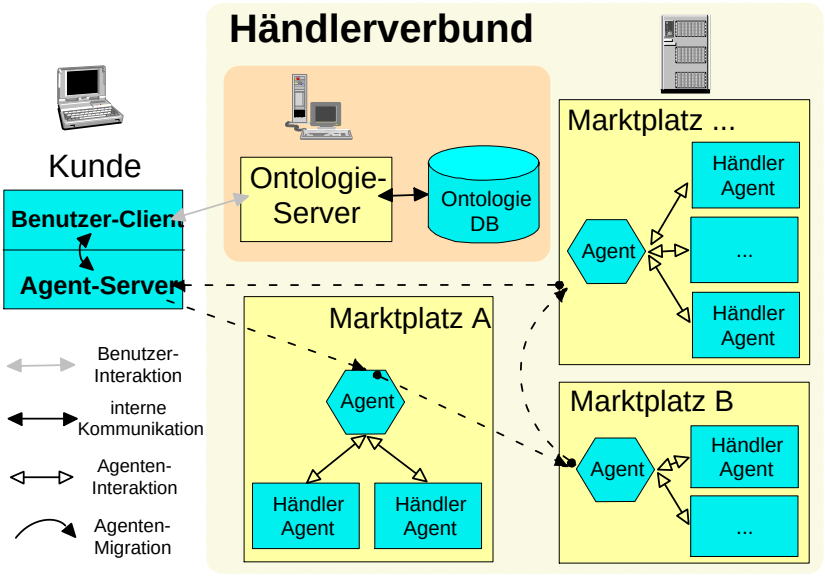

Abbildung 2: Produktrecherche mit lokaler Benutzerschnittstelle.

in regelmäßigen Zeitintervallen. Geht dies Hand in Hand mit einer Überprüfung der SmartCards auf Manipulationen und mutwilligen Veränderungen, so können etwaige Betrugsversuche der am Händlerring teilnehmenden Händler aufgedeckt und diese ggf. rechtlich belangt werden.

5 Fazit

Generische Sicherheitsdiensten erlauben die Erstellung sicherer, mobiler Anwendungen in offenen Agentensystemen. Basisdienste wie auch komplexere, anwendungsorientierte Sicherheitsdienste verringern den Entwicklungsaufwand. Die Konfigurierbarkeit des Software-Rahmens schafft Spielraum für die Befriedigung individueller Sicherheitsbedürfnisse von mobilen Anwendungen. So erlaubt der Rahmen die transparente Ersetzung einzelner Module, wenn zum Beispiel höhere Sicherheitsanforderungen gestellt werden oder sich herausstellt, dass ein konkreter Mechanismus nicht die Eigenschaften des Dienstes, welchen er realisieren soll, erfüllt.

Die Auseinandersetzung mit dem Paradigma mobiler Agenten zeigt, dass diese Technologie ein großes Potential für zukünftige Entwicklungen und Einsatzszenarien bietet, insbesondere für mobile Anwendungen im elektronischen Handel. Die Frage der Sicherheit und des Schutzes von mobilen Agenten wird dafür auch in Zukunft ein wichtiger Aspekt von zentraler Bedeutung bleiben.

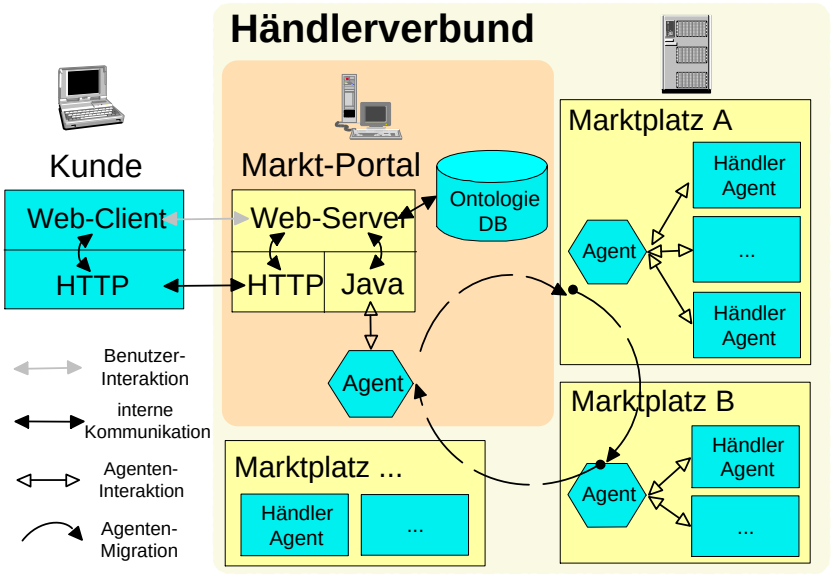

Abbildung 3: Produktrecherche mit Web-Schnittstelle als Zugangsportal.

Literatur

Baentsch, M.; Buhler, P.; Eirich, T.; Höring, F.; Oestreicher, M.: JavaCard — From Hype to Reality. In: *IEEE Concurrency*, **7**(4), 1999, S. 36–43.

Bohn, J.; Karjoth, G.: Sicherheitsdienste für mobile Agentenanwendungen. In: *Kommunikation in Verteilten Systemen*, 2001, Springer-Verlag, S. 305–314.

Chess, D.; Harrison, C.G.; Kershenbaum, A.: Mobile Agents: Are they a good idea? In: Mobile Object Systems – Towards the Programmable Internet, Lecture Notes in Computer Science 1222, 1997, Springer-Verlag, S. 25–47.

Devanbu, P.T.; Stubblebine, S.G.: Stack and Queue Integrity on Hostile Platforms. In: Proceedings IEEE Symposium on Research in Security and Privacy. 1998, IEEE Computer Society Press, S. 198–207.

Karjoth, G.: Secure Mobile Agent-Based Merchant Brokering in Distributed Marketplaces. Agent Systems, Mobile Agents, and Applications. Lecture Notes in Computer Science 1882, 2000, Springer-Verlag, S. 44–56.

Karjoth, G.; Posegga, J.: Mobile Agents and Telcos' Nightmares. *Annales des Télécommunications*, **55**(7/8) 2000, S. 29–41.

Lange, D.B.; Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley Longman, 1998.

Mattern, F.: Mobile Agenten. *it+ti – Infor-*

mationstechnik und Technische Informatik, 1998, S. 12–17.

Siegemund, F.; Cap, C.H.; Heuer, A.: Einsatz von mobilen Agenten und XML zur Angebotsrecherche im Business-to-Consumer-Commerce. *Wirtschaftsinformatik*, 2001, Heft 2, S. 157–166.

Vigna, G. (Hrsg.): Mobile Agents and Security. Lecture Notes in Computer Science 1419, 1998, Springer-Verlag.

Dipl.-Inform. Jürgen Bohn
ETH Zürich
Haldeneggsteig 4, IFW
CH-8092 Zürich
bohn@inf.ethz.ch

Dr. Günter Karjoth
IBM Forschungslabor Zürich,
Säumerstrasse 4
CH-8803 Rüschlikon
gka@zurich.ibm.com