Master-Abschlußarbeit

Lizenzierung bei modularer und erweiterbarer Software

Herrn Prof. Dr. Friedrich Steimann
am
Lehrgebiet Programmiersysteme
des
Fachbereichs Informatik
der
FernUniversität Hagen
vorgelegt
zur Erlangung des akademischen Grades eines
Master Of Computer Science
von

Daniel Dreher

Aschenreutestraße 8

78591 Durchhausen

pentad@pentad.de

geboren am

16. Dezember 1977

in Spaichingen

Matrikel-Nummer 65 20 456

Durchhausen, den 27. März 2006

Hiermit versichere ich an Eides Statt, daß ich die vorliegende Arbeit selbst verfaßt habe und keine anderen Quellen und Hilfsmittel als die angegebenen verwendet habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, wurden in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Das gleiche gilt für beigefügte Skizzen und Darstellungen. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Durchhausen, den 27. März 2006

_____

Zusammenfassung

Mit der zunehmenden Leistungsfähigkeit der Rechnersysteme wird die zugehörige Software immer komplexer und umfangreicher. Zur Finanzierung der Entwicklungsaufwände sollte der Markt möglichst groß sein. Aus diesem Grund bietet sich die Entwicklung modularer Software an, wobei die Module einzeln an den Kunden lizenziert werden können, um auch Endkunden bedienen zu können, die nicht den gesamten Umfang oder die gesamte Komplexität benötigen.

Mit zunehmender Verbreitung von Software und einer steigenden Vernetzung der Nutzer von Computersystemen wird es immer einfacher, auch Programme zu nutzen, die man nicht lizenziert hat. Wer sich auf einschlägigen Seiten im Internet umsieht, wird schnell erkennen, daß diejenigen Kreise, welche Kopierschutzmechanismen umgehen und ihre Erkenntnisse verbreiten, immer größer werden und besser miteinander vernetzt sind. Aus diesem Grund müssen sich auch die Verfahren zum Schutz von Software weiterentwicklen. Dabei ist das Verhältnis des Aufwands für den Schutz im richtigen Verhältnis zur tatsächlichen Bedrohung für das Produkt zu sehen.

Da aus oben genannten Gründen auch die Bedeutung von modularer Software für die Hersteller zunimmt, werden zunehmend Lösungen benötigt, die eine flexible Lizenzierung von Software an Endkunden ermöglichen. Fertige Systeme, die diese Problematik zu lösen versuchen, sind nur rudimentär verfügbar.

Diese Arbeit gibt einen Überblick über die auf dem Markt und in der Literatur beschriebenen Kopierschutzmechanismen und legt dar, wie diese genutzt und durch begleitende Maßnahmen verbessert werden können. Es werden mögliche Abläufe für die Lizenzierungsvorgänge beschrieben und Möglichkeiten zur flexiblen Vergabe von Lizenzen bei modularer Software vorgestellt. Für das flexiblen Hinzufügen von Modulen gibt es bislang keine Produkte, welche mehr als Grundfunktionen unterstützen. Auch konnte ich keine Vorschläge für deren Realisierung finden.

In dieser Arbeit sind Vorschläge zur flexiblen Lizenzierung modularer und erweiterbarer Software ausgearbeitet.

Die verschiedensten Anforderungen an Kopierschutz und modulare und erweiterbare Gestaltung von Software können nicht mit Patentrezepten gelöst werden. Diese Arbeit zeigt Möglichkeiten zur Lösung dieser Probleme um Modellierung flexibler Lizenzierung und Schutz vor unberechtigter Nutzung auf und vermittelt dem Entwickler Kriterien für die Auswahl der geeigneten Verfahren. Nicht jeder Entwickler modularer und erweiterbarer Software hat die Zeit und die Möglichkeiten, umfangreiche eigene Nachforschungen

über die verschiedenen Ansätze zum Vertrieb und Schutz solcher Produkte durchzuführen.

Mein Beitrag im Rahmen dieser Arbeit besteht darin,

a) die Lizenzierung sowie die modulare und erweiterbare Software, einschließlich der notwendigen Prozesse, zu beschreiben und einzuordnen,
b) die in der Literatur und vorhandenen Produkten erkennbaren Lösungsansätze für den Schutz vor unlizenzierter Verwendung zusammenzustellen und zu bewerten,
c) diese – ergänzt um eigene Lösungvorschläge – so auszuarbeiten, daß sie dabei helfen bei der Entwicklung von Softwareprodukten die richtigen Lösungen auszuwählen,
d) Abläufe beim Vertrieb modularer und erweiterbarer Software vorzuschlagen, zu klassifizieren und zu bewerten,
e) einen Vorschlag für die Gestaltung von Lizenzierungsinformationen bei modularer und erweiterbarer Software auszuarbeiten.

Dank

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben, insbesondere

- Herrn Prof. Steimann und seinen Mitarbeitern vom Lehrgebiet Programmiersysteme, besonders Frau Dr. Keller, für die Betreuung,
- meinem Arbeitgeber, der Interflex Datensysteme GmbH & Co. KG, einem Tochterunternehmen der Ingersoll-Rand Corporation, der mich logistisch und mit Anregungen versorgt hat, besonders den Herren Manfred Klostermeier, Albrecht Dietrich und Oliver Utz, die mir Einblick in deren bisherigen Lizenzierungs- und Vertriebsprozesse geben konnten; außerdem ermöglichte er mir während der Bearbeitungszeit der Arbeit eine flexible Urlaubsplanung und Gestaltung der Arbeitszeit,
- allen, die meine Arbeit Korrektur gelesen haben und mich auf Fehler und Unstimmigkeiten hingewiesen haben und mir – auch unfreiwillig – Anregungen für diese Arbeit gaben,
- Herrn Krzysztof Dyki, Doktorand an der Fakultät Informatik (Wydział Informatyki) der Technischen Universität Stettin (Politechniki Szczecińskiej) für seinen Buchtip [Eil05] und Informationen über seine Forschungen im Bereich im Bereich Dongles mit FPGAs (vgl. Abschnitt 6.3.1.3.2 und [Dyk06]),
- meinen Kollegen des Gemeinderats Durchhausen und allen Bürgern der Gemeinde, die meine ungewohnt schlechte Vorbereitung für die Sitzungen des letzten Vierteljahres ertragen mußten,
- allen, die mich bei Laune hielten und meine Launen ertragen mußten.

Inhalt

1. Einführung

1.1. Motivation

Komplexe Softwaresysteme enthalten heute eine Vielzahl von Funktionen, wobei nicht jeder Anwender jede dieser Funktionen auch tatsächlich nutzt. Um ein solches System auch für einfachere Aufgaben einsetzen zu können, wird der Funktionsumfang des Systems nachträglich verkleinert und dem Anwender eine Lizenz für die Nutzung dieser Untermenge erteilt. Der Anbieter eines solchen Systems kann mit einfachem Entwicklungsaufwand unterschiedlichste Anwenderanforderungen abdecken.

Firmen können mit ihrer modularen Software auf verschiedene Art und Weise Einkünfte erzielen:

- durch den Verkauf von Lizenzen für die Software an Endkunden,
- durch den Verkauf von Lizenzen an Dritthersteller, die es diesen erlauben, eigene Module in Ergänzung zu der Software zu vertreiben,
- durch Beratung,
- durch Dienstleistungen wie
  - Inbetriebnahme und Einstellung der Parameter,
  - Wartung,
  - Entwicklung von kundenspezifischen Zusatzmodulen.

Die Firmen haben ein Interesse, zu verhindern, daß die Software über ihren lizenzierten Umfang hinaus verwendet wird. Hierzu werden Sperrmechanismen benötigt; diese können sowohl durch Software alleine, als auch darüber hinaus durch spezielle Peripheriegeräte (“Dongles”) erreicht werden.

Bei der Vergabe einer Lizenz sind zwei Gruppen von Lizenznehmern zu unterscheiden: Endanwender, die vom Hersteller bereitgestellte Module nutzen möchten und Dritthersteller, die eigene Module zur Erweiterung des Systems entwickeln. Bei letzteren muß sichergestellt werden, daß beim Endanwender nur lizenzierte Module von Drittanwendern zum Einsatz kommen. Darüber hinaus sollen Dritthersteller auf Wunsch die Möglichkeit haben, die Lizenzen, die sie weitergeben, weiter einzuschränken und später selbständig bis zur vom Hersteller angegebenen Grenze zu erweitern.

Wünscht ein Kunde, daß ein zusätzliches Modul freigeschaltet wird, so sollte

a) dieser Vorgang ohne großen Aufwand durchzuführen sein und
b) die korrekte Abrechnung der neu erbrachten Leistung gewährleistet sein.

Die Möglichkeit, einzelne Funktionen für einen beschränkten Zeitraum zusätzlich freizuschalten, kann genutzt werden, um den Kunden zum Kauf weiterer Module zu bewegen.

Im Rahmen dieser Arbeit wird untersucht, welche Möglichkeiten es gibt, um

1. modulare Software vor nicht-lizenzierter Verwendung zu schützen,
2. modulare Software vor nicht-lizenzierten Modulen von Drittherstellern zu schützen,
3. Module temporär freizuschalten,
4. Module durch einen Techniker dauerhaft freizuschalten, wobei die Rechnungsstellung gewährleistet ist,
5. es dem Kunden zu ermöglichen, selbst weitere Module zu kaufen und sofort freizuschalten,
6. die Freischaltfunktionen vor Ort einzuschränken,
7. Beziehungen zwischen den Modulen zu modellieren,
8. Module flexibel zu lizenzieren,
9. Lizenzen an den Kunden zu vertreiben, d. h. die Prozesse und möglichen Vertriebsarten zu beschreiben.

Inzwischen spielen unlizenzierte Kopien von Software und illegaler Nachbau von Hardware auch außerhalb von PC-Umgebungen eine Rolle. So werden, wie in [Sch06] berichtet wird, vermehrt Maschinen einschließlich der Steuerungsprogramme 1:1 kopiert. Diese Arbeit legt einen Schwerpunkt auf Softwareschutz in PC-Systemen; viele der beschriebenen Methoden lassen sich jedoch auch auf andere Bereiche übertragen.

Diese Arbeit hilft einem Entwickler modularer und erweiterbarer Software und seiner Firma dabei,

- einen Überblick über die Möglichkeiten, Software modular und erweiterbar zu gestalten, zu gewinnen,
- die Rolle von Lizenzen kennenzulernen und die Notwendigkeit, seine Software vor unlizenzierter Nutzung zu schützen, abzuschätzen,
- aus den vorgeschlagenen Vertriebwegen den oder die geeigneten für das Produkt auszuwählen,
- Standardabläufe bei der Lizenzierung kennenzulernen und die geplanten Abläufe zu beurteilen,
- einen Überblick über die Möglichkeiten und Risiken der auf dem Markt angebotenen Kopierschutzprodukte zu gewinnen,
- aus den beschriebenen Verfahren zum Schutz vor unlizenzierter Nutzung die für sein Produkt geeigneten auszuwählen,
- sein Konfigurationsmanagement so anzupassen, daß es den Anforderungen modularer und erweiterbarer Software gerecht wird,
- Lizenzierungsinformationen so zu gestalten, daß Module flexibel freigeschaltet werden können.

Ziel dieser Arbeit ist es, ein Einführung in die Lizenzierung von und den Lizenzvertrieb bei Software – insbesondere auch modularer und erweiterbarer Software – zu geben, die Kopierschutztechniken, die in Produkten eingesetzt und in der Literatur beschrieben werden, darzustellen und eigene Ideen in den Bereichen Schutz vor unlizenzierter Nutzung sowie dem Kon-

figurationsmanagement und der Gestaltung von Lizenzierungsinformationen bei flexiblen modularen Produkten einzubringen.

1.2. Illustration

An den Stellen, an denen das Verständnis für die beschriebenen Sachverhalte durch Angabe eines Beispiels erleichtert wird, wird das Beispiel eines komplexen, modularen Systems in den Bereichen Zeitwirtschaft, Zutrittskontrolle, Personaleinsatzplanung und Betriebsdatenerfassung verwendet. Bei diesem Beispiel können Module wie die Betriebsdatenerfassung einzeln ausgeschaltet werden. Außerdem gibt es die Möglichkeit, Erweiterungen durch andere Hersteller, wie eine Koppelung an ein Lohnbuchhaltungssystem, anzuschließen. Schließlich gibt es in diesem Bereich auch Abhängigkeiten, so setzt beispielsweise die Koppelung mit einer Alarmanlage voraus, daß das Modul „Zutrittskontrolle“ aktiviert ist.

1.3. Aufbau

In dieser Arbeit werden in Abschnitt 2 zunächst der Begriff Lizenz definiert und die verschiedenen Arten von Lizenzen sowie die beteiligten Personen beschrieben. Im Folgenden wird dann in Abschnitt 3 näher auf wesentliche Merkmale modularer Software eingegangen. Im Abschnitt 4 wird auf Gründe für den Schutz vor unlizenzierter Nutzung, die verschiedenen Lizenzierungsmodelle und deren Folgen eingegangen. Darauf folgt in Abschnitt 5 die Beschreibung der Vertriebswege von Software und der zugehörigen Abläufe, die durchlaufen werden müssen, damit der Endkunde sein erworbenes und lizenziertes Programm auch nutzen kann. Der Abschnitt 6 beginnt mit einer Untersuchung und Bewertung der auf dem Markt angebotenen Kopierschutz-Produkte und der in der Literatur vorgeschlagenen Ansätze. Basierend auf diesen Techniken und um eigene Vorschläge ergänzt, werden Vorschläge zum Schutz vor unlizenzierter Nutzung und deren Vertrieb herausgearbeitet. Schließlich werden in Abschnitt 7 eigene Vorschläge für das Konfigurationsmanagement bei modularer Software sowie die flexible Modellierung von Lizenzierungsinformationen vorgestellt.

1.4. Beitrag dieser Arbeit

Mein Betrag in dieser Arbeit besteht darin,

- zu beschreiben, aus welchem Gründen modulare und erweiterbare Software benötigt wird und wie Software vor unlizenzierter Nutzung geschützt werden kann (in den Abschnitten 3 und 4),

- die Vertriebswege modularer Software zu klassifizieren und Kriterien für die Auswahl geeigneter Abläufe zu finden (im Abschnitt 5),
- die in der Literatur beschriebenen oder in bestehenden Produkten eingesetzten Lösungsansätze für den Schutz und Vertrieb lizenzierter Software zusammenzustellen und zu bewerten (im Abschnitt 6.2),
- auf Basis der bestehenden Lösungen und eigener Ideen Vorschläge für die Realisierung geschützter modularer und erweiterbarer Software herauszuarbeiten (im Abschnitt 6.3), dabei habe ich folgende Techniken für diese Arbeit neu entwickelt, für deren Einsatz im Bereich Lizenzierung ich weder in der Literatur noch bei bestehenden Produkten Anhaltspunkt finden konnte:
  - der Einsatz von RFID-, Magnet- oder Chipkarten als Träger für die Lizenzierungsinformationen (im Abschnitt 6.3.1.3.3),
  - die Prüfung biometrischer Merkmale zur Bindung einer Lizenz an eine Person (im Abschnitt 6.3.1.3.4),
  - den Vorschlag mit Hilfe eines speziellen Compilers kundenspezifischen Maschinencode zu erstellen (im Abschnitt 6.3.5.3),
  - den Vorschlag Schlüssel mittels Steganographie in Bildern oder Mediendateien zu verstecken (im Abschnitt 6.3.5.4),
  - die Verwendung nicht-lokaler Sprünge, um es zu erschweren, beim Beenden eines Programms aufgrund unlizenzierter Nutzung, die Ursache des Programmabbruchs zu erkennen (im Abschnitt 6.3.10 auf Seite 83),
- Vorgehensweisen für die Entwicklung und das Konfigurationsmanagement modularer und erweiterbarer Software vorzuschlagen (im Abschnitt 7.1),
- eine mögliche Implementierung der Lizenzierungsinformationen für flexible modulare und erweiterbare Software vorzuschlagen, wobei die zugehörigen Prozesse, die durchlaufen werden müssen, bis der Endkunde mit dem erworbenen Produkt arbeiten kann, mit einbezogen werden (im Abschnitt 7.2).

2. Begriffsbestimmungen

2.1. Lizenzen

2.1.1. Was ist eine Lizenz?

Das Urheberrecht regelt den Umgang mit „persönlichen geistigen Schöpfungen“ (vgl. § 2 [Urh03]). Der Urheber eines Computerprogrammes entscheidet nach § 69c UrhG [Urh03], wer das Programm

- vervielfältigen darf,
- übersetzen, bearbeiten, arrangieren oder umarbeiten darf,
- verbreiten darf und
- wiedergeben darf.

Das Recht zur Nutzung des Programmes, das der Eigentümer dem Benutzer zubilligt, wird als Lizenz bezeichnet.

Vielfach wird unter einer Lizenz auch die Menge von Daten oder Gegenständen verstanden, die benötigt wird, um ein Produkt, für welches man eine Lizenz erworben hat, auch tatsächlich nutzen zu können. Dabei kann es sich z. B. um einen Freischaltcode, eine CD-ROM oder einen Dongle handeln. Dieser Sachverhalt wird in dieser Arbeit – sofern dies zur Unterscheidung notwendig ist – mit Lizenzierungsinformation bezeichnet.

Es werden zwei verschiedene Klassen von Lizenzen unterschieden:

2.1.2. Freie Lizenzen

nutzt der Urheber, um allen oder einer bestimmten Gruppe von Nutzern eine Untermenge der Nutzungsmöglichkeiten zu erlauben, ohne dafür ein Entgelt zu erhalten[1].

So gibt es beispielsweise Software, deren Urheber die Nutzung für Mitglieder von Hochschulen ohne Entgelt freigegeben haben. Außerdem gibt es Lizenzen wie die GNU General Public License (GPL) [Fre], die jedem eine Nutzung erlauben, wenn bestimmte Bedingungen erfüllt sind.

2.1.3. Kommerzielle Lizenzen

vergibt der Urheber aus wirtschaftlichem Handeln gegen Entgelt. Er hat dabei das Interesse, daß nur Inhaber einer Lizenz das Programm auch nut-

[1] In der Literatur (vgl. [Wika]) wird der Begriff etwas enger definiert: Der Benutzer soll das Programm
a) für jeden beliebigen Zweck ausführen
b) untersuchen und modifizieren
c) kopieren
d) modifizierte Versionen weiter vertreiben
dürfen. In dieser Arbeit ist jedoch die Definition „in irgendeiner Form unentgeltlich nutzbar“ gemeint.

zen, und dies auch nur im Umfang der Lizenz. Programme, die kommerziell vertrieben werden, enthalten oft Code zur Verhinderung der unlizenzierten Nutzung; dies wird vereinfacht auch als Kopierschutz bezeichnet.

In dieser Arbeit wird die Problematik der kommerziellen Lizenzen behandelt, da freie Lizenzen i. allg. keines Schutzes vor Lizenzverletzungen bedürfen. Sofern ein Urheber seine Software nur an einem bestimmten Nutzerkreis und mit gewissen Einschränkungen vertreiben möchte, kann er dies auch in diesem Fall mit den in dieser Arbeit beschriebenen Methoden sicherstellen.

2.1.4. Share- und Freeware

sind Überbegriffe, die unterschiedlichste Lizenzierungsmodelle abdecken. Bei Freeware wird davon ausgegangen, daß ein Programm von jedermann ohne Entgelt nutzbar ist. Manchmal wird die Nutzung auf nicht-kommerzielle Anwendung beschränkt. Unter Shareware versteht man Programme, die zunächst kostenfrei getestet werden können, während für die normale Nutzung Entgelt verlangt wird. Manche Programme werden vor Erwerb der Vollizenz in ihrem Nutzungsumfang beschränkt, andere zeigen nur zusätzliche Hinweise an. Auch kann die Möglichkeit zur Nutzung der Testversion zeitlich eingeschränkt werden. Es gibt auch Software, die beliebig lange getestet werden könnte und alle Funktionen der Vollversion enthält; bei dieser wird jedoch über die Lizenz festgelegt, daß der Benutzer sie nur zum Test verwenden darf, d. h. auch wenn die Einhaltung der Beschränkung nicht kontrolliert wird, ist sie dennoch für den Benutzer bindend.

2.2. Beteiligte Partner am Vertrieb und Nutzung von Lizenzen

In diesem Abschnitt werden die einzelnen am Vertrieb und der Nutzung von Software-Systemen beteiligten Partner beschrieben.

2.2.1. Hersteller

Hersteller einer Software ist diejenige Firma bzw. Person, die das System entwickelt.

2.2.2. Dritthersteller

Ein Dritthersteller ist eine andere Firma bzw. Person, die zu einem bestehenden System eines Herstellers Erweiterungen entwickelt.

2.2.3. Endkunde

Der Endkunde erwirbt das Nutzungsrecht für dieses Software-System. Bei dem Endkunden kann es sich sowohl um eine natürliche als auch um eine juristische Person handeln.

2.2.4. Benutzer

Der Benutzer ist derjenige Mensch, der ein Software-System tatsächlich nutzt und bedient. Er kann, muß aber nicht, mit dem Endkunden übereinstimmen.

3. Modulare Software

3.1. Definition

Modulare Software ist (in Anlehnung an [Marb]) Software, die aus verschiedenen Modulen bestehen. Ein Modul ist dabei eine selbständige Einheit, die in einem abgrenzbaren Bereich arbeitet. Die Module verfügen über definierte Schnittstellen und sind somit austauschbar. Üblich ist, daß die Module alleine verwendbar sind. Aspekte, d. h. modulübergreifende Anliegen (vgl. [Scha]), werden im folgenden wie Module behandelt, da sie im Rahmen modularer Software auch eingeständig aktivierbar und deaktivierbar sein können. Module können in ihrem Umfang durch Angabe einer Beschränkung wie die maximale Anzahl Benutzer, die sie verwenden dürfen, eingeschränkt werden.

In der Praxis werden auch solche Untermengen der Gesamtfunktionalität als Module bezeichnet, deren Abschaltung weiterhin ein sinnvolles, lauffähiges Programm zurückläßt. In dieser Arbeit wird von letzterer Definition ausgegangen.

Vielfach dienen definierte Schnittstellen zu Modulen auch dazu, es Dritten zu ermöglichen, weitere Funktionalität in ein Produkt zu bringen; solche Module werden als “Plug-In” bezeichnet.

3.2. Gründe für modulare Software

3.2.1. Technische Gründe

Aus technischer Sicht gibt es im wesentlichen folgende Gründe für modulare Software:

- Das Programm läßt sich künftig leichter um neue Funktionalität erweitern.
- Funktionalität, die aus technischen oder fachlichen Gründen nicht gleichzeitig aktiv sein darf, kann gleichzeitig in einem Programm ausgeliefert werden, wobei über die Modulkontrolle die gleichzeitige Verwendung verhindert wird.
- Eine modulare Struktur erleichtert Drittherstellern oder Endkunden die Integration eigener Leistungsmerkmale.
- Durch Austausch von Modulen gegen solche mit analoger Funktionalität kann den unterschiedlichen Anforderungen und Möglichkeiten von Groß- und Kleinsystemen gerecht werden.

3.2.2. Betriebswirtschaftliche Gründe

Ein einmal erstelltes umfassendes System läßt sich ohne weitere Aufwände sowohl für die High-End-Speziallösung als auch für den Massenmarkt mit einfacherer Funktionalität einsetzen; dabei kann es dann für den jeweiligen Funktionsumfang zu marktgerechten Preisen angeboten werden. Dadurch vergrößert sich der Markt für ein Produkt.

Großsysteme können beispielsweise durch notwendige Performanceuntersuchungen in der Entwicklung höhere Aufwände verursachen als Standardanwendungen im kleineren Maßstab; aus diesem Grund werden die Lizenzkosten häufig an die Menge der zu verarbeitenden Daten gekoppelt.

3.3. Beziehungen zwischen Modulen

Zwischen den einzelnen Modulen gibt es verschiedene Beziehungen, die im folgenden dargestellt sind:

3.3.1. Enthaltensein in einem anderen Modul

Ein Modul ist in einem anderen vollständig enthalten. Im kombinierten Zutrittskontroll- und Zeitwirtschaftssystem ist z. B. das Modul „Urlaub verwalten“ im Modul „Zeitwirtschaft“ enthalten.

3.3.2. Unverträglichkeit mit einem anderen Modul

Zwei Module können nicht gleichzeitig miteinander genutzt werden, weil sie entweder in einem fachlichen Widerspruch stehen oder aus technischen Gründen nicht zusammenarbeiten können. Beispiele für einen fachlichen Widerspruch sind Koppelungen zu Fremdsystemen, die dem eigentlichen System gewisse Entscheidungen abnehmen; in diesem Fall kann immer nur eine Koppelung aktiv sein. Aus technischen Gründen kann von Modulen, die unterschiedliche Versionen einer Fremdbibliothek benötigen, welche aufgrund ihrer Schnittstellen nicht gleichzeitig aktiv sein können, immer nur eines aktiv sein. Ein anderes Beispiel für Unverträglichkeiten in einem Zeitwirtschaftssystem sind Kopplung an verschiedene Lohnabrechnungsprogramme, wobei beim einem Endkunden nur immer ein System gekoppelt sein kann.

3.3.3. Ein anderes Modul ist Voraussetzung

Ein Modul benötigt ein anderes Modul, da es einen Teil von dessen Möglichkeiten nutzt. In manchen Fällen gibt es auch eine Liste von alternativen

Modulen. Beispielsweise setzt das Modul „Berechnung von Überstunden“ voraus, daß die Arbeitszeit erfaßt wurde. Dazu wird mindestens eines der Module „Erfassung über Zeiterfassungsterminals“, „Erfassung über Web-Clients“ oder „Erfassung durch manuelle Eingabe“ benötigt.

3.3.4. Anteilige Module

Anteilige Module sind Module, bei denen der Umfang mindestens eines Leistungsmerkmals skaliert werden, wie z. B. die maximale Anzahl Mitarbeiter, die in einem System verwaltet werden können oder die maximale Anzahl Peripheriegeräte. Diese Verwendung dieser Art von Modulen kann dazu führen, daß verschiedene Leistungsmerkmale unterschiedlich eingeschränkt sind. Dies ist z. B. bei einem kombinierten Zeitwirtschafts- und Zutrittskontrollsystem der Fall, bei dem in einer Firma nur eine gewisse Anzahl Mitarbeiter vorhanden sind, jedoch darüber hinaus auch bestimmte Fremdfirmen und Gäste Zutritt zum Gelände bekommen können.

3.3.5. Basis- und Erweiterungsmodule

Module können grob in Basis- und Erweiterungsmodule eingeteilt werden. Dabei sind Basismodule solche, die entweder in jedem System vorhanden sein müssen, um es überhaupt nutzen zu können, oder sie haben in einem Unterbereich des Funktionsumfangs eine grundlegende Stellung, so daß die anderen Module ohne sie nicht arbeiten können. Basismodule könnten beispielsweise die Module „Basis Zeiterfassung“ oder „Basis Zutrittskontrolle“ sein.

Im Gegensatz dazu enthalten Erweiterungsmodule oft nur isolierte zusätzliche Leistungsmerkmale für einen Bereich, wie beispielsweise das Modul „Unterstützung des arabischen Kalenders“. Ein Erweiterungsmodul kann auch dazu dienen, ein anteiliges Modul in seinem Umfang zu erweitern. So kann beispielsweise das Modul „Besucherverwaltung für 100 Gäste“ um ein Erweiterungsmodul „Besucherverwaltung für 50 Gäste“ erweitert werden, so daß der Kunde danach bis zu 150 Gäste verwalten kann.

3.3.6. Pakete

Schließlich gibt es noch sogenannte Pakete, d. h. Sammlung von Modulen eines oder mehrerer Bereiche, welche die wichtigsten Funktionalitäten für ein bestimmtes Kundenprofil enthalten. Ein Beispiel für ein solches Paket ist „Zeitwirtschaft für den öffentlichen Dienst in den Kommunen Baden-Württembergs“.

3.4. Abgrenzung von Modulen

Eine Schwierigkeit bei der Aufteilung in funktionale Module liegt darin, daß man als Hersteller festlegen muß, wo die Grenzen zwischen den einzelnen Modulen sein sollen.

Ein Modul sollte inhaltlich so abgeschlossen sein, daß keine Paare von Modulen definiert werden, die zusammen eine funktionell nicht zu trennende Einheit bilden. So macht es in einem Zeitwirtschaftssystem keinen Sinn, die Kommen- und die Gehen-Buchung jeweils in ein eigenes Modul zu packen, da diese entweder zusammen oder überhaupt nicht benötigt werden. Im Gegensatz dazu kann die Möglichkeit, an einem Zeiterfassungsterminal seinen Urlaub einzutragen, getrennt davon gesehen werden.

Werden allerdings durch ihre Funktion festgelegte Module zu sehr eingeschränkt, erwirbt der Kunde eine zu eingeschränkte Lizenz und kann die gewünschte Funktionalität nicht nutzen. Das wäre beispielsweise der Fall, wenn ein Zutrittskontrollsystem zwar die Berechtigung einer Person prüft und anzeigt, für das Öffnen der Türe jedoch ein gesondertes Modul benötigt würde.

3.5. Module für den Endkunden

Der Endbenutzer benötigt Module, die funktional abgegrenzt sind. Er erwirbt Pakete von Leistungen, einzelne Leistungsmerkmale oder Erweiterungsmodule, wie er beim Kauf eines Neuwagens die einzelnen Zubehörteile festlegt. Für ihn ist es wichtig, daß diese Module so beschrieben werden, daß er anhand der eigenen Bedürfnisse die entsprechenden Lizenzen erwerben kann. Die technologischen Hintergründe spielen für die eigentliche Nutzung keine Rolle; daher muß der Endkunde bei seiner Bestellung mit so vielen Informationen versorgt werden, daß er weiß, welche Kombinationen technisch möglich und fachlich sinnvoll sind.

3.6. Module für den Dritthersteller

Für einen Dritthersteller kann auch eine andere Aufteilung von Modulen sinnvoll sein. Er lizenziert den Zugriff auf Applikation und Daten über eine API. In diesem Zusammenhang sind Module Untermengen der API.

Dabei sind verschiedene Aufteilungen möglich:

vertikal

Die Module sind gegeneinander durch ihre Funktionalität abgegrenzt, wie es auch der Endkunde erwartet. So kann es einem Dritthersteller gestattet werden, in einem kombinierten Zeitwirtschafts- und Zutrittssystem nur die Funktionalität zu nutzen, die sich mit Zeitwirtschaft befaßt.

horizontal
Die Grenzen eines Moduls werden nicht von der Funktionalität bestimmt, sondern von der Mächtigkeit der Möglichkeiten. So kann es einem Drittthersteller gestattet werden, Berichte über alle Daten zu erstellen, ohne daß er jedoch die Möglichkeit zu Änderungen an diesen hat.

kombiniert
Die horizontale und vertikale Modularisierung läßt sich auch kombinieren, wie z. B. bei einem Modul, das es erlaubt, Zutrittskontrolle (als vertikales Modul) durchzuführen mit der Prüfung biometrischer Merkmale (als horizontales Modul, da diese Prüfung auch bei der Zeiterfassung angewendet werden kann) kombiniert zu nutzen.

Wählt der Hersteller seine Module zu beschränkt, so hat er einen relativ hohen Verwaltungsaufwand, wenn einem Dritthersteller erst ein sehr kleiner Umfang lizenziert wird (der damit auch nur relativ geringe Einnahmen mit sich bringt) und dann immer wieder neue Module nachlizenziert werden müssen.

3.6.1. General- und Einzelkundenmodule

Das Recht des Drittherstellers darauf, ein bestimmtes Modul zu nutzen, kann entweder für beliebig viele an Endkunden zu vergebende Lizenzen oder nur für eine einzelne gewährt werden.

3.6.2. Eigene Module

Schließlich kann auch vorgesehen werden, dem Dritthersteller die Möglichkeit zu geben, eigene Module innerhalb seiner Erweiterungssoftware zu beschreiben. Diese Module können auch dazu führen, daß der Dritthersteller, wenn er seinem Kunden nur einen Teil der Module verkauft, auch nur einen Teil der Drittherstellerlizenz miterwerben muß, sofern diese an die Menge der verkauften Produkte gekoppelt sind.

3.7. Einschränkungen auf Modulebene

Im Abschnitt 3.3.4 wird beschrieben, wie ein Leistungsmerkmal wie die Anzahl Benutzer, die gleichzeitig mit dem System arbeiten dürfen, anhand seiner Quantität eingeschränkt werden kann. Solche Einschränkungen, die sich aus der Lizenzierung an die Endbenutzer ergeben, können nicht nur auf das Gesamtsystem, sondern auch auf einzelne Module angewandt werden. So könnte man in einem kombinierten Zeiterfassungs- und Zutrittskontrollsystem dem Endkunden gestatten, daß zu einem Zeitpunkt gleichzeitig fünf

Benutzer im Zeiterfassungsbereich arbeiten dürfen, aber zehn Benutzer im Zutrittskontrollbereich.

3.8. Modellierung und Implementierung komplexer modularer und erweiterbarer Strukturen

Bei der Implementierung modularer Software ist es hilfreich, die benötigten Modulstrukturen auf verschiedenen Ebenen zunächst einmal zu beschreiben. Als Hilfsmittel kann hier das “Feature Modelling” verwendet werden, wie es in [CE00, Kapitel 4] beschrieben ist. Feature Modelling umfaßt lt. [CE00, Abschnitt 4.3] alle notwendigen Aktivitäten zur Modellierung aller gemeinsamen und speziellen Eigenschaften von Konzepten und deren Abhängigkeiten untereinander, sowie die Organisation dieser in einem zusammenhängenden Modell. Diese Vorgehensweise ist dazu geeignet, die unterschiedlichsten Arten von Modulen zu beschreiben. Dabei können neben der Aufteilung in funktional abgegrenzte Einheiten parallel dazu auch Aspekte beschrieben werden. So werden beispielsweise die Lizenzen zur Nutzung von Zeitwirtschaft oder Zutrittskontrolle als Module beschrieben, während die Funktionalität zur Protokollierung aller Bedienereingaben – da sie auf praktisch alle Module wirkt – eher einen Aspekt beschreibt.

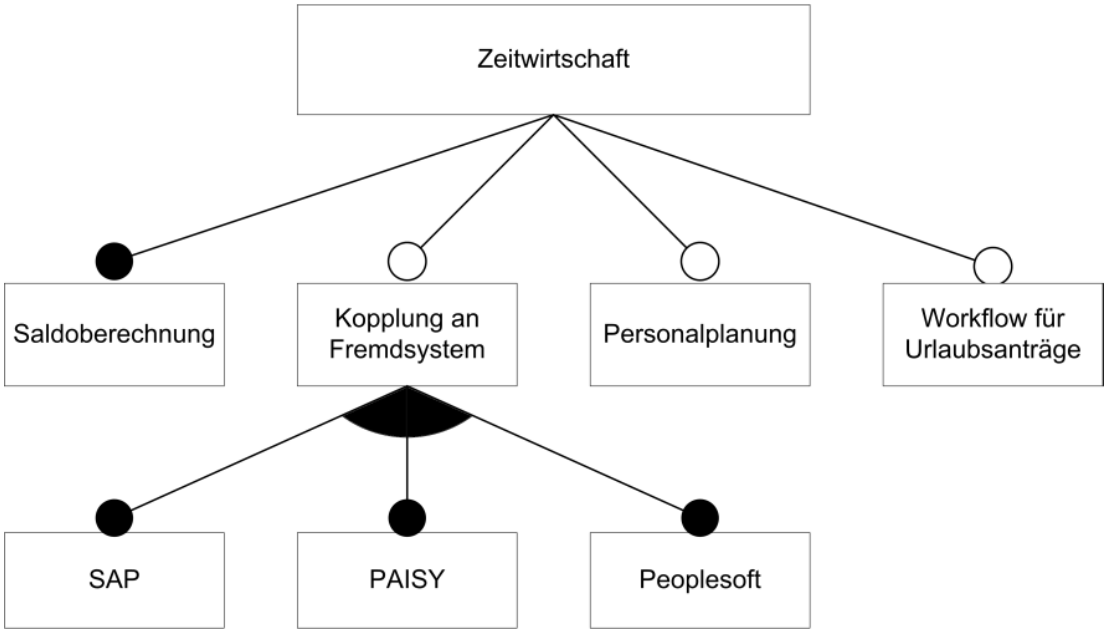

Abbildung 1: Feature-Diagramm am Beispiel Zeitwirtschaftssystem

Abbildung 1 zeigt ein Feature-Diagramm für ein Zeitwirtschaftssystem. Das Modul Saldoberechnung ist zwingend, während die Module Koppelung

(an Fremdsystem), Personalplanung und Workflow Urlaubsanträge optional sind. Das Modul benötigt zwingend genau eines der Module für die Koppelung an die Fremdsysteme SAP, PAISY oder Peoplesoft.

Schließlich stellt die Lizenzprüfungsfunktionalität selbst auch einen Aspekt dar, da sie auf alle Bereiche der Software wirkt und Bestandteil jedes einzeln lizenzierbaren Moduls ist.

Bei der Modellierung eines modularen und erweiterbaren Softwaresystems sollte dies berücksichtigt werden. Aspektorientierte Sprachen wie AspectJ [Ecl] können dabei sowohl bei der Implementierung der Lizenzprüfungsfunktionalität, als auch bei der Programmierung der Aspekte des Softwaresystems helfen. Es ist anzunehmen, daß die bestehenden Sprachen stark verbessert und neue dazukommen werden. Hilfreich wären an dieser Stelle auch Designtools, die es erlauben, ein Feature Model einzugeben und mit dessen Hilfe aspektorientiert Programme zu entwickeln.

3.9. Konfigurationsmanagement bei modularer Software

Die Norm DIN ISO 10007 [DIN03] stellt einen Leitfaden für das Konfigurationsmanagement dar. In dieser Norm wird Konfigurationsmanagement als „koordinierte Tätigkeiten zur Leitung und Lenkung der Konfiguration“ [DIN03, Abschnitt 3.6] definiert; dabei versteht man unter einer Konfiguration „miteinander verbundene funktionelle und physische Merkmale eines Produkts [...]“ [DIN03, Abschnitt 3.2]. Im Bereich der modularen und erweiterbaren Software läßt sich Konfigurationsmanagement auch als Vorgehensweise beschreiben, die dazu führt, daß bestimmte Leistungsmerkmale und Softwarestände in einem tatsächlich ausgelieferten Softwareprodukt enthalten sind.

Bei modularer Software ergibt sich hierbei die Schwierigkeit, daß sich nach dem erstmaligen Vertrieb eines Produkts die Konfiguration ändern kann. Bei erweiterbarer Software kommt erschwerend noch dazu, daß die Erweiterungskomponenten zum Zeitpunkt der Freigabe einer Konfiguration u. U. noch gar nicht feststeht. Eine Änderungslenkung im Sinne der Norm ist in den Fällen, in denen der Hersteller überhaupt keine Kenntnis von den Erweiterungen hat, nicht möglich. Aus diesem Grund ist bei der Konfigurationsmanagement-Planung darauf zu achten, daß die einzelnen Module (in der Norm als „Konfigurationseinheiten“ bezeichnet) derart gestaltet und die Schnittstellen so umfangreich dokumentiert werden, daß der Hersteller der Erweiterungssoftware („Dritthersteller“) für das durch seine Ergänzungen entstehende Gesamtprodukt die notwendigen Schritte für das Konfigurationsmanagement durchführen kann. Bei den herstellereigenen Modulkonfigurationen ändert sich die Aufgabe der Verfügungsstelle dahingehend, daß diese nicht mehr Entscheidungen über eine jeweils konkrete Konfiguration trifft, sondern Richtlinien festlegt, welche Konfigurationen (d. h. Modulzusammenstellun-

gen) den Kunden ermöglicht werden sollen. Analog dazu sollte bei von Drittherstellern durchgeführten Erweiterungen vorgegangen werden; in diesem Fall bietet sich an, mit diesen zu vereinbaren, daß sie für die Qualitätskontrolle des neuen erweiterten Produkts verantwortlich sind; so kann sich der Hersteller davor schützen, für durch Wechselwirkung mit dem Fremdprodukt entstandene Fehler zu haften, ohne einen Einfluß darauf zu haben. Modulare und erweiterbare Programme stellen höhere Anforderungen an genaue Definition und Dokumentation von Schnittstellen und des Verhaltens von Modulen als Programme, die zwar modular entwickelt, aber nur als Einheit bzw. eine stark begrenzte Menge von Modulen vertrieben werden.

4. Schutz vor unlizenzierter Nutzung

4.1. Einführung

4.1.1. Weshalb wird Schutz benötigt?

Es gibt drei Klassen von Geschäftsmodellen, mittels derer sich in der Software-Industrie Erträge erwirtschaften lassen und die sich bezüglich ihrer Schutzwürdigkeit unterscheiden:

4.1.1.1. Entwicklung projektbezogener kundenspezifischer Software

Ein Kunde vergibt beispielsweise einen Werkvertrag über die Erstellung einer bestimmten auf ihn zugeschnittenen Software. Er vergütet die Leistung nach Aufwand und erhält das Recht zur Nutzung und oft auch zur Weiterentwicklung des Programms. Eine Nutzung durch Dritte ist aufgrund der Leistungsmerkmale der Software nicht interessant. Da daher ein Verkauf von Lizenzen an andere Kunden keinen Erfolg verspricht, sind i.d.R. keine besonderen Schutzmaßnahmen erforderlich. Eine Ausnahme davon stellt eine zeitliche Beschränkung der Software dar, die vom Hersteller erst nach Bezahlung der Leistungen aufgehoben wird.

4.1.1.2. Unentgeltliche Weitergabe der Software und Erwirtschaften des Ertrags über Service und Beratung

Der Hersteller verbreitet die von ihm erstellte oder andere Software im Rahmen einer freien Lizenz (vgl. Abschnitt 2.1.2). Er sieht dabei die Nutzer der Software als potentielle Kunden für Service- und Beratungsleistungen. Daher liegt sein Interesse an einer möglichst großen Verbreitung der Software. Mit derartigen Geschäftsmodellen arbeiten beispielsweise viele Hersteller von Linux-Distributionen. Ein besonderer Schutz der Software würde diesem Ansinnen entgegenstehen und ist daher schädlich.

4.1.1.3. Vergabe von Software-Lizenzen an mehrere Kunden eines Marktes

Die Software wird einem breiten Markt unter einer kommerziellen Lizenz (vgl. Abschnitt 2.1.3) angeboten. Um Ertrag zu erwirtschaften, müssen einem möglichst großen Kundenkreis – bei gleichem einmaligen Entwicklungsaufwand – Lizenzen verkauft werden. All diejenigen, welche die Software bereits ohne gültige Lizenz nutzen, haben oft kein Interesse daran, die Software zu kaufen, da sich an ihren tatsächlichen Nutzungsmöglichkeiten nichts ändert. Viele Programme werden im nicht-öffentlichen Bereich genutzt, so daß sich die zivil- und strafrechtliche Verfolgung von Personen, die im Widerspruch zum Urheberrecht Software nutzen, schwierig gestaltet. Für den Hersteller

ist es dann effektiver, die unlizenzierte Nutzung mit technischen Maßnahmen zu verhindern.

4.1.1.4. Mischformen

Über die drei beschriebenen Klassen hinaus gibt es noch Mischformen mit eigenem Schutzbedürfnis. Eine modulare und in großem Maße parametrierbare Software wird zwar an einen großen Kundenkreis verkauft, muß jedoch vom Hersteller oder Händler an den einzelnen Kunden angepaßt werden. Bei solchen Produkten ist ein Schutz vor unberechtigter Nutzung nicht-lizenzierter Module wichtiger als ein Schutz vor Nutzung durch Dritte, welche die Software ohne entsprechende Dienstleistungen im allgemeinen nicht produktiv nutzen könnten.

4.1.2. Wozu Versionen mit künstlich eingeschränkten Nutzungsmöglichkeiten?

Welche Gründe gibt es, Software in ihren Nutzungsmöglichkeiten einzuschränken?

4.1.2.1. Betriebswirtschaftliche Gründe

Um in einem Markt mit Wettbewerb ein Produkt erfolgreich verkaufen zu können, bedarf es sowohl eines den Bedürfnissen der Abnehmer entsprechenden Produkts, als auch eines marktgerechten Preises. Für ein umfangreiches komplexes Programm läßt sich somit ein hoher Preis erzielen, der dem Hersteller einen gewissen Ertrag sichert. Ein einfaches Programm mit beschränktem Umfang erfordert zwar geringeren Entwicklungsaufwand, kann jedoch auch nur einen niedrigeren Preis erzielen. Schafft es jedoch ein Hersteller eines komplexen Produktes, dieses durch technische Maßnahmen so einzuschränken, daß es dann die einfachen Aufgaben noch lösen kann, läßt sich mit einfachem Entwicklungsaufwand ein größerer Markt abdecken. Der größere Markt ermöglicht mehr Umsatz und dieser bedeutet bei gleichen Aufwänden einen höheren Ertrag. Dazu kommt, daß es einem Kunden, der bereits die einfache Version lizenziert hat und sich an die Anwendung des Programms gewöhnt hat, erleichtert wird, in komplexere Lösungen zu investieren. Diese Kundenbindung führt zu einer besseren Position des Herstellers auf dem Markt.

Nachteil einer solchen Lösung ist, daß technische Lösungen, die auf Großsysteme mit entsprechender Infrastruktur ausgelegt sind, bei Kleinsystemen u. U. nicht praktikabel sind. Dieser Effekt kann abgeschwächt werden, indem der Hersteller in Form von Modulen verschiedene Lösungen für die unterschiedlichen Größenordnungen von Systemen zur Verfügung stellt. So kann der Hersteller in einer Software, die relationale Datenhaltung benötigt, in

großen Systemen ein kommerzielles, komplexes Datenbanksystem einsetzen, in Kleinsystemen jedoch eine freie Lösung für geringe Datenmengen.

4.1.2.2. Juristische und politische Gründe

Auch Gesetze in verschiedenen Rechtsräumen können dazu führen, daß einzelne Leistungsmerkmale eines Programms abschaltbar sein müssen. So kennt beispielsweise Frankreich ein Verbot von Kryptographie ab einer gewissen Stärke, so daß für französische Kunden die Verschlüsselung abschaltbar sein müßte. In Nordamerika ist es möglich Patente auf Software zu erlangen; dies kann dazu führen, daß Module, die in Europa problemlos verkauft werden können, dort entweder überhaupt nicht oder nur mit Lizenzzahlungen an den Patentinhaber vertrieben werden können. Außerdem kennen viele Staaten Exportbeschränkungen und Embargos für die Lieferung bestimmter Technologien wie kryptographischen Algorithmen an gewisse Staaten (so die Vereinigten Staaten für Lieferungen an den Iran, Kuba oder Nordkorea).

Außerdem kann komplexe Software mehrere Geschäftsbereiche abdekken. Hersteller können sich beispielsweise dafür entscheiden, an einen Mitbewerber in einem solchen Bereich nur die Module aus den anderen Bereichen auszuliefern.

4.2. Arten von Lizenzen

4.2.1. Endanwender-Lizenz

Endanwender-Lizenzen werden i.d.R. für die Nutzung und Ausführung eines Programms vergeben. Ein Endanwender soll keine Veränderungen am Programm durchführen oder dies weiterverbreiten.

Aber auch die Nutzung des Programms unterliegt oft bestimmten Einschränkungen; diese werden im folgenden dargestellt:

4.2.1.1. Lizenz an Benutzer gekoppelt

Nur ein Benutzer darf das Programm nutzen. Um einen unlizenzierten Benutzer sicher ausschließen zu können, müßte ein spezifisches Merkmal des Benutzers kontrolliert werden. Codes oder Kennwörter erfüllen diese Bedingung nicht, da sie weitergegeben werden könnten.

4.2.1.2. Lizenz an einen Benutzer gleichzeitig gekoppelt

Nur ein Benutzer darf das Programm zu einem Zeitpunkt nutzen. Um unlizenzierte Benutzung auszuschließen, müßten alle Geräte, auf denen eine Nutzung möglich wäre, miteinander in Verbindung stehen und dies überprüfen.

4.2.1.3. Lizenz an eine Maschine gekoppelt

Das Programm darf nur auf einer bestimmten Maschine benutzt werden. Zur Überprüfung der Lizenz würde genügen, daß diese Maschine eindeutig identifiziert werden kann und Merkmale besitzt, die sie von jedem anderen System unterscheiden.

4.2.1.4. Firmenlizenzen

Das Programm darf nur im Rahmen der Geschäfttätigkeit von den Mitarbeitern einer Unternehmung genutzt werden. Manche Produkte erlauben darüber hinaus sogar noch die private Nutzung durch den Mitarbeiter.

4.2.1.5. Bestimmte Anzahl von Benutzern gleichzeitig

Eine festgelegte Anzahl von Benutzern darf die Software gleichzeitig nutzen. Dies wird auch als “floating licences” (vgl. Abschnitt 6.2.2.4 auf Seite 51) bezeichnet. Die Nutzung kann entweder von einer zentralen Maschine kontrolliert werden, die über ein Netzwerk mit allen anderen Systemen verbunden ist, oder ein Netz von gleichberechtigen Systemen stimmt die Nutzung mit der Umgebung ab.

4.2.1.6. Beschränkung auf Leistungsmerkmale der Hardware

Die Nutzung der Software wird nur für Hardware bis zu einer bestimmten Leistungsgrenze lizenziert. Beispiele für solche Beschränkungen sind:

- Anzahl Prozessoren,
- maximale Taktfrequenz des Prozessors,
- maximale Summe der Taktfrequenzen der Prozessoren,
- maximale Anzahl Schnittstellen zu anderer Hardware (z. B. Anzahl serieller Schnittstellen).

4.2.1.7. Beschränkung auf Leistungsmerkmale der Software

Außerdem kann die Nutzung der Software auf einen bestimmten Umfang der Nutzung gewisser Leistungsmerkmale beschränkt werden. Beispiele für solche Grenzen sind:

- maximale Datenbankgröße,
- maximale Anzahl Clients,
- maximale Anzahl bestimmter Datensätze (z. B. Anzahl Personen, die in einem Zutrittskontrollsystem verwaltet werden),
- maximale Anzahl Bearbeitungsvorgänge,
- maximale Anzahl Schnittstellen zu anderer Software (z. B. Koppelungen zu anderen Systemen).

4.2.1.8. Evaluierungsversionen

Zu Demonstrationszwecken werden Testversionen eingesetzt. Um zu verhindern, daß diese auf Dauer vom Kunden genutzt werden können, ohne daß dieser eine vollständige Version erwirbt, sind folgende Maßnahmen üblich:

- Beschränkung der Nutzungsdauer auf einen bestimmten Zeitraum oder eine bestimmte Anzahl von Aktionen (Nutzungszeit in Minuten, Bearbeitungsschritte, Programmstarts o.ä.),
- Beschränkung der Nutzungsdauer bis zu einem festgelegten Enddatum (z. B. für das Ende eines kostenfreien Beta-Tests),
- Einschränkung der Funktionalität auf Kernfunktionen (so sind die Bearbeitungsfunktionen eines Editors vorhanden, das Ergebnis kann aber nicht gespeichert werden),
- Zeitverzögerungen im Programmablauf und zusätzliche Informationsdialoge, die darauf hinweisen, daß das Programm nur zu Testzwecken genutzt werden darf,
- Beschränkung von Netzwerkversionen auf das lokale System,
- Markierung von Arbeitsergebnissen, wie Hinweise auf die Testversion in Ausdrucken, auf die der Benutzer keinen Einfluß hat.

4.2.2. Erweiterungslizenz für Dritthersteller

Der Hersteller kann es einem Dritthersteller erlauben, Erweiterungsmodule für die Applikation zu erstellen. Wahlweise kann daran die Lizenz zur Nutzung des zugehörigen normalen Moduls des Herstellers gekoppelt werden (im diesem Fall erhält der Dritthersteller z. B. eine Generallizenz für ein Modul und kann Unterlizenzen an die Endkunden vergeben).

In diesem Zusammenhang sollte definiert werden, ob Endkunden Funktionalität, die sie nicht lizenziert haben, nutzen dürfen, wenn sie Bestandteil eines von ihnen ordnungsgemäß lizenzierten Erweiterungsmodul eines Drittherstellers sind.

Auf folgende Art und Weise kann ein Hersteller einem Dritthersteller Lizenzen zur Erstellung von Erweiterungen erteilen:
Der Dritthersteller darf Funktionen des System in einem definierten, aber nicht notwendigerweise beschränkten Umfang verwenden und das damit erstellte Programm an einen einzelnen Kunden verkaufen. Alternativ kann ihm auch gestattet werden, diese Applikation an beliebig viele Kunden zu vertreiben. Außerdem kann das Recht zur Entwicklung von Software, die diese Schnittstellen nutzt, entweder an eine gesamte Firma oder an einen oder eine definierte Anzahl von Entwicklern vergeben werden.

4.3. Rolle temporärer und funktionsbeschränkter Lizenzen

Ein Hersteller kann zeitlich und in der Funktion beschränkte Lizenzen vergeben. Auf diese Art und Weise können beispielsweise Interessenten die Funktionalität des Systems prüfen.

Ein weiteres Einsatzgebiet zeitlich beschränkter Lizenzen ist die Überbrükkung der Zeit, bis eine Bestellung abgewickelt ist bzw. die Rechnung bezahlt wurde. Dies gilt gleichermaßen für Neusysteme als auch für temporär aktivierte neue Module. In dieser Zeit kann der Endkunde das System im vollen Umfang nutzen und Verzögerungen werden vermieden.

4.4. Verhinderung der unlizenzierten Nutzung

4.4.1. Notwendigkeit des Schutzes

Das Urheberrechtsgesetz [Urh03] gesteht dem Urheber einer Software das Recht zu, darüber zu entscheiden, wer diese in welchem Umfang und unter welchen Bedingungen nutzen darf. Würde man davon ausgehen, daß die Nutzer der Software sich an das Recht halten, bedürfte es keiner weiteren Maßnahmen, um eine eingeschränkte Lizenz durchzusetzen. In der Praxis gibt es zwei Möglichkeiten im Umgang mit unlizenzierter Nutzung:

1. Die Software braucht nicht besonders geschützt werden, wenn
   - sie derart speziell ist, d. h. für einen einzelnen Nutzer oder eine kleine Nutzergruppe entwickelt wurde, so daß neben den Kunden, die eine Lizenz erworben haben, sie niemand mehr produktiv nutzen kann.
   - Die Software ist hoch-komplex und stark parametrierbar, so daß ein Einsatz ohne Unterstützung des Herstellers unwahrscheinlich ist.
   - Die Software dient hauptsächlich zur Nutzung spezieller Hardware und kann ohne diese nicht genutzt werden.
2. Die Software benötigt Schutz vor unlizenzierter Nutzung, wenn
   - sie von einem großen Kreis an Nutzern produktiv genutzt werden kann (z. B. Textverarbeitungsprogramm oder Betriebssystem),
   - sie zur Unterhaltung dient und für einen überschaubaren Zeitraum genutzt wird (z. B. Spiele),
   - sie sehr modular vertrieben wird, um sicherzustellen, daß dem Endkunden auch die tatsächlich genutzte Funktionalität berechnet wird; dies gilt selbst dann, wenn eine im ganzen unlizenzierte Nutzung nicht zu erwarten ist.

Dazu kommen Programme, die in ihrer Schutzwürdigkeit zwischen den beiden Extremen angesiedelt sind. Dazu gehören die hoch-komplexen Programme, die der Einrichtung und Wartung durch den Hersteller bedürfen und bei denen daher nicht zu erwarten ist, daß sie ohne jegliche Lizenz

genutzt werden. Dennoch muß sichergestellt sein, daß nur die lizenzierten Leistungen bzw. Module auch durch den Kunden nutzbar sind, da ansonsten der Hersteller keine Möglichkeit mehr hat, diese als Erweiterungslizenzen zu vertreiben.

4.4.2. Arten von Schutz

Der Schutz vor unlizenzierter Nutzung von Programmen läßt sich mit verschiedenen Maßnahmen oder einer Kombination derselben erreichen.

4.4.2.1. Juristische Maßnahmen

In bekanntgewordenen Fällen von Urheberrechtsverletzungen wird Strafanzeige gegen den Täter erstattet und es werden Schadenersatzforderungen an diesen gestellt und, wenn notwendig, auch auf dem Klageweg durchgesetzt. Diese Maßnahmen sollen einerseits dazu führen, daß der Hersteller seine entgangenen Umsätze ausgleichen kann, andererseits sollen sie andere davon abschrecken, die Software ebenfalls unlizenziert zu nutzen. Allerdings sind derartige Maßnahmen nicht immer bei der breiten Masse der Bevölkerung populär, wie es sich bei den Reaktionen auf eine Kampagne der Filmwirtschaft gegen das nichtlizenzierte Kopieren und Verbreiten von Filmen zeigt (vgl. den Artikel [Hei03a] und die Kommentare im zugehörigen Forum).

4.4.2.2. Kontrollen

Die Kontrolle darüber, ob ein Softwaresystem ohne gültige Lizenz verwendet wurde, ist bei nicht-öffentlich genutzten Programmen schwierig. Maßnahmen, die einen Zugriff auf das nicht öffentlich zugängliche System des Kunden erfordern würden, wären ohne dessen Zustimmung strafbar (vgl. § 202a StGB [StG05]). Bei öffentlichen Systemen kann i. d. R. leicht festgestellt werden, welche Software verwendet wird. Falls der Hersteller alle seine Kunden kennt, könnte er damit überprüfen, ob das entsprechende System lizenziert ist. Werden die Lizenzen allerdings anonym und über Zwischenhändler vertrieben, ist dies nicht so einfach möglich. In diesem Fall ist dies nur mit Hilfe von technischen Maßnahmen, wie sie weiter unten beschrieben werden, durchzuführen.

4.4.2.3. Anreize zur Meldung unlizenzierter Nutzung

Eine Möglichkeit, die Wirksamkeit von Kontrollen auch auf den nicht-öffentlichen Bereich auszuweiten, ist es, Dritte, die von der unlizenzierten Nutzung von Software Kenntnis erhalten, dazu zu bewegen, dies dem Hersteller mitzuteilen. Anreiz für diese Personen könnte eine Geld- oder Sachprämie sein.

Eine solche Vorgehensweise könnte von vielen allerdings als Denunziation abgelehnt werden.

4.4.2.4. Technische Maßnahmen

Es gibt zwei Klassen von Vorgehensweisen; auf sie wird im Abschnitt 6 näher eingegangen:

a) Die unlizenzierte Nutzung wird durch geeignete Hard- und/oder Software verhindert.
b) Die unlizenzierte Nutzung wird so gekennzeichnet, daß sie leicht von außen zu Erkennen ist, z. B. in den Arbeitsergebnissen.

Beide Methoden setzen voraus, daß eine in einem konkreten System laufende Software erkennen kann, ob sie korrekt lizenziert wurde oder nicht.

4.5. Weitere Aspekte geschützter Software

4.5.1. Nutzung ohne gültige Lizenz

Neben den durch das Urheberrecht bestimmten straf- und zivilrechtlichen Folgen der Nutzung unlizenzierter Software können sich für den Benutzer auch weitere Probleme ergeben:

- Zur Umgehung von Schutzmechanismen werden oft Programme angeboten, die entweder gültige Lizenzierungsinformationen erzeugen (sogenannte “Keygens”) bezeichnet) sucht, kann sich davon überzeugen, daß dies bei vielen Programmen so schon geschehen ist (vgl. die 10 Millionen gefundenen Einträge bei Google [Goob]). oder die geschützte Software so verändern oder ergänzen, daß die Lizenzprüfung umgangen wird (“Cracks”). Für diese Programme wird keine Gewähr für die Funktion übernommen, die Quellen sind oft nicht verfügbar und sie können Viren enthalten, Hintertüren auf dem eigenen System aufmachen oder sonstige Schadfunktionen enthalten.
- Beim Vertrieb mancher Softwareprodukte erwirbt der Kunde das Recht auf weitere Leistungen wie Unterstützung bei der Installation oder Behebung von Fehlern (dies i. d. R. schon im Rahmen der gesetzlichen Gewährleistung) mit. Jemand, der die Software unlizenziert nutzt, kann diese nicht wahrnehmen.
- Manche Programme sind derart komplex, daß sie ohne entsprechende Schulung oder Unterstützung des Herstellers nicht produktiv zu betreiben sind.
- Gibt der Hersteller Updates oder Fehlerbehebungen heraus, gibt es keine Garantien, daß auch diese weiterhin genutzt werden können.

Diese Punkte tragen neben rechtlichen und ethischen Gesichtspunkten dazu bei, daß Menschen von der unlizenzierten Nutzung von Software abgehalten werden.

4.5.2. Probleme bei der Nutzung geschützter Software

Andererseits können technische Schutzmechanismen den Benutzer auch davon abhalten, ein Programm im Rahmen seiner Rechte zu nutzen.

So hat der Benutzer beispielsweise das Recht Sicherungskopien von den Originaldatenträgern zu erstellen, die er im Falle der Zerstörung von letzterem nutzen kann, um die Software erneut installieren zu können. Kopierschutzmechanismen, die ein Programm an diesen Datenträger binden, verhindern dies.

Probleme können auch Programme bereiten, die nur auf einem bestimmten Hardware lauffähig sind. Der Bundesgerichtshof hat in seinem Urteil vom 6. Juli 2000 (Az. 1 ZR 244/97, nachzulesen unter [Bun]) klargestellt, daß die Beschränkung einer Lizenz auf konkrete Hardware und damit auch die Verhinderung des Weiterverkaufs die Rechte des Kunden einschränken kann. Selbst wenn es das Urheberrecht ermöglicht, beim direkten Vertragsabschluß mit dem Kunden die Lizenz an ein konkretes System zu binden – wie es [Bun, Abs. 19 ff] nahelegt – würde dies für den Kunden bedeuten, daß er nur eine sehr eingeschränkte Leistung erwirbt. Außerdem ist es bei Ausfall des Systems oder einer seiner Komponenten, nicht ohne weiteres möglich, die Software auf einem Ersatzsystem zu nutzen.

Die fehlende Möglichkeit, Sicherungkopien zu erstellen und die Tatsache, daß es schwierig ist, bei Ausfall eines Systems auf ein Ersatzsystem auszuweichen, führt dazu, daß der Endkunde im Fehlerfall nicht mehr dazu in der Lage ist, Software zu nutzen, für die er eigentlich ein Nutzungsrecht erworben hat. Ein Indiz für die große Bedeutung dieses Problems ist, daß sich inzwischen selbst „populärwissenschaftliche“ Zeitschriften (wie in [BIL06]) damit beschäftigen.

Schließlich können fehlerhaft implementierte Kopierschutzmaßnahmen die Stabilität, Integrität oder Sicherheit des Systems des Kunden beeinflussen. Beispiele hierfür sind ein CD-Kopierschutz der Firma Sony, der offenbar ein Rootkit auf dem Rechner des Nutzers installiert hat (vgl. [Hei05]) oder die Diskussion über Fehler in der Implementierung von Star-Force (vgl. [Kla06]).

5. Vertriebswege und Abläufe bei der Lizenzierung

5.1. Vertriebswege

Um einem Endkunden eine Lizenz für ein Softwareprodukt zu verkaufen, sollten i. allg. folgende Schritte erfolgreich durchlaufen sein:

1. Der Kunde bestellt eine Software.
2. Das System wird beim Kunden installiert (entweder von ihm selbst, dem Hersteller oder einem dritten Dienstleister).
3. Der Kunde erhält die Lizenz und ggf. die zum Betrieb notwendigen Lizenzierungsinformationen.

Diese Schritte können auf verschiedenen, im folgenden beschriebenen, Wegen erreicht werden. Die Gestaltung des Lizenzvertriebs und des Softwareschutzes müssen dann darauf abgestimmt werden. Vielfach werden für ein Produkt auch mehrere dieser Wege beschritten.

5.1.1. Direktvertrieb

Der Kunde bestellt die Software direkt beim Hersteller und dieser liefert sie auch direkt an den Kunden aus. Oft werden die Programmdateien und Lizenzierungsinformationen über Versandhandel oder mittels eines Netzwerks (wie des Internets) weitergegeben. Auf diese Art und Weise vertriebene Software kann i. allg. ohne großartige Kenntnis des Programms vom Nutzer installiert und betrieben werden. Auf diesem Wege werden oft sogenannte “Shareware”-Programme (vgl. Abschnitt 2.1.4) verbreitet. Diese Nutzbarkeit, ohne daß es großer Schulung bedarf, hat zur Folge, daß Software, die auf diesem Wege vertrieben wird, besonders anfällig für unlizenzierte Nutzung ist und daher besonderen Schutzes bedarf.

5.1.2. Verkauf über Vertriebsniederlassung

Der Hersteller verfügt in seinen Märkten über ein Netz von Niederlassungen, die den eigentlichen Vertrieb an den Kunden übernehmen und oft auch Dienstleistungen wie Einrichtung der Applikation, Schulung der Nutzer oder Wartung des Systems anbieten. Oft bieten die Niederlassungen auch kleinere Software-Anpassungen oder -Erweiterungen an. Beim Lizenzschutz liegt das Hauptaugenmerk darauf, sicherzustellen, daß diejenigen Leistungen, welche der Kunde nutzen kann, auch tatsächlich in Rechnung gestellt wurden. Ansonsten wäre es nicht mehr möglich, über den Verkauf von Erweiterungen zusätzlichen Ertrag zu erwirtschaften.

5.1.3. Verkauf über Händler

Die Software kann vom Kunden nicht direkt beim Hersteller erworben werden, sondern nur über einen Händler. Dieser erwirbt i. d. R. gleich eine größere Anzahl von Lizenzen, um diese dann an seine Kunden weiterzuverkaufen. Vorteil für den Hersteller ist, daß er kein eigenes Vertriebsnetz aufrecht erhalten muß. Die Händler haben die Möglichkeit, verschiedene – auch konkurrierende – Produkte anzubieten und können ihr Risiko so minimieren. Der Kunde hat die Chance, viele Produkte beim Händler seines Vertrauens zu erwerben. Je nach Komplexität der Software können entsprechend ausgebildete Händler auch weitere Dienstleistungen wie Schulungen erbringen. Beim Lizenzschutz muß auch darauf geachtet werden, daß Händler nicht mehreren Endkunden ein und dieselbe Lizenz mehrfach verkaufen.

5.1.4. OEM-Verkauf

Sogenannte Original Equipment Manufacturers (OEMs) verkaufen ihre Software nicht an Kunden, sondern an andere Firmen, welche diese dann ggf. kombiniert mit anderen Bestandteilen als System verkaufen. So gibt es Firmen, die ein Zeitwirtschafts- und ein Personalplanungsmodul bei jeweils anderen Herstellern zukaufen, diese koppeln und an Zeitwirtschaftsterminals eines weiteren Herstellers koppeln.

Eine Variante des OEM-Verkaufs ist der Verkauf über Dritthersteller, die ein Erweiterungsmodul in die eigentliche Applikation integrieren und dann an den Endkunden verkaufen. Beispielsweise kann eine Firma ein Zeitwirtschaftssystem eines Herstellers nutzen, um eine Betriebsdatenerfassungskomponente ergänzen und das Gesamtprodukt dann an den Kunden verkaufen.

5.1.5. Vermietung/Leasing

Neben einem dauerhaften Nutzungsrecht kann der Hersteller auch zeitlich begrenzte Lizenzen an die Kunden vergeben. Dabei wird das Entgelt nach der Nutzungsdauer festgelegt. Bei Systemen, deren Daten aus handelsrechtlichen oder steuerlichen Gründe eine gewisse Zeit vorhanden sein müssen, ist dies bei der Gestaltung zu berücksichtigen, so daß nach Ablauf der Mietdauer die Daten für diese Zwecke noch verfügbar sind.

Beim Schutz vor unlizenzierter Nutzung ist darauf zu achten, daß die Nutzung nach Ablauf der vereinbarten Zeit nicht oder nicht mehr im vollen Umfang möglich ist. Oft werden solche Verträge mit Wartungsdienstleistungen gekoppelt.

5.1.6. Application Service Providing (ASP)

Unter “Application Service Providing” versteht man, daß eine Firma einen bestimmten Dienst auf einem zentralen System für andere Firmen anbietet. Letztere Firmen brauchen dieses System dann nicht mehr selbst zu unterhalten und greifen über eine Netzwerkverbindung darauf zu. Die Software muß sicherstellen, daß die verschiedenen Kunden nur auf ihren eigenen Daten arbeiten können. Das Zurverfügungstellen des Systems wird entweder pauschal nach Betriebsgröße oder nach tatsächlichen Vorgängen berechnet.

5.1.7. Direktfreischaltung von Updates

Auch wenn eine Software ursprünglich auf einem anderen Weg vertrieben wurde, hat der Hersteller die Möglichkeit, dem Endkunden Updates und Funktionserweiterungen direkt zu verkaufen. Der Hersteller kann diese kleineren Ergänzungen mit relativ geringem Verwaltungsaufwand bearbeiten, während ein Händler die zusätzlichen Module erst bestellen und dann weiterverkaufen müßte. Bei kleineren Ergänzungen würden die Kosten für die Abwicklung den benötigten Preis so stark anheben, daß ein Kunde dies nicht akzeptieren würde. Daher bietet sich hier ein automatisierter Update-Vorgang an, bei dem der Kunde aus der Applikation heraus die gewünschten Erweiterungen bestellt und diese beim Hersteller auch automatisiert in Rechnung gestellt werden.

5.2. Abläufe bei der Lizenzierung und deren Bewertung

Die einzelnen möglichen Abläufe beim Vertrieb von und Lizenzschutz für Software lassen sich nach den folgenden Kriterien bewerten:

5.2.1. Bedienerfreundlichkeit

Wie leicht kann die Applikation durch den Benutzer genutzt werden? Welche Fehlerquellen gibt es? Inwieweit verzögert der Lizenzschutz den normalen Betrieb der Applikation?

5.2.2. Sicherheit für Benutzer

Ist der Lizenzschutz so gestaltet, daß das System des Benutzers sicher gegen maliziöse Software bleibt? Ist sichergestellt, daß Daten des Benutzers, insbesondere personenbezogene Daten, nicht von Dritten eingesehen werden können?

5.2.3. Sicherheit für Hersteller

Wie leicht ist es möglich, die Software ohne gültige Lizenz zu nutzen? Wie wahrscheinlich ist es, daß dies auch tatsächlich geschieht? Gibt es eine Zielgruppe, die zu unlizenzierten Nutzung bereit wäre? Gibt es eine Gruppe, die bewußt mehr als die lizenzierten Module verwenden würde?

5.2.4. Aufwand für Implementierung des Lizenzschutzes für die Software

Wird der Lizenzschutz einfach auf bestehende Software aufgetragen oder müssen im Quellcode des Programms besondere Maßnahmen ergriffen werden? Genügt der Aufruf bestimmter APIs oder müssen auf Maschinencodeebene komplexe Verschleierungsmaßnahmen eingebaut werden?

5.2.5. Aufwand für Lizenzvertrieb und Aktivierung

Je nach Vorgehensweise muß für den Vertrieb der Lizenzen und den Aktivierungsvorgang eine bestimmte Infrastruktur zur Verfügung gestellt werden. Beim Vertrieb über Niederlassungen werden flächendeckend Verkäufer und Techniker benötigt, beim Verkauf über Händler müssen entsprechend Händler gefunden werden, bei der Vermietung von Software muß eine Vorgehensweise definiert sein, wie solche Verträge einfach verlängert werden können. Beim Direktvertrieb wird eine entsprechende Plattform benötigt; dazu kann ein Webserver mit der notwendigen Funktionalität gehören, aber auch ein sogenanntes Call-Center. Werden Lizenzinformationen über Datenträger verteilt, müssen Geräte vorhanden sein, mit denen sich diese in der benötigten Menge erstellen lassen. Wird die Funktionalität zum nachträglichen Erweitern von Lizenzen benötigt, so braucht man beim Hersteller eine Datenbank, welche die belieferten Kunden und deren bereits erworbene Lizenzen beinhaltet.

5.2.6. Grenzen der Einsetzbarkeit

Nicht jeder Ablauf und jeder Schutzmechanismus ist für jeden Kunden geeignet oder wird von jedem akzeptiert. Oft ist eine Internetverbindung auf das HTTP-Protokoll beschränkt, oder der Endkunde läßt überhaupt keinen automatisierten Datenaustausch mit dem Hersteller zu, da er um die Integrität seiner Systeme fürchtet. Entsprechendes gilt für Software, die ohne Verbindung zum Internet, auch auf Notebooks eingesetzt werden soll. Auch kann die Verwendung von Kopierschutzsteckern daran scheitern, daß das Zielsystem die notwendige Schnittstelle nicht enthält oder diese aus Sicherheitsgründen vom Administrator deaktiviert wurde.

5.2.7. Art der Lizenzbindung

Wie soll der Umfang bestimmt werden, in dem eine Lizenz gilt? Dabei gibt es insbesondere folgende Möglichkeiten wie die Bindung

- an ein bestimmtes einzelnes System wie einen konkreten PC,
- an eine Person, unabhängig davon, wo sie es einsetzt,
- an ein einzelnes System, das durch die Anwesenheit eines Kopierschutzsteckers oder Datenträgers bestimmt wird,
- an ein Netzwerk mit limitierter Anzahl von Benutzern,
- an eine gesamte Firma mit all ihren Mitarbeitern,
- an eine Kombination oder Summe der oben genannten Elemente, so beispielsweise eine Lizenz für zehn beliebige Nutzer, die zu einem bestimmten Zeitpunkt das Programm nutzen dürfen.

5.2.8. Weitere Einschränkungen

Werden weitere Einschränkungen benötigt? Eine solche Einschränkung kann die Aufteilung eines Systems in einzelne Module (vgl. Abschnitt 3) sein, die auch getrennt lizenziert werden können. Darüber hinaus gibt es noch andere Arten der Beschränkung:

- die maximale Anzahl verwalteter Personen oder maximale Datenbankgröße; damit wird es ermöglicht, mit einer Software sowohl Groß-, als auch Kleinkunden zu bedienen.
- eine zeitliche Beschränkung der Nutzungsdauer wie bei der Vermietung oder beim Leasing.
- die Notwendigkeit zum Vertrieb von Demoversionen; dabei sollte zusätzlich noch bekannt sein, ob die Demoversion zeitlich oder im Funktionsumfang beschränkt sein soll. Außerdem muß geprüft werden, ob eine funktional beschränkte Version schon den Programmcode für alle Funktionen enthalten soll oder nicht; ersteres ist notwendig, wenn die Vollversion über einen einfachen Zahlencode freischaltbar sein soll.

5.2.9. Veränderung von bestehenden Lizenzen

Wie wird vorgegangen, wenn sich der Umfang einer Lizenz verändern soll, wie beispielsweise bei der Freischaltung eines zusätzlichen Moduls? Dabei gibt es folgende Möglichkeiten des Vorgehens:

- Eine Veränderung der Lizenz ist nicht vorgesehen, wie beispielsweise bei einfachen, nicht-modularen Programmen.
- Bei einer Veränderung der Lizenz werden die Lizenzierungsinformationen als Ganzes ausgetauscht. Speichert der Hersteller alle Informationen seiner Kunden, so kann er diese bei sich anpassen und erneut an

den Kunden vertreiben. Hat der Hersteller diese Informationen nicht zur Verfügung, muß er diese zunächst beim Kunden anfordern.
- Die Lizenzierungsinformationen sollten beim Kunden vor Ort angepaßt werden können. Anhand der beim Hersteller bekannten Kundendaten werden Freischaltcodes generiert und an den Kunden übermittelt. Dieser trägt die Schlüssel an geeigneter Stelle in die Software ein und die Lizenzierungsinformationen werden so angepaßt, daß sie die veränderten Daten enthalten.

5.2.10. Notwendigkeit zur laufenden Kontrolle der Nutzung

Soll der Hersteller die Möglichkeit haben festzustellen, wie oft und in welchem Umfang der Kunde die Software nutzt? Soll er die Möglichkeit haben, festzustellen, wie verbreitet Demonstrationsversionen genutzt werden und ob ggf. ein Mißbrauch vorliegt?

5.2.11. Behandlung von Fremdsoftware von Drittherstellern

Wie soll Fremdsoftware behandelt werden, die zur der eigenen hinzugefügt werden kann? Dabei gibt es folgende Ansätze:
- Schnittstellen zu Fremdsoftware sind nicht vorgesehen.
- Fremdsoftware kann die Schnittstellen jederzeit nutzen.
- Die Schnittstelle zur Fremdsoftware muß vom Dritthersteller einmal für jedes seiner Produkte oder für alle Produkte lizenziert werden.
- Die Schnittstelle zur Fremdsoftware muß vom Dritthersteller für jede einzelne Lizenz, die er an seine Kunden verkauft, beim Hersteller lizenziert werden.
- Die Nutzung der Schnittstelle durch die Fremdsoftware muß vom Kunden lizenziert werden, der die Drittsoftware nutzen will.

5.2.12. Verwaltung von Lizenzen

Auf welche Art und Weise möchte der Hersteller die Lizenzen verwalten? Folgende Modelle sind möglich:
- Anonym, d. h. der Hersteller hat zwar einen Überblick über alle vergebenen Lizenzen und deren Funktionsumfang, kann diese aber nicht speziellen Kunden zuordnen.
- Namentlich, d. h. der Hersteller hat eine Übersicht über alle Endkunden und die von ihnen lizenzierte Konfiguration.
- Keine Verwaltung, d. h. der Hersteller vertreibt seine Lizenzierungsinformationen mit, ohne danach eine Rückmeldung zu erhalten, wer die

Lizenz erworben hat. Dies ist beispielsweise der Fall, wenn Standardsoftware auf Datenträgern vertrieben wird und der Lizenzschutz alleine mit dem Datenträger gekoppelt ist.

5.2.13. Integration von Nebenleistungen

Müssen Nebenleistungen wie Inbetriebnahme oder Wartungsverträge in den Vertriebs- und Lizenzierungsprozeß integriert werden? Ein Techniker, der beim Kunden Arbeiten durchführt, könnte anhand der Lizenzierungsinformationen feststellen, ob er die Tätigkeit gegenüber dem Kunden abrechnen muß und diesen gleich darüber informieren, so daß spätere Streitigkeit über die Vergütung dieser Arbeit minimiert werden können.

5.3. Aktivierung und Registrierung

Bei den Prozessen zur Bindung von Software wird im groben zwischen den Verfahren „Aktivierung“ und „Registrierung“ unterschieden. Die einzelnen Hersteller vermischen die beiden Begriffe jedoch oft.

5.3.1. Aktivierung

Damit eine Software genutzt werden kann, muß sie zunächst beim Hersteller freigeschaltet werden. Dabei wird sie in der Regel an ein bestimmtes System gebunden. Werden gewisse Systemkomponenten ausgetauscht oder die Software auf einen anderen Rechner übertragen, so muß der Aktivierungsprozeß erneut durchgeführt werden. Oft ist nur eine begrenzte Anzahl von Aktivierungen möglich. Abbildung 2 auf Seite 32 zeigt einen typischen Ablauf.

5.3.2. Registrierung

Bei der Registrierung gibt der Benutzer persönliche Daten (z. B. Name oder Firma) an, anhand derer er Lizenzierungsinformationen erhält, die er für den Betrieb des Programms benötigt. Bei gegen Entgelt vertriebener Software wird damit eine Zahlung verbunden (z. B. per Kreditkarte, durch Bankeinzug oder per Vorkasse). Oft enthalten die Arbeitsergebnisse des Programms diese Informationen im Klartext oder verschlüsselt. Außerdem ermöglichen es diese Informationen bei unlizenzierten Kopien, die Quelle für die Erstellung der Kopie zu identifizieren. Von der Aktivierung unterscheidet die Registrierung, daß danach dem Hersteller die Person des Endkunden bekannt ist.

Bei manchen Produkten wird die Software bei der Registrierung nicht verändert, der Hersteller sammelt nur die Daten für Werbezwecke. Oft ist eine

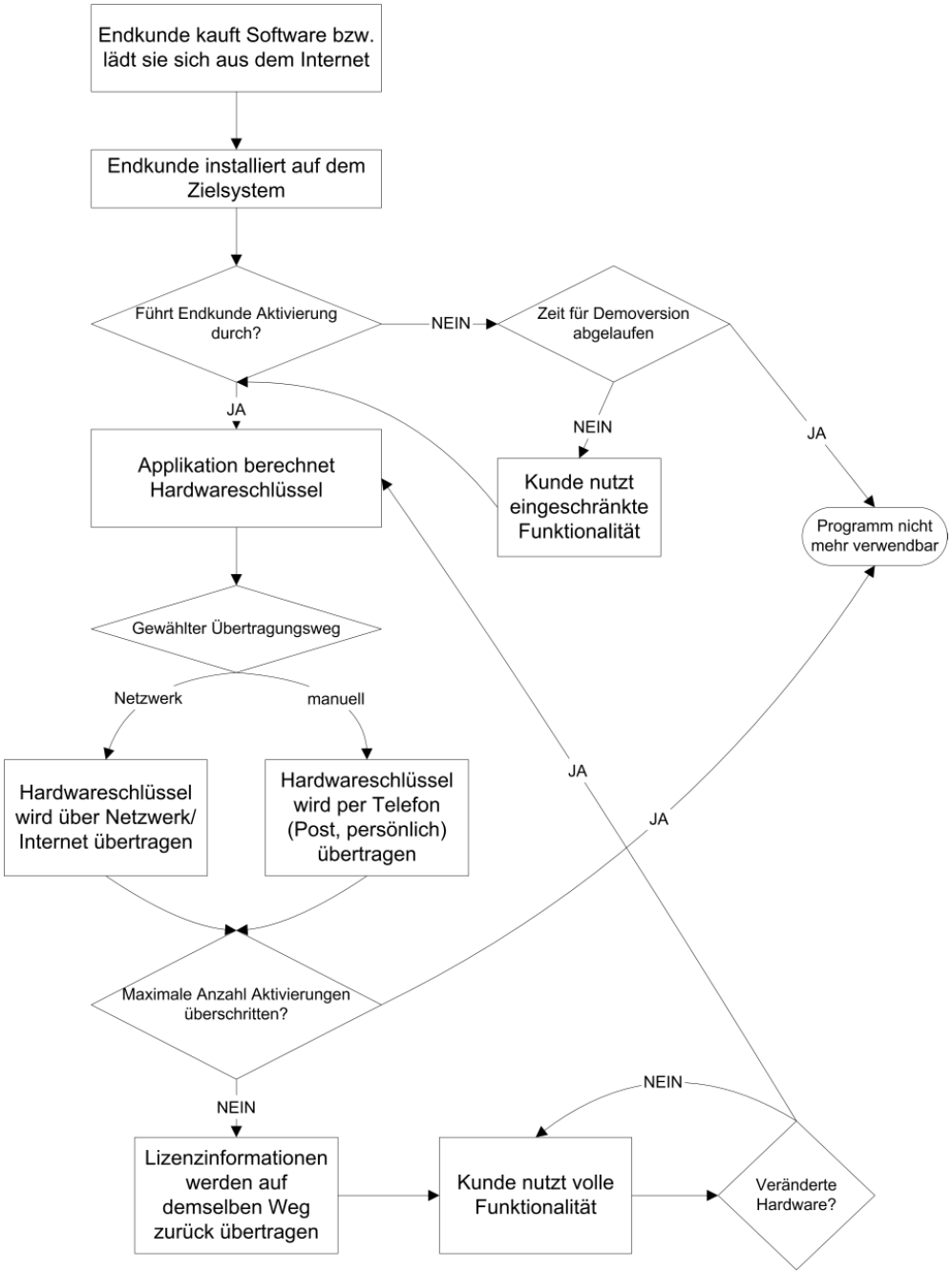

Abbildung 2: Typischer Ablauf eines Aktivierungsvorgangs

solche Registrierung auch Voraussetzung für die Nutzung unentgeltlich abgegebener Nutzungslizenzen, wobei vielfach von Schutzmaßnahmen abgesehen wird.

5.4. Abläufe bei der Lizenzierung

Bei den Prozessen im Zusammenhang mit der Lizenzvergabe lassen sich folgende drei Klassen unterscheiden:

- Erstmalige Freischaltung der Software bei einem Kunden,
- Erweiterung um zusätzliche Module bzw. Update auf eine neue Version,
- Erweiterung mit der Software eines Drittherstellers, einschließlich der Vergabe des Rechts zur Erstellung von Erweiterungen.

Dabei müssen die im Abschnitt 5.1 beschriebenen Vertriebswege berücksichtigt werden. Zunächst werden die verschiedenen Abläufe vorgestellt, danach werden weitere bei der Gestaltung der Abläufe zu berücksichtige Aspekte besprochen.

5.4.1. Erstmalige Freischaltung

Für die erstmalige Freischaltung einer Software gibt es verschiedene Vorgehensweisen:

5.4.1.1. Mitgelieferte Lizenzierungsinformationen

Der Hersteller liefert die notwendigen Lizenzierungsinformationen mit der Software aus. Dabei können diese auf dem Datenträger oder in den über Netzwerk übertragenen Dateien enthalten sein oder in Form von Lizenzschlüsseln auf Papier übertragen werden. In diesen Fällen ist eine Koppelung der Lizenz an einen bestimmten PC nicht möglich. Alternativ kann auch ein Kopierschutzstecker mitgeliefert werden.

Diese Vorgehensweise ist für den Direktvertrieb, den Vertrieb über Vertriebsniederlassungen, über Händler und über OEMs geeignet.

5.4.1.2. Lizenz wird nach Erhalt der Software durch den Endkunden vom Hersteller angefordert

Der Endkunde hat bereits die Software erhalten und diese installiert. Gegebenenfalls kann er diese auch schon als zeitbeschränkte Evaluierungsversion verwenden. Er fordert beim Hersteller die endgültigen Lizenzinformationen an. Dabei sind folgende Wege üblich:

- über ein Netzwerk; heute wird dazu i. d. R. das Internet verwendet. Alternativ könnte dem Endkunden auch die Möglichkeit gegeben werden, sich über eine Einwahlverbindung (z. B. über Modem oder ISDN) mit dem Hersteller zu verbinden;

- über Telefon; da die Übertragung längerer alphanumerischer Schlüssel sehr fehleranfällig ist, sollte die Schlüssellänge bei diesem Verfahren beschränkt werden;
- über Post; dabei können die Informationen entweder auf Papier oder mittels eines Datenträgers ausgetauscht werden;
- andere Wege, wie die persönliche Abholung der Lizenzierungsinformationen durch den Kunden, sind aufgrund der Entfernungen und des Aufwands wohl nur in Ausnahmefällen praktikabel.

Sofern die Lizenz an ein konkretes System gekoppelt werden soll, werden zunächst die Daten, welche die aktuelle Hardwarekonfiguration beschreiben, an den Hersteller übertragen, welcher dann die Lizenzinformationen zurückschickt. Sofern der Endkunde das Entgelt für die Lizenz noch nicht bezahlt hat (wie es z. B. bei Shareware üblich ist), werden noch Daten zur Person des Kunden und den Bezahlungsmodalitäten übertragen.

Diese Vorgehensweise ist für den Direktvertrieb gut geeignet. Sofern der Kunde bereits eine Lizenz erworben hat und nur noch die notwendigen Lizenzierungsinformationen benötigt, kann sie auch beim Vertrieb über Vertriebsniederlassungen, Händler oder OEMs eingesetzt werden.

5.4.1.3. Lizenz wird von der Vertriebsniederlassung, dem Händler oder dem OEM erstellt

Der Endkunde erhält seine Lizenzierungsinformationen von seiner Vertriebsniederlassung, seinem Händler oder dem OEM. Einer von diesen – im folgenden vereinfacht als Händler bezeichnet – verwaltet die Lizenzen seiner Kunden. Für den Hersteller muß sichergestellt sein, daß der Händler ihm die vereinbarten Entgelte für die verkauften Lizenzen auch tatsächlich bezahlt. Dazu gibt es verschiedene Geschäftsmodelle:

- Der Händler erhält für ein einmaliges oder jährliches Entgelt das Recht, beliebig viele Lizenzen an seine Kunden zu verkaufen. Vorteil dieser Lösung ist, daß der Hersteller nicht zu kontrollieren braucht, wieviele Lizenzen tatsächlich vertrieben wurden. Außerdem kann er seine Einnahmen kalkulieren, ohne Abweichungen befürchten zu müssen. Allerdings werden ihm gerade bei Produkten, die sich besser als erwartet verkaufen lassen, viele mögliche Einnahmen entgehen.
- Der Hersteller vertreibt die Lizenzen einzeln an seine Händler. Dabei ist der Verwaltungsaufwand pro Lizenz höher. Es gibt i. d. R. zweierlei Lizenztypen mit zugehörigen Lizenzierungsinformationen: die Lizenz vom Hersteller an den Händler, welche die Erstellung einer zugehörigen Lizenz für den Endkunden erlaubt. Die Lizenzinformation wird anonym erstellt und ggf. beim Hersteller auf den Kunden angepaßt. In diesem Falle muß mit technischen Mitteln sichergestellt werden, daß der Händler Endkundenlizenzen nur in dem Umfang verkauft, wie er die zugehö-

rigen Lizenzen auch vom Hersteller erworben hat. Die andere Möglichkeit ist, daß der Hersteller dem Händler bereits fertig an den Endkunden angepaßte Lizenzen liefert und der Händler sie unverändert an den Endkunden weiter gibt.
- Der Händler verkauft gewisse Mengen von Lizenzen. Die Vorgehensweise entspricht dem vorherigen Punkt, wobei der Verwaltungsaufwand minimiert wird.

5.4.1.4. Application Service Providing

Beim ASP wird die Applikation auf dem System des Herstellers ausgeführt. An die Stelle der Freischaltung der Software tritt die Bereitstellung der Zugangsdaten für den Kunden. Diese können dann diesem auf Papier, auf einem Datenträger oder einem Dongle übergeben werden. Die Software auf dem System des Herstellers muß dann nur noch zur Laufzeit prüfen, welche Module der Kunde lizenziert hat.

5.4.2. Erweiterung um zusätzliche Module bzw. Update

Für den Wechsel auf eine neuere Version bzw. eine Erweiterung um zusätzliche Module gibt es ebenfalls mehrere Ansätze:

5.4.2.1. Erweiterung durch den Hersteller

Folgende Wege bieten sich an, auf denen der Kunde Erweiterungen seiner Software bestellen kann:
- Der Kunde kauft die zusätzliche Lizenz im Versandhandel (z. B. über die Post, Fax, Telefon oder das Internet) und erhält von Hersteller einen Freischaltcode, mit dessen Hilfe er Module freischalten oder erweitern kann. Bei diesem Verfahren kann allerdings nur Funktionalität aktiviert werden, deren Code bereits beim Kunden vorhanden ist. Ein Versionsupdate ist auf diese Weise nicht möglich.
- Der Kunde kauft die Lizenz im Versandhandel, bekommt die Erweiterung als Datei(en) zugeschickt oder erhält die Möglichkeit, diese über das Internet herunterzuladen. Die Dateien können außer über ein Netzwerk auch auf einem Datenträger oder Dongle versandt werden. Auf diese Art und Weise können auch Module hinzugefügt werden, die beim Kunden noch nicht als Code vorhanden sind.
- Der Kunde bestellt die zusätzlichen Module über eine Online-Verbindung und diese werden im Rahmen dieser Verbindung übertragen. Die Rechnungsstellung durch den Hersteller erfolgt automatisch. Der Hersteller muß an dieser Stelle die Konsistenz der neuen Konfiguration prüfen. Dies kann automatisiert oder manuell durch einen Angestellten des Herstellers geschehen. Optional kann der Hersteller auch noch andere

Gegebenheiten prüfen, so beispielsweise eine Bonitätsprüfung vornehmen.
- Ein Techniker des Hersteller nimmt die notwendigen Anpassungen beim Kunden vor. Dazu kann er entweder die Lizenzierungsinformationen austauschen oder mittels entsprechender Freischaltcodes auf dem Zielsystem anpassen.

5.4.2.2. Erweiterung durch Vertriebsniederlassung, Händler oder OEM

Ein Händler (hier wieder stellvertretend auch für Vertriebsniederlassungen oder OEMs) kann in seiner Beziehung zum Kunden auf dieselbe Art und Weise vorgehen wie der Hersteller. Der Hersteller muß allerdings wie schon bei der erstmaligen Freischaltung (vgl. 5.4.1) sicherstellen, daß die an den Endkunden vertriebenen Lizenzen entsprechend vergütet werden. Der Hersteller kann auch hier Mengen von Lizenzen vertreiben. Diese Lizenzen erlauben es, die Lizenzierungsinformationen des Endkunden entsprechend zu modifizieren; dies kann entweder beim Händler geschehen, der eine Kopie der Lizenzierungsinformationen hat, oder beim Endkunden, wo der Händler durch Eingabe eines Freischaltcodes die Lizenz erweitert.

5.4.2.3. Application Service Providing

Wünscht der Kunde bei einer mittels ASP betriebenen Applikation eine Erweiterung der Funktionalität, kann er diese entweder über den Versandhandel beim Hersteller bestellen oder die Erweiterung direkt im System auslösen. Bei letzterem ist auf eine entsprechende Vergabe der Benutzerrechte zu achten, damit nicht jeder Mitarbeiter, der beispielsweise mittels eines Zeiterfassungssystems seinen Urlaub eintragen kann, Systemerweiterung bestellen kann. Die Prüfung des Erweiterungswunsches entspricht derjenigen bei Bestellung über eine Online-Verbindung, wie sie oben beschrieben wurde.

5.4.3. Vertrieb der Erweiterungen eines Dritthersteller an den Endkunden

Bei den Lizenzen um Erweiterungen durch Dritthersteller sind zwei Aspekte zu unterscheiden:

a) Die Lizenz des Dritthersteller, Erweiterungen zu erstellen und zu vertreiben.
b) Die Lizenz, die der Dritthersteller an den Endkunden vergibt, damit letzterer die Erweiterungen nutzen darf.

Für beide Bereiche müssen Abläufe definiert werden:

5.4.3.1. Erweiterungslizenz für Dritthersteller

Der Hersteller kann dem Dritthersteller wie bereits in 4.2.2 beschrieben, in unterschiedlichem Umfang Lizenzen für Erweiterungen gewähren. Gewährt er einem Dritthersteller das Recht, uneingeschränkt Erweiterungen zu erstellen, so genügt es dem Dritthersteller, einmalig eine Lizenzierungsinformation auszuliefern, die jener dann mit seiner Erweiterung vertreibt. Wird die Erweiterungslizenz jedoch nur in Zusammenhang mit einem Endkunden erteilt, so kann die zugehörige Lizenzierungsinformation entweder vom Hersteller wie eine eigene Erweiterung auf den entsprechenden Vertriebswegen vertrieben werden oder der Dritthersteller wird wie ein Händler behandelt, der die entsprechende Lizenz an seinen Kunden weitervertreibt. Für beide Fälle sind mögliche Vorgehensweisen bereits im Abschnitt 5.4.2 beschrieben.

5.4.3.2. Lizenz des Drittherstellers an den Endkunden

Der Dritthersteller kann die Lizenz für seine Erweiterungen auf dieselbe Art und Weise vertreiben, wie dies auch der Hersteller mit seinen eigenen Produkten macht; dies ist bereits in vorhergehenden Abschnitten beschrieben. Alternativ kann der Hersteller diese Lizenzen mitverwalten. Dabei muß die Lizenzvergabe des Herstellers durch den Drittherstellers entsprechend kontrolliert werden. Die Vorgehenweise kann analog zur Kontrolle der Händler durch den Hersteller gestaltet werden.

5.4.4. Einschränkung und Sicherheit

5.4.4.1. Recht zur Bestellung

Soll es dem Endkunden ermöglicht werden, aus dem Programm heraus neue Module zu bestellen, muß festgelegt werden, wer diesen Vorgang durchführen darf. Dazu gibt es folgende Ansätze:

- Jeder Benutzer kann Modulerweiterungen bestellen. Diese Vorgehensweise eignet sich für Programme, die i. allg. nur von einer Person genutzt werden.
- Sofern das Softwaresystem über eine eigene Benutzerverwaltung verfügt, kann die Gruppe der Bestellberechtigten darüber eingestellt werden.
- Zur Bestellung wird ein spezielles Paßwort benötigt oder der Berechtigte wird über seinen Besitz (z. B. RFID-Ausweis oder Dongle) authentifiziert. Dies wird vom Hersteller an den Endkunden mit ausgeliefert, der dann über den weiteren Zugriff entscheiden kann.
- Der berechtige Besitzer wird über seine biometrischen Merkmale authentifiziert. Dies setzt voraus, daß diese Daten entweder vorab dem Hersteller zur Verfügung gestellt werden oder daß sie bei der Installation der Software erfaßt werden.

Entsprechend sollte die Möglichkeit eines Bestellvorgangs durch Servicetechniker des Herstellers überdacht werden. Da ein solcher Vorgang mit Kosten für den Endkunde verbunden ist und ohne Zustimmung des Kunden kein Kaufvertrag zustandekommen kann, sollten die Rechte des Technikers auf die temporäre Freigabe von Lizenzen beschränkt werden.

Bei bestimmten Endkunden kann es auch Sinn machen, die Erstellung von temporären Lizenzen vor Ort komplett zu deaktivieren, wenn der Hersteller Mißbrauch befürchtet.

5.4.4.2. Datenschutz, Datensicherheit und Datenintegrität

Bei der Gestaltung der Prozesse ist zu berücksichtigen, daß Daten des Kunden nicht von Dritten mitgelesen werden können. Außerdem sollte eine Manipulation der Datenübertragung durch andere verhindert werden.

Damit ein Endkunde einen Lizenzierungsprozeß akzeptiert, kommt es nicht nur auf die entsprechende technische Gestaltung an, sondern auch darauf, wie der Benutzer diese subjektiv wahrnimmt (vgl. die Seminararbeit [Dre01]). Aus diesem Grund sollten die Abläufe derart transparent gestaltet werden, daß der Benutzer den Eindruck hat, daß er alle Vorgänge unter Kontrolle hat. Außerdem sind Transaktionen, wie Bestellvorgänge, so zu gestalten, daß sie nur als Ganzes oder gar nicht durchgeführt werden und daß der Benutzer erkennen kann, ob der Vorgang erfolgreich war.

5.5. Verhinderung von Behinderungen durch irrtümlicherweise nicht freigeschaltete Funktionen

Die Abläufe sollten im Interesse der Endkunden so gestaltet werden, daß fehlerhafte Konfigurationen diesen möglichst wenig behindern. Eine Möglichkeit dazu ist es, die Prozesse so zu gestalten, daß eine Konfiguration mit wenig Aufwand in kurzer Zeit angepaßt werden kann. Dazu muß das dazu notwendige Personal entweder dauernd verfügbar sein (und nicht nur innerhalb enger Geschäftszeiten) oder die Abläufe sind derart automatisiert, daß der Kunde sie rund um die Uhr nutzen kann.

Ist es aus technischen oder logistischen Gründen nicht möglich, Lizenzen kurzfristig anzupassen, kann es hilfreich sein, Leistungsmerkmale übergangsweise mittels vereinfachter Verfahren einmalig zeitbeschränkt freizuschalten. In dieser Zeit können dann weitere Prüfungen oder die Bestellabwicklung vorgenommen werden.

Außerdem sollte darauf geachtet werden, daß die Abgrenzung der einzelnen Module so gewählt wird, daß eine zusätzliche vom Kunden gewünschte Funktionalität möglichst eng abgegrenzt ist. Wird das die Funktionalität enthaltende Modul zu groß gestaltet, kann der Preis einschließlich des Überhangs so groß werden, daß der Kunde diesen nicht mehr akzeptiert. Sicher

kann man in einem solchen Fall dem Kunden das entsprechende Modul zu einem geringerem Preis lizenzieren, jedoch schließt dies aus, daß die im Überhang enthaltene Funktion dann nochmals Ertrag bringen kann, wenn sie der Kunde später benötigen wird.

5.6. Beispielabläufe

Im Anhang im Abschnitt B werden exemplarisch verschiedene gängige Abläufe bei der Lizenzierung dargestellt.

6. Lösungen für den Schutz vor unlizenzierter Nutzung

Auf dem Markt werden verschiedene Produkte angeboten, die nach Einbindung in die eigene Software letztere vor unlizenzierter Nutzung schützen sollen. Diese Produkte enthalten auch in unterschiedlichem Umfang Funktionalität, welche die Vorgänge um die Übertragung der Lizenzierungsinformationen an den Endkunden abdeckt. Außerdem werden in der Literatur vorgestellten Ansätze für den Kopierschutz beschrieben.

Im Abschnitt 6.2 werden zunächst einmal die verschiedenen Ansätze beschrieben und bewertet. Es wird dann in Tabellenform ein Überblick darüber gegeben, welche Techniken in welchen der untersuchten Produkten verwendet werden.

Basierend auf diesen beschrieben Lösungsansätzen werden im Abschnitt 6.3 Vorschläge für Lizenzschutzmaßnahmen gemacht und mit eigenen Ideen sowie Vorschlägen aus dem akademischen Umfeld ergänzt.

Zunächst einmal wird ein Überblick darüber gegeben, was die unlizenzierte Nutzung, vor der geschützt werden soll, beinhaltet.

6.1. Übersicht

Die folgenden Absätze beschreiben, vor welchen Arten von unberechtigter Verwendung Software geschützt werden sollte:

6.1.1. Verwendung ohne Lizenz

Der Benutzer nutzt die Software, ohne eine gültige Lizenz für die Nutzung zu besitzen. Um dies in gewissem Umfang zu verhindern, können die Schutzmechanismen angewandt werden, die in den folgenden Abschnitten vorgestellt werden.

6.1.2. Schutz vor Verwendung über die vorhandene Lizenz hinaus

Der Benutzer besitzt eine Lizenz für die Nutzung der Software, diese ist jedoch in Bezug auf die Funktion eingeschränkt. So ist beispielsweise die Nutzung bestimmter Teilfunktionen ausgeschlossen. Der Lizenzschutz sollte gewährleisten, daß die nichtlizenzierten Module nicht verwendet werden können.

6.1.3. Schutz vor Umgehung der Lizenzschutzmaßnahmen

Neben dem Schutz vor unlizenzierter Nutzung werden Techniken benötigt, die eine Umgehung dieses Schutzes verhindern. Dies wird als “Anti-Spoofing” bezeichnet. Es nutzt dem Hersteller einer Software wenig, wenn diese einen Schutz erhält, der die Möglichkeit, diese auszuführen, auf ein bestimmtes System beschränkt, ein unberechtigter Nutzer diesen Schutz jedoch schon mit einfachsten Mitteln umgehen kann.

6.2. Bestehende Lösungen

Im Rahmen dieser Arbeit wurden 44 Produkte verschiedener Hersteller anhand deren Datenblätter und Produktbeschreibungen untersucht; am Ende dieses Abschnitts wird in Form von Tabellen ein Überblick über deren Leistungen gegeben.

6.2.1. Integration in eigene Software

Um eigene Software mittels der in dieser Arbeit beschriebenen Produkte vor unlizenzierter Nutzung schützen zu können, muß die Schutzsoftware mit der eigenen gekoppelt werden. Im folgenden wird dargestellt, welche Vorgehensweisen gebräuchlich sind.

6.2.1.1. Funktionalität in dynamischer Bibliothek

Eine vom Hersteller der Schutzsoftware zur Verfügung gestellte Bibliothek enthält die notwendigen Funktionen und wird vom Entwickler innerhalb seiner Software aufgerufen. Alternativ können die Funktionen in einer OCX-(ActiveX-)Komponente aufgerufen werden.

Der Aufwand für den Entwickler besteht in

- der Einarbeitung in die Lizenzierungs-API,
- der Anpassung des Quellcodes, so daß die Lizenzierungsfunktionen aufgerufen werden und
- der Anpassung des Installationsprogramms, so daß die Bibliothek mit installiert wird.

Vorteile dieser Lösung sind:

- ein geringer Aufwand für Integration in eigene Software,
- die Lizenzierungs-API kann leicht gegen neuere Version ausgetauscht werden, wenn die Schnittstelle erhalten bleibt,
- angepaßte Lösungen, z. B. mit eigenen Lizenzierungsdialogen, werden erleichtert.

Nachteile sind dagegen:

- die Lizenzierungs-API kann leicht umgangen werden, indem die Bibliothek durch eine andere mit derselben Schnittstelle ersetzt wird; dies wird dadurch erleichtert, daß die API ja öffentlich[2] ist, da sie zur Entwicklung benötigt wird,
- für unterschiedliche Systeme werden unterschiedliche Dateien benötigt.

6.2.1.2. Funktionalität in statischer Bibliothek

Der Aufwand liegt in

- der Einarbeitung in die Lizenzierungs-API,
- der Anpassung des Quellcodes, so daß die Lizenzierungsfunktionen aufgerufen werden.

Vorteile dieser Vorgehensweise sind:

- ein geringer Aufwand für Integration in eigene Software,
- die Lizenzierungs-API kann leicht gegen neuere Version ausgetauscht werden, wenn die Schnittstelle erhalten bleibt, allerdings nicht nach erfolgter Generierung der Binärdateien,
- angepaßte Lösungen, z. B. mit eigenen Lizenzierungsdialogen, werden erleichtert.

Dagegen stehen folgende Nachteile:

- Die Lizenzierungs-API kann umgangen werden, da die Einsprungpunkte der Funktionen bei bekannter API leicht im Binärcode zu lokalisieren sind; dann genügt es, den Code dieser Funktionen zu manipulieren,
- für unterschiedliche Systeme werden unterschiedliche Dateien benötigt; die statischen Bibliotheken funktionieren u. U. nicht mit jedem Linker.

6.2.1.3. Verschlüsselung der Binärdatei

Bei dieser Technologie liegt der Aufwand

- im Aufruf des Verschlüsselungsprogramms, das eine Binärdatei in eine verschlüsselte Datei umwandelt.

Vorteile dieser Lösung sind:

- ein sehr geringer Aufwand für Integration in eigene Software,
- die Lizenzierungsfunktionen können einfach gegen ein anderes Produkt ausgetauscht werden,
- in der Entwicklung kann das Originalprogramm unverschlüsselt getestet und gedebuggt werden, die Verschlüsselung wird erst vor dem Vertrieb hinzugefügt.

Nachteile sind dagegen:

- wenn die Verschlüsselung einmal „geknackt“ und ein Dekompressionsprogramm erstellt wurde, dann kann jedes mit dieser Schutzsoftware versehene Programm unlizenziert genutzt werden; es gibt inzwischen

[2] zumindest kennt sie jeder andere, der dieselbe Schutzsoftware erworben hat

Freewareprogramme wie PEiD [PEi], welche die gängigsten Pack- und Verschlüsselungsprogramme erkennen können,
- für angepaßte Lösungen – wie komplexe Verwaltung von Modulen – wird dennoch ein Zugriff auf die oft vorhandene Lizenzierungs-API aus der Applikation heraus benötigt, um festzustellen, welche Module freigeschaltet sind; damit erhöht sich der Aufwand für die Integration wieder.

6.2.1.4. Funktionalität wird in den Quellcode integriert

Das zu ist notwendig:
- die Integration der Verschlüsselungsmakros im Quellcode, zur Sicherheit an mehreren Stellen.

Ein Vorteil dieser Lösung ist:
- der Schutz kann nur dadurch umgangen werden, daß im Binärcode alle Schutzfunktionalität gefunden und entfernt wird.

Von Nachteil ist dagegen:
- der hoher Aufwand für Integration in eigene Software,
- dies wird von Fremdherstellern nur selten angeboten, da deren Kunden ja das ganze Know-How erhalten würden.

6.2.2. Klassifizierung der Arten von Lizenzprüfungen

In diesem Abschnitt werden die verschiedenen in den Produkten implementierten Vorgehensweisen bei der Lizenzprüfung und -verwaltung beschrieben.

6.2.2.1. Systeme mit Sicherungsmaßnahmen gegen Umgehung des Kopierschutzes

[Čer02] und [Fel99] schlagen verschiedene Maßnahmen vor, welche ein “Crakken” der Software erschweren sollen. Dies ist auch unter dem Begriff “Anti-Spoofing” bekannt. Derartige Maßnahmen sind sinnvoll, um eine Umgehung der in den folgenden Abschnitten vorgestellten Techniken zu erschweren und werden auch in den vorgestellten Produkten verwendet. Viele dieser Maßnahmen basieren jedoch alleine darauf, den Angreifer zu täuschen und aufzuhalten:
- **Ablenkung und Täuschung**
  - Namen von Variablen und Funktionen nicht sprechend benennen.
  - Zur Ablenkung zusätzliche Variablen und Funktionen mit prägnanten Namen einführen, die jedoch nicht die angenommene Wirkung haben.
  - Sprechende Dateinamen für Lizenzinformationen vermeiden.
  - Fenster, die an die unregistrierte Nutzung erinnern, vermeiden. Diese Fenster erleichtern die Lokalisierung des Prüfcodes.

  - Speichern Sie Lizenzinformationen an ungewöhnlichen Orten, wie beispielsweise Datenbankfeldern.
  - Bringen Sie Lizenzierungsinformationen an ungewöhnlichen Orten unter, z. B. in Hilfs-DLLs im Systemverzeichnis. Verschlüsseln Sie diese Informationen.
  - Verteilen Sie Seriennummern an mehrere Stellen oder speichern Sie solche Codes redundant.
  - Sorgen Sie dafür, daß Informationstexte für die Evaluierungsversionen nicht im Klartext im Code stehen.
  - Experimentieren Sie mit überflüssigen Funktionen und Zeichenketten, um einen Angreifer zu verwirren.
  - Programmieren Sie bestimmte Abschnitte zur Verwirrung bewußt unsauber (z. B. „Spaghetti-Code“).
  - Geben Sie hartcodierten Schlüsseln Namen, die entweder wie Programmcode aussehen oder wie geläufige, reservierte Funktionsnamen aus der API aussehen.
  - Verstecken Sie Tabellen mit Schlüsseln mitten im Quellcode.
  - Lenken Sie “Cracker” in eine falsche Richtung. Fügen Sie beispielsweise viele NOOP-Befehle ein, so daß selbstmodifizierender Code vermutet und niemals gefunden wird.
  - Sorgen Sie dafür, daß Registrierungsnummern und Codes nicht im Klartext im Speicher zu finden sind.
- **Sonstige Behinderung der Analyse und Umgehung der Kopierschutzes**
  - Fügen Sie nach einer Codeeingabe eine kurze Wartezeit ein, um “Brute Force”-Angriffe zu erschweren.
  - Reagieren Sie nicht sofort auf falsche Codeeingaben und unberechtigte Nutzung, sondern warten Sie eine längere Zeit (bis zu mehreren Tagen). Der “Cracker” meint, bereits den Lizenzschutz umgangen zu haben und kann dann nochmal von vorne beginnen.
  - Sorgen Sie dafür, daß sich Ihr Code bei jedem Aufruf so modifiziert, daß jeweils andere Prüfroutinen aufgerufen werden. Auf diese Weise führt eine einmalige Manipulation des Codes durch den “Cracker” nicht zum Ziel. Sie können auch den Code zur Laufzeit so verändern, daß beim Programmstart ein ganz anderer Code sichtbar ist.
  - Verwenden Sie keine Funktionsaufrufe zur Validierung der Lizenz. Wenn Sie den Code an mehreren Stellen prüfen und jedes Mal “inline” aufrufen, genügt es nicht, diese Funktion einmalig zu manipulieren (z. B. durch Füllen mit NOOP-Befehlen).
  - Vermeiden Sie deaktivierte Menüs oder Schaltflächen. Sie lassen sich leicht zur Laufzeit aktivieren. Die Funktion muß immer auch im eigentlichen Code überprüft werden.

  - Veröffentlichen Sie oft neue Versionen. Mit jeder neuen Version funktionieren bestehende Binärpatches nicht mehr.
  - Verwenden Sie möglichst große Lizenzierungsschlüssel.
  - Ersetzen Sie gewisse Funktionen, wenn Sie von den Beta-Versionen zur freigegebenen weitergehen.
  - Verwenden Sie Prüfsummen über Binärdateien und Bibliotheken.
  - Vertreiben Sie CDs nur in kleinen Chargen mit leicht veränderten Kopierschutzmechanismen bei ansonsten funktionsgleicher Software. Dann ist, wenn es jemand gelingt, den Kopierschutz zu umgehen, nur diese eine Charge betroffen.
  - Bei Programmierung in Assembler: Verwenden Sie Sprünge in Abhängigkeit des Carry-Flags. Der Wert dieses Flags ist bei der Disassemblierung schwer zu bestimmen. Plazieren Sie im allgemeinen in kritischen Bereichen möglichst viele bedingte Sprünge. Diese sind, wenn sie massiv auftreten, schwer nachzuvollziehen.
  - Erkennen Sie Kernel-Mode-Debugger wie SoftICE frühzeitig und „schießen“ Sie dann das System über eine ungültige Operation ab.
  - Vermeiden Sie Ausgaben über unlizenzierte Nutzung; sie erleichtern die Lokalisierung der Prüfroutinen.
  - Führen Sie Lizenzprüfungen nicht immer beim Programmstart durch, sondern erst später; führen Sie die Prüfungen an mehreren verschiedenen Stellen und nur mit einer gewissen Wahrscheinlichkeit durch. Dies erschwert es, alle Prüfungen zu finden.
  - Verändern Sie den Code während der Laufzeit; dies verwirrt jeden, der sich an einer Analyse versucht.
  - Sorgen Sie dafür, daß eine Testroutine sehr lange läuft (mehrere Sekunden) und viel unterschiedlichen Code enthält. Ein Debuggen und Analysieren des Codes wird damit sehr aufwendig.
  - Wenn Ihr Produkt „gecrackt“ wurde, geben Sie eine neue Version frei. Häufige Updates bedeuten viel Aufwand für das Umgehen der Schutzmaßnahmen.
- **Sonstiges**
  - Verwendung asymmetrischer Verschlüsselung. Verwenden Sie nur „richtige Verschlüsselung“ und keine Primitivcodes wie simple Anwendung der Antivalenz (XOR), Cäsar-Chiffre oder ROT-13.
  - Vertrauen Sie der Systemzeit nicht. Prüfen Sie darüber hinaus die letzten Änderungen regelmäßig beschriebener Systemdateien (z. B. Windows-Registrierung) oder einen Zeitserver im Netzwerk, um ein Verstellen der Uhrzeit zur Verlängerung von Evaluierungsversionen zu erkennen. Verzichten Sie am besten ganz auf solche Zeitprüfungen, wenn es keinen sachliche Grund wie ein Lizenzierungsmodell mit Vermietung von Software oder Bedürfnissen des Marketings nach Demonstrationsversionen gibt.

- Schätzen Sie das Interesse an Ihrer Software und deren illegaler Nutzung richtig ein. Bei vielen Vertriebskanälen spielen unlizenzierte Kopien keine große Rolle.
- Verändern Sie die vom Benutzer geschaffenen Daten, so daß beispielsweise beim Drucken fehlerhafte Resultate entstehen.
- Stellen Sie Fallen: Lassen Sie unlizenzierte Versionen sich nach einer gewissen Zeit mit einem speziellen Fehlercode beenden.
- Vertrauen Sie Packprogrammen für ausführbare Programme nicht. Für die meisten gängigen Programme sind inzwischen Entpackprogramm verfügbar.
- Je länger Lizenzierungsinformationen und Codes sind, desto besser.
- Die Änderungzeitpunkte von Systemdateien wie der Windows-Registry geben einen Hinweis auf Manipulationen an der Systemzeit.
- Liefern Sie beschränkte Versionen am besten so aus, daß sie den nicht-lizenzierten Code gar nicht enthalten.
- Nutzen Sie Online-Registrierung.
- Testen Sie Ihren Schutz unter den verschiedenen Betriebssystemen.
- Beobachten Sie die im Internet kursierenden „Crack-Programme“; wer ihre Funktionsweise versteht, kann vieles am eigenen Schutz verbessern. Außerdem kann auf neue Entwicklungen in diesen Kreisen nur dann reagiert werden, wenn man auch Kenntnis davon erlangt.
- Je ungewöhnlicher und seltener ein Kopierschutz ist, desto länger brauchen „Cracker“, um sich darauf einzustellen.

6.2.2.2. Systeme mit Kopierschutz durch Koppelung an ein konkretes System oder anderes Medium

Bei diesen Systemen soll die unrechtmäßige Nutzung dadurch verhindert werden, daß sie Lizenzinformationen enthalten, die eine Ausführung des Programms nur in Kombination mit bestimmter festgelegter Hardware ermöglichen.

6.2.2.2.1. Koppelung an PC-Hardware

Die Lizenzierungsinformation enthält Informationen über die Hardware des Systems, auf der sie alleine ausführbar sein soll. Während der Ausführung des Programmes werden diese dann mit dem aktuellen System verglichen und bei Differenzen die Ausführung des Programms eingeschränkt oder abgebrochen. Oft werden diese Parameter nicht im Klartext gespeichert, sondern sie dienen als Basis für eine Prüfsumme, einen Hash-Wert oder einen kryptographischen Schlüssel.

Der Hersteller benötigt in diesem Fall zur Erstellung der Lizenzinformation die entsprechenden Daten des Zielsystems.

Die gebräuchlichsten Hardware-Parameter sind:

- Die **CPU-ID**; dies ist eine eindeutige Seriennummer des Prozessors, die bei neueren Modelle per Software abgefragt werden kann.
- Die **Volume-ID** eines Datenträgers, d. h. die Bezeichnung, die der Anwender an einen Datenträger oder eine Partition vergeben kann. Nachteil ist, daß dieser Wert problemlos angepaßt werden kann, um dann eine nicht-lizenzierte Software ausführen zu können.
- Die **Hersteller-ID** eines Datenträgers; dies ist eine eindeutige, vom Hersteller vergebene Nummer, die der Benutzer im allgemeinen nicht ändern kann. Eine solche gibt es oft bei neueren Festplatten.
- Das **BIOS** kann eine Funktion zur Abfrage der Seriennummer des Motherboards zur Verfügung stellen. Außerdem können die im CMOS gespeicherten Daten des System Setups, die Hardwarekonfigurationseinstellungen enthalten, geprüft werden.
- Die **MAC-Adresse** der Netzwerkkarte[3] ist eine Kennung, die zur eindeutigen Kennung eines Geräts in einem Netzwerk dient. Sie wird vom Hersteller des Chips bei seiner Herstellung eindeutig vergeben. Oft kann sie aber auch über ein Dienstprogramm nachträglich angepaßt werden. Eine Prüfung der MAC-Adresse verhindert aber zumindest die gleichzeitige mehrfache Nutzung der Software auf Geräten im selben Netzwerk.

Nachteil der Bindung an die PC-Hardware ist, daß sobald sich eine der geprüften Komponenten z. B. wegen eines Defektes oder Ausbaus ändert, die Lizenzinformationen vom Hersteller angepaßt werden müssen. Außerdem läßt sich kein Backup-System einrichten, das im Falle eines Defektes die Aufgaben des ursprünglichen übernimmt, ohne daß dafür eine zusätzliche Lizenzierungsinformation benötigt wird. Manche Hersteller bieten daher auch tolerantere Systeme an, die Abweichungen bis zu einem bestimmten Grad erlauben. Sie erlauben bespielweise den Tausch von einer bestimmten Anzahl von Komponenten.

6.2.2.2.2. Koppelung an Netzwerkparameter

Neben Hardwareparametern können auch Netzwerkeinstellungen Bestandteil der Lizenzierungsinformation sein. So müssen beispielsweise Netzwerkadressen im selben Netzwerk eindeutig sein; dadurch kann eine gleichzeitige Nutzung auf Geräten im selben Netzwerk verhindert werden.

Üblich sind hierbei:

[3] Sicher könnte man diesen Wert auch dem Abschnitt 6.2.2.2.2 zuordnen, jedoch werde ich mich dort auf die üblicherweise vom Benutzer eingestellten Werte konzentrieren.

- Die **IP-Adresse** oder eine andere Adresse, die einen Rechner im Netzwerk eindeutig identifiziert.
- Der **Hostname**, d. h. der Name unter dem der Rechner im Netzwerk erreichbar ist.
- Der **NetBIOS-Name**, der zur Kommunikation unter Windows-Rechnern verwendet wird.

Diese Werte lassen sich einfach manipulieren, jedoch lassen sich zwei Systeme nicht gleichzeitig mit den selben Einstellungen im selben Netzwerk betreiben.

6.2.2.2.3. Lizenzschutz durch Koppelung an Datenträger

Die Software wird so abgesichert, daß sie nur ausgeführt werden kann, wenn ein bestimmter Datenträger vorhanden ist. Dies kann die Diskette oder CD-ROM sein, von der das Produkt installiert wurde oder auch ein extra zu diesem Zweck mitgelieferter Memory-Stick. Werden nur die auf dem Datenträger abgespeicherten Daten geprüft, so funktioniert das Programm in diesem Fall auch mit einer bitweisen Kopie des Datenträgers; dies ist auch dann der Fall, wenn Lizenzierungsinformationen außerhalb der Dateien im Dateisystem versteckt sind (z. B. Prüfung auf Vorhandensein gelöschter Dateien und deren Inhalt).

Pavol Červeň beschreibt in [Čer02, Kapitel 4] verschiedene Techniken zum Kopierschutz mittels CD-ROMs. Die einfachste (und am leichtesten zu umgehende) Technik ist dabei eine Windows-API-Funktion, die prüft, ob es sich bei einem Laufwerk um ein CD- oder DVD-Laufwerk handelt (vgl. [Micf] Funktion *GetDriveTypeA*); ein eigener manipulierter Treiber genügt, um diesen Schutz zu umgehen. Weitere CD-Kopierschutz-Techniken sind:

- Gezielt erstellte fehlerhafte Blöcke auf der CD, die sich mit handelsüblichen Brennern nicht auf die Kopie übernehmen lassen.
- Die CD enthält verschlüsselt die Information über den Winkel zwischen den ersten und letzten Datenblöcken. Bei Kopien kann sich dieser vom dem auf dem Originaldatenträger unterscheiden.
- Die CD ist größer als 74 Minuten Laufzeit. Dies wurde in den Anfangszeiten der beschreibbaren CDs als Kopierschutz eingesetzt, als bei diesen noch keine größeren Rohlinge verfügbar waren. Heute spielen diese Beschränkungen keine Rolle mehr, da moderne Brenner mit ihrer Software und geeigneten Rohlinge problemlos solche Kopieren erstellen können.
- Die CD enthält ein defektes Inhaltverzeichnis, um das Erstellen von Kopien zu verhindern. Kann mit modernen Kopierprogrammen leicht umgangen werden.
- Das Inhaltsverzeichnis der CD wurde so manipuliert, daß beim Kopieren auf Festplatte Dateien entstanden, die größer als die Kapazität der CD

waren, oft sogar mehr als 2 GB groß. Alte Brennprogramme konnten dann keine neuen CDs daraus brennen.

Die Umgehung dieser Maßnahmen erfolgt im allgemeinen nicht, indem man den Datenträger originalgetreu nachbildet, sondern indem man das Binärprogramm so manipuliert, daß die Prüfungen umgangen werden.

Nachteil dieser Lösungen ist, daß ein berechtigter Nutzer keine funktionierenden Sicherungskopien der Software erstellen kann.

6.2.2.2.4. Lizenzschutz durch Koppelung an Dongles

Zur Lizenzabsicherung wird eine spezielle Hardware erstellt, die als Kopierschutzstecker oder “Dongle” bezeichnet wird. Der Zugriff erfolgt über eine spezielle API und ggf. noch über einen Treiber.

Umgangen wird eine solche Schutzmaßnahme, indem

a) die Hardware nachgebildet wird *oder*
b) das Protokoll nachgebildet wird *oder*
c) die Prüffunktion innerhalb der Software manipuliert wird.

Ein Nachteil von Kopierschutzsteckern ist, daß bei Produkten im untersten Preissegment die Kosten für diese sowie deren teureren Vertrieb die Größenordnung des kalkulierten Produktpreises erreichen können. Außerdem können viele diese Stecker nachgebaut oder durch Software simuliert werden.

6.2.2.2.5. Arten der Übertragung und Speicherung der Lizenzierungsinformationen

Lizenzierungsinformationen werden entweder in einfachen Nummern gespeichert, die der Kunde in kurzer Zeit mit der Tastatur eingeben kann oder über komplexere Dateien, die auf Datenträgern oder WANs übertragen werden.

Zum Speichern von Lizenzierungsinformationen auf dem lokalen System werden folgende Orte verwendet:

- einfache Dateien,
- versteckt in anderen Dateien,
- in der Windows-System-Registrierung,
- versteckt innerhalb des Dateisystems.

6.2.2.2.6. Probleme bei Systemen ohne Sicherungsmassnahmen gegen Umgehung des Kopierschutzes

Manche Systeme beinhalten Maßnahmen, welche die unlizenzierte Nutzung eines Programm verhindern, enthalten aber keine Maßnahmen, welche die Entfernung dieses Schutzes verhindern. Die einfachste Methode, eine Softwareabsicherung zu umgehen, ist zu verhindern, daß die entsprechenden

Funktionen überhaupt aufgerufen werden. Sie bietet auch dann einen Ansatzpunkt, wenn die Nachbildung einer bestimmten Hardware oder die Nutzung gewisser Netzwerkparameter nicht möglich ist.

Mit folgenden Gefahren sieht man sich konfrontiert:

a) Die Lizenz wird in einer Funktion in einer dynamisch gebundenen Bibliothek geprüft; dabei liefert die Funktion zurück, ob eine gültige Lizenz vorhanden ist. Dann läßt sich diese Bibliothek durch eine eigene ersetzen, die immer eine gültige Lizenz vorgibt. Dies wird dadurch begünstigt, daß die Firmen, die solche Systeme zur Absicherung der eigenen Programme nutzen, eine Beschreibung der Lizenz-API zur Integration in ihre eigene Software benötigen. Oft wird diese sogar zu Werbezwecken veröffentlicht, um zu illustrieren, wie einfach die Lizenzprüfungsroutinen in eigene Software integriert werden kann.

b) Die Lizenz wird innerhalb des Hauptprogramms an zentraler Stelle geprüft. Dann läßt sich diese Funktion mit Hilfe von Debuggern oder Disassemblern lokalisieren und über eine Änderung des Maschinencodes so manipulieren, daß das Programm von einer gültigen Lizenz ausgeht. Die Vielfalt der im Internet verfügbaren manipulierten Programme (oft als “Crack” bezeichnet) illustriert, daß dieser Weg durchaus gangbar ist. So findet die Internet-Suchmaschine Google bei der Suche nach “Crack” ca. 148 Millionen Seiten [Gooa]; dabei finden sich schon auf der ersten Seite solche Angebote.

6.2.2.2.7. Koppelung an andere Software

Neben der Prüfung des Vorhandenseins gewisser Hardware läßt sich ein Programm auch dadurch schützen, daß es gewisse Merkmale anderer Software abfragt. In diesem Falle müßte a) sichergestellt sein, daß diese auch auf dem Zielsystem vorhanden ist und b) auch über einen eigenen Schutz vor unlizenzierter Nutzung verfügt. Um dies zu gewährleisten, wird i.d.R. eine eindeutige Kennung des Betriebssystems überprüft (z. B. die “Product ID” bei Microsoft Windows). Außerdem ist denkbar, daß Produkte angeboten werden, deren einziger Zweck die eindeutige Markierung eines Systems ist und dessen Werte dann von verschiedenen Herstellern in ihren Produkten überprüft werden. In diesen Fällen verläßt sich der Hersteller einer Software dann darauf, daß der Lizenzschutz des Drittherstellers so funktioniert, daß jedes System über andere Kennung verfügt. Ist die zugrundegelegte Software jedoch schon eine unlizenzierte Kopie, so versagt auch die Lizenzschutz des eigenen Programms.

6.2.2.3. Systeme mit Online-Lizenzprüfung beim Programmstart

Diese Systeme erfordern beim Start der Applikation eine Netzwerkverbindung zum Hersteller (i.d.R. über das Internet). Dieser prüft dann die Lizenz

und teilt der Applikation mit, ob und in welchem Umfang die Funktionalität nutzbar sein soll.

Vorteil dieser Lösung für den Hersteller ist, daß er recht genaue Daten über die Nutzung seines Programms hat. Damit er unlizenzierte Nutzung erkennen kann, muß die Software darüber hinaus in der Lage sein, zwischen mehreren Systemen zu unterscheiden, die mit derselben Lizenz betrieben werden. Um dies sicher feststellen zu können, werden Techniken wie bei der Hardwarekoppelung von Software (vgl. 6.2.2.2.1) benötigt. Außerdem lassen sich im Rahmen einer solchen Onlineverbindung automatisch Updates und Fehlerbehebungen einspielen.

Nachteile ergeben sich hauptsächlich für den Endkunden. So kann der Hersteller genaue Statistiken über die Nutzung der Software erstellen und diese darüber hinaus noch einer Person zuordnen. Der Inhalt der übertragenen Daten ist oft für den Endkunden nicht transparent. Außerdem ist eine Nutzung ohne Onlineverbindung (z. B. bei der Nutzung von Notebooks auf Reisen) nicht möglich. Außerdem sind diese Produkte nicht mehr nutzbar, wenn der Hersteller sie nicht mehr unterstützt. Ein Nachteil für den Hersteller ist, daß er die Lizenzprüfungsinfrastruktur unbeschränkt zur Verfügung stellen muß und auch Kosten für die entsprechende Bandbreite trägt.

6.2.2.4. Systeme mit Netzwerk-Lizenzserver

Bei derartigen System wird davon ausgegangen, daß ein Endkunde mehrere Lizenzen für eine Software innerhalb eines Netzwerks einsetzen will, wobei i. allg. nur die gleichzeitig aktiven Systeme berechnet werden (sogenannte “floating licenses”). Ein System verwaltet die Lizenzen und übernimmt beim Programmstart die Lizenzprüfung (wie bei 6.2.2.3). Allerdings verlassen die Lizenzdaten dabei nicht das Firmennetz und der Hersteller bekommt keine Informationen über die Nutzung des Programms. Die Nutzung funktioniert auch, wenn der Hersteller das Produkt nicht mehr unterstützt. Schließlich eignet das System sich gut für Produkte, die ohnehin eine Client-Server-Architektur haben, da die Prüffunktionalität in den Server der Applikation eingebettet werden kann. Außerdem muß der Hersteller keine Infrastruktur zur Verfügung stellen.

Ein Nachteil ist, daß auch dieses System ohne die notwendige Netzwerkverbindung zum Server nicht funktioniert.

Zu beachten ist, daß der Server noch auf andere Weise vor unlizenzierter Nutzung geschützt werden muß, da ansonsten mit derselben Lizenz in mehreren Netzwerken gearbeitet werden könnte.

6.2.2.5. Systeme mit verteilter Netzwerk-Lizenzprüfung

Diese Technologie sieht vor, daß jedes System seine Lizenz zunächst einmal selbst prüft. Dann wird versucht, zu anderen Instanzen dieses Programms

im gleichen Netzwerk Kontakt aufzunehmen, um die Lizenzierungsinformationen zu vergleichen. So kann dann erkannt werden, daß noch weitere Systeme mit derselben Lizenz betrieben werden und das Programm kann mit einer passenden Meldung beendet werden. Diese Prüfung kann im Grundsatz auch auf das Internet ausgedehnt werden. Nachteil dieser Technik ist, daß in modernen Netzwerkinfrastrukturen der Zugriff auf das Netzwerk oft eingeschränkt wird, so daß eine Verbindungaufnahme zu anderen Systemen nicht möglich ist. Außerdem können Nutzer, die ein Programm mit ein und derselben Lizenzierungsinformation mehrfach nutzen wollen, entweder ihre Geräte für die Zeit der Nutzung physikalisch vom Netzwerk trennen oder den Datenaustausch mittels Programmen wie einer Benutzerfirewall verhindern.

6.2.2.6. Systeme mit Lizenzprüfung beim Update

Neben der regelmäßigen Prüfung bei der Ausführung eines Programms besteht auch die Möglichkeit, Lizenzen beim Update auf einen neue Version zu prüfen. Dazu verbindet sich das Programm auf einen zentralen, über das Internet erreichbaren Server, um sich selbst gegen eine neuere Version auszutauschen. Im Rahmen dieses Verfahren prüft der Server das vorhandensein einer gültigen Lizenz. Für manche Systeme wie z. B. XPress Update [Pix] ist dies die einzige Lizenzprüfung, die vorgenommen wird.

6.2.2.7. Lizenzgenerierung, -verwaltung und -vertrieb

Viele System bieten neben der eigentlichen Lizenzschutzfunktionalität auch weitere Leistungen, die eher dem administrativen Umfeld zuzuordnen sind.

Dazu gehören:

- ein Programm zur Erstellung der Lizenzinformationen,
- ein Programm, das darüber hinaus noch die Verwaltung der verschiedenen bereits vertriebenen Lizenzen ermöglicht,
- eine API zur Lizenzerzeugung, die sich in Verwaltungsapplikationen des Herstellers (z. B. ERP-Systeme), mit einbinden läßt,
- ein eigener Webserver, der die notwendige Funktionalität für Online-Aktivierung oder -Registrierung zur Verfügung stellt,
- eine API, die es erlaubt, Online-Aktivierung oder -Registrierung in eigene Webapplikationen einzubinden.

6.2.3. Übersicht über die Systeme

6.2.3.1. Leistungsübersicht der untersuchten Produkte

In den folgenden Tabellen, gruppiert nach dem Grundprinzip der Absicherung, werden die Leistungsmerkmale der einzelnen Produkte dargestellt. Die Tabellen 1 und 2 sind aus Gründen der Übersicht getrennt und geben einen

Überblick über Systeme, deren Lizenzschutz PC-Hardware prüft, während Tabelle 3 die Systeme behandelt, die sonstige Hardware prüfen.

Produkt	ACProtect [Ris]	ActiveLock [Fer]	AntiCrackProtector [Sis]	Armadillo™[Dig; Mil]	ASProtect [ASP]	Atma Software 1Way [Atm]	Atma Software AppPackager [Atm]	Byssus [Bys]	CodeProtector [Cus]	CrypKey [Cry]	Easy Licenser [Agi]	Exe Shield [Exe]	HASP SL [Alaa]	Ionworx [Ion]	Licence Protector [Mir]	Nalpeiron Protector [Nal]
Hersteller	Risco Software Inc.	Nelson Ferraz	Siskinsoft	Milde Software Solutions	ASProtect Software	Atma Software	Atma Software	Byssus Software Ltd.	Custom Software Solutions Inc.	CrypKey Inc.	Agiglis Software LLC	Exeshield	Aladin Knowledge Systems	Ionworx Technology	Mirage Computer Systems	Nalpeiron
Prüfung von Hardware und anderer Software	X		X	X	X		-	X	-	X	X	X	X	X	X	X
Hostname															X	
NetBIOS-Name															X	
MAC-Adresse				X				X							X	
CPU ID				X										X	-	
Festplatte/Datenträger				X				X							X	
Volume ID															X	
Hersteller ID				X											-	
CD/DVD-Laufwerke															-	
vom Programmierer zu implementieren	X		X							X	X				-	
kann von Programmierer implementiert werden						X										
BIOS				X										X	-	
Motherboard															-	
Speicher				X											-	
IP-Adresse															X	
nicht spezifizierte Hardware																
				⋮												

Produkt	ACProtect [Ris]	ActiveLock [Fer]	AntiCrackProtector [Sis]	Armadillo™[Dig; Mil]	ASProtect [ASP]	Atma Software 1Way [Atm]	Atma Software AppPackager [Atm]	Byssus [Bys]	CodeProtector [Cus]	CrypKey [Cry]	Easy Licenser [Agi]	Exe Shield [Exe]	HASP SL [Alaa]	Ionworx [Ion]	Licence Protector [Mir]	Nalpeiron Protector [Nal]
Installationspfad (im Netzwerk)															X	
Prüfung der Betriebssystem ID											X				X	
Integration in Applikation																
Verschlüsselung der Binärdateien	X		X	X			X			X		X	X	X		
Einbau in Binärdatei																
Zusatz-Bibliothek																
dynamisch (DLL)		X				X					X				X	X
statisch (LIB)								X								
OCX/COM-DLL																
Quellcode									X							
Funktionseinschränkungen																
Evaluierungsversion	X		X	X	X	X		X		X	X	X	X	X	X	X
Zeitbeschränkung Anzahl Tage	X		X	X	X	X				X	X	X	X	X	X	X
Zeitbeschränkung Fixes Enddatum	-		-	-									X	X	X	
Zeitbeschränkung Laufzeit in Minuten													X			
Anzahl Aufrufe	X		X	X	X	X							X	X	X	X
Sonstige Zähler															X	
eingeschränkter Funktionsumfang	X		X	X	X								X		X	
Verlängerungsschlüssel																
sonstige Beschränkungen																
Anzahl Benutzer															X	
Anzahl Benutzer pro Gruppe/Rolle															X	
Anzahl gleichzeitig im Netzwerk arbeitender Benutzer											X				X	X
Anzahl Prozessoren											X				-	
Taktfrequenzen der Prozessoren											X				-	
⋮																

Produkt	ACProtect [Ris]	ActiveLock [Fer]	AntiCrackProtector [Sis]	Armadillo™[Dig; Mil]	ASProtect [ASP]	Atma Software 1Way [Atm]	Atma Software AppPackager [Atm]	Byssus [Bys]	CodeProtector [Cus]	CrypKey [Cry]	Easy Licenser [Agi]	Exe Shield [Exe]	HASP SL [Alaa]	Ionworx [Ion]	Licence Protector [Mir]	Nalpeiron Protector [Nal]
Anzahl Computer															X	
Anzahl Element/Zähler																X
Ländercode													X		X	
Module																
direkt				X											X	X
über benutzerspezifische Daten in der Lizenz											X			X		
volle Möglichkeiten pro Modul															X	
benutzerspezifische Daten																
3rd party licencing																
Hilfsfunktionalität																
Lizenzprüfungs-API für Zielprogramm																
notwendig		X						X			X				X	X
optional			X			X				X						
Netzwerklizenzserver											X	X			-	
Dezentrale Netzwerklizenzprüfung															X	
Internetlizenzserver			X													
Lizenzgenerierung																
Programm	X									X	X				X	
API											X				X	
integrierbar in Webserver															X	
Lizenzverwaltung																
Programm																
API																
Blacklist	X		X		X										X	
Lizenztransfer													X		X	
Aktivierung																
über Webserver													X	X	X	
über Webserver beim Hersteller								X							X	
über Datei													X	-	X	
⋮																

Produkt	ACProtect [Ris]	ActiveLock [Fer]	AntiCrackProtector [Sis]	Armadillo™[Dig; Mil]	ASProtect [ASP]	Atma Software 1Way [Atm]	Atma Software AppPackager [Atm]	Byssus [Bys]	CodeProtector [Cus]	CrypKey [Cry]	Easy Licenser [Agi]	Exe Shield [Exe]	HASP SL [Alaa]	Ionworx [Ion]	Licence Protector [Mir]	Nalpeiron Protector [Nal]
über Code (auch Telefon, Fax, usw.)													X	X	X	
Online-Lizenzprüfung																
immer beim Programmstart																
optional beim Programmstart															X	
optional zyklisch															X	
bei der Aktivierung															X	
integrierter Webserver	X														X	
Prüfung beim Update														X		
Anpassung an eigene Applikation																
eigene Dialoge															X	
eigene Texte															X	
Anti-Spoofing		-				-									X	
Verschlüsselung des Programms	X		X	X	X		X			X	X	X		X		X
Komprimierung des Programms	X		X	X	X		X				X	X		X		X
Erkennung von Debuggern	X				X						X	X	X	X		
Erkennung/Behinderung von Disassemblern	X				X						X	X		X		
Erschwerung von Memory Dumps	X				X						X	X		X		
Erschwerung von Memory Patches	X		X		X						X	X		X		
Erschwerung von Traces											X	X		X		
Polymorphismus	X		X													
Erkennung der Rückstellung der Systemzeit										X	X	X				
Erkennung eines Rollbacks oder Klonens des Systems											X	X				
Audit/Überwachung der Keys											X					
⋮																

Produkt	ACProtect [Ris]	ActiveLock [Fer]	AntiCrackProtector [Sis]	Armadillo™[Dig; Mil]	ASProtect [ASP]	Atma Software 1Way [Atm]	Atma Software AppPackager [Atm]	Byssus [Bys]	CodeProtector [Cus]	CrypKey [Cry]	Easy Licenser [Agi]	Exe Shield [Exe]	HASP SL [Alaa]	Ionworx [Ion]	Licence Protector [Mir]	Nalpeiron Protector [Nal]
Markierung von DLLs															X	
CRC check von DLLs															X	
Plattformen und Umgebungen																
Programmiersprachen und Entwicklungsumgebungen																
können C/C++ DLLs aufrufen	X		X	X		X		X				X		X	X	
können COM Objekte aufrufen																
C/C++									X							
Java											X				X	
Visual Basic 6											X				X	
Microsoft .NET								X	X		X				X	
Delphi															X	
sonstige															X	
Betriebssysteme (Zielapplikation)																
Windows 9x	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
Windows 2000, XP, 2003	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
Linux auf i386								X			X					
anderer Unix-Derivate											X					
MacOS																
DOS, 16bit Windows																
sonstige																

Tabelle 1: Leistungsmerkmale der untersuchten Produkte: Bindung an PC-Hardware und/oder Netzwerkparameter I

Produkt	Obsidium Software Protection [Tor]	OLicense Suite [Opt]	PC Guard [Cek]	PELock [Woi]	Quick License Manager [Int]	RSS Solid Licence [Rei]	SDProtector [SDP]	Sheriff SDK [Lic]	softwareKEY Protection Plus [Con]	SoftwareShield [Sof]	SVK Protector [Ant]	Themida [Ore]	Xope Systems Licence Management Studio [Xop]	XPressUpdate [Pix]
Hersteller	Obsidium Software	Optimum GmbH	Software Protection Labs	Bartosz Wojczik	Interactive Studios Inc.	Reinhold Software Services	SDProtector	Licensing Technologies Inc.	Concept Software Inc.	SoftwareShield Technologies	AntiCracking	Oreans Technologies	Xope Systems	Pixel Planet GmbH
Prüfung von Hardware und anderer Software	X	X	X		X	X	X	X	X	X	X	X	X	
Hostname														
NetBIOS-Name														
MAC-Adresse	X	X							X					
CPU ID	X		X											
Festplatte/Datenträger	X		X											
Volume ID									X					
Hersteller ID			X											
CD/DVD-Laufwerke			X											
vom Programmierer zu implementieren														
kann von Programmierer implementiert werden									X					
BIOS			X						X					
Motherboard														
Speicher														
IP-Adresse			X											
			⋮											

Produkt	Obsidium Software Protection [Tor]	OLicense Suite [Opt]	PC Guard [Cek]	PELock [Woi]	Quick License Manager [Int]	RSS Solid Licence [Rei]	SDProtector [SDP]	Sheriff SDK [Lic]	softwareKEY Protection Plus [Con]	SoftwareShield [Sof]	SVK Protector [Ant]	Themida [Ore]	Xope Systems Licence Management Studio [Xop]	XPressUpdate [Pix]
nicht spezifizierte Hardware									X					
Installationspfad (im Netzwerk)														
Prüfung der Betriebssystem ID	X													
Integration in Applikation														
Verschlüsselung der Binärdateien			X				X		X		X	X	X	
Einbau in Binärdatei									X					
Zusatz-Bibliothek														
dynamisch (DLL)		X				X		X	X					X
statisch (LIB)									X					
OCX/COM-DLL					X			X	X	X				
Quellcode									X					
Funktionseinschränkungen														
Evaluierungsversion	X	X	X		X		X	X	X	X	X	X	X	
Zeitbeschränkung Anzahl Tage	X	X	X		X			X	X	X	X	X		
Zeitbeschränkung Fixes Enddatum	X	X	X						X	X		X		
Zeitbeschränkung Laufzeit in Minuten			X									X		
Anzahl Aufrufe	X		X						X	X		X		
Sonstige Zähler									X					
eingeschränkter Funktionsumfang		X	X				X	X	X	X	X		X	
Verlängerungsschlüssel												X		
sonstige Beschränkungen														
Anzahl Benutzer									X					
⋮			⋮											

Produkt	Obsidium Software Protection [Tor]	OLicense Suite [Opt]	PC Guard [Cek]	PELock [Woi]	Quick License Manager [Int]	RSS Solid Licence [Rei]	SDProtector [SDP]	Sheriff SDK [Lic]	softwareKEY Protection Plus [Con]	SoftwareShield [Sof]	SVK Protector [Ant]	Themida [Ore]	Xope Systems Licence Management Studio [Xop]	XPressUpdate [Pix]
Anzahl Benutzer pro Gruppe/Rolle									X					
Anzahl gleichzeitig im Netzwerk arbeitender Benutzer			X					X	X					
Anzahl Prozessoren														
Taktfrequenzen der Prozessoren									X					
Anzahl Computer									X					
Anzahl Element/Zähler														
Ländercode									X			X		
Module														
direkt		X							X					X
über benutzerspezifische Daten in der Lizenz			X											
volle Möglichkeiten pro Modul														
benutzerspezifische Daten									X					
3rd party licencing			X											
Hilfsfunktionalität														
Lizenzprüfungs-API für Zielprogramm														
notwendig		X			X	X		X	X	X				
optional	X		X	X			X							
Netzwerklizenzserver		X	X						X					
Dezentrale Netzwerklizenzprüfung														
Internetlizenzserver														
Lizenzgenerierung														
			⋮											

Produkt	Obsidium Software Protection [Tor]	OLicense Suite [Opt]	PC Guard [Cek]	PELock [Woi]	Quick License Manager [Int]	RSS Solid Licence [Rei]	SDProtector [SDP]	Sheriff SDK [Lic]	softwareKEY Protection Plus [Con]	SoftwareShield [Sof]	SVK Protector [Ant]	Themida [Ore]	Xope Systems Licence Management Studio [Xop]	XPressUpdate [PiX]
Programm		X						X	X	X				
API	X	X			X	X	X		X					
integrierbar in Webserver									X					X
Lizenzverwaltung														
Programm														
API														
Blacklist	X			X			X		X					
Lizenztransfer		X					X		X					
Aktivierung														
über Webserver					X		X	X	X				X	
über Webserver beim Hersteller									X					
über Datei								X	X					
über Code (auch Telefon, Fax, usw.)							X	X	X					
Online-Lizenzprüfung														
immer beim Programmstart														
optional beim Programmstart														
optional zyklisch														
bei der Aktivierung														
integrierter Webserver														
Prüfung beim Update														X
Anpassung an eigene Applikation														
eigene Dialoge									X					
eigene Texte									X					
Anti-Spoofing														
⋮														

Produkt	Obsidium Software Protection [Tor]	OLicense Suite [Opt]	PC Guard [Cek]	PELock [Woi]	Quick License Manager [Int]	RSS Solid Licence [Rei]	SDProtector [SDP]	Sheriff SDK [Lic]	softwareKEY Protection Plus [Con]	SoftwareShield [Sof]	SVK Protector [Ant]	Themida [Ore]	Xope Systems Licence Management Studio [Xop]	XPressUpdate [Pix]
Verschlüsselung des Programms	X		X	X			X		-		X	X		
Komprimierung des Programms	X			X			X		X			X		
Erkennung von Debuggern				X			X	X	X		X	X		
Erkennung/Behinderung von Disassemblern				X					X		X	X		
Erschwerung von Memory Dumps				X			X		X			X		
Erschwerung von Memory Patches				X			X		X		X	X		
Erschwerung von Traces				X			X				X	X		
Polymorphismus														
Erkennung der Rückstellung der Systemzeit		X					X		X					
Erkennung eines Rollbacks oder Klonens des Systems									X					
Audit/Überwachung der Keys														
Markierung von DLLs														
CRC check von DLLs														
Plattformen und Umgebungen														
Programmiersprachen und Entwicklungsumgebungen														
können C/C++ DLLs aufrufen	X				X	X	X		X					
⋮														

Produkt	Obsidium Software Protection [Tor]	OLicense Suite [Opt]	PC Guard [Cek]	PELock [Woi]	Quick License Manager [Int]	RSS Solid Licence [Rei]	SDProtector [SDP]	Sheriff SDK [Lic]	softwareKEY Protection Plus [Con]	SoftwareShield [Sof]	SVK Protector [Ant]	Themida [Ore]	Xope Systems Licence Management Studio [Xop]	XPressUpdate [Pix]
können COM Objekte aufrufen		X												
C/C++				X										
Java														
Visual Basic 6									X					
Microsoft .NET									X					
Delphi				X										
sonstige														
Betriebssysteme (Zielapplikation)														
Windows 9x	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Windows 2000, XP, 2003	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Linux auf i386		X												
anderer Unix-Derivate		X												
MacOS														
DOS, 16bit Windows			X											
sonstige														

Tabelle 2: Leistungsmerkmale der untersuchten Produkte: Bindung an PC-Hardware und/oder Netzwerkparameter II

Produkt	ContraCopy [Alk]	GAPS [GAP]	Guardant [Akt]	HASP SL [Alaa]	KEY-LOCK [Mica]	Matrixlock [TDi]	CRYPTO-BOX [MARc]	Nalpeiron Flash [Nal]	PACE [PAC]	Dinkey Dongles [Rei]	SecuROM [Son]	Sentinel UltraPro [Saf]	SG-Lock [SG]	StarForce Pro [Pro]	Wizzkey [Wiz]
Hersteller	Alkonost Software	Global Anti-Piracy Systems	Aktiv Company	Aladin Knowledge Systems	Microcomputer Applications Inc.	TDi GmbH	Marx Software Security	Nalpeiron	PACE Anti Piracy	Reinhold Software Services	Sony DADC	SafeNet Inc.	SG Intec Ltd. & Co. KG	Protection Technology Russia	Wizzkey Technology b.v.
Prüfung von Hardware und anderer Software															
Floppy	X														
USB Stick															
Flash-Speicher								X							
CD-ROM/DVD-ROM		X									X			X	
Dongle		X	X	X	X	X	X		X	X		X	X		X
ohne freien Speicher															
mit freiem Speicher		X	X	X	X			X		X			X		X
Steckkarte (ISA/PCI)							X								
zusätzlich Prüfung von PC Hardware		X												X	
Integration in Applikation															
Verschlüsselung der Binärdateien			X	X	X	X	X		X		X			X	
Einbau in Binärdatei						X									
Zusatz-Bibliothek															
dynamisch (DLL)	X	X	X	X	X		X	X		X		X	X		X
statisch (LIB)															
statisch (OBJ)					X										X
OCX/COM-DLL							X								
Quellcode			X				X								
Funktionseinschränkungen															
Evaluierungsversion		X	X	X			X	X	X			X	X	X	
Zeitbeschränkung Anzahl Tage				X				X	X			X	X	X	
Zeitbeschränkung Fixes Enddatum					X		X		X			X	X		

⋮

Produkt	ContraCopy [Alk]	GAPS [GAP]	Guardant [Akt]	HASP SL [Alaa]	KEY-LOCK [Mica]	Matrixlock [TDi]	CRYPTO-BOX [MARc]	Nalpeiron Flash [Nal]	PACE [PAC]	Dinkey Dongles [Rei]	SecuROM [Son]	Sentinel UltraPro [Saf]	SG-Lock [SG]	StarForce Pro [Pro]	Wizzkey [Wiz]
Zeitbeschränkung Laufzeit in Minuten				X											
Anzahl Aufrufe			X					X				X	X	X	
Sonstige Zähler					X							X	X		
eingeschränkter Funktionsumfang			X	X				X	X						
sonstige Beschränkungen															
Anzahl Benutzer					X										
Anzahl Benutzer pro Gruppe/Rolle															
Anzahl gleichzeitig im Netzwerk arbeitender Benutzer			X	X	X		X	X							
Anzahl Prozessoren															
Taktfrequenzen der Prozessoren															
Anzahl Computer					X										
Anzahl Element/Zähler															
Ländercode															
Module			X	X		X									
direkt															
über benutzerspezifische Daten in der Lizenz															
volle Möglichkeiten pro Modul															
benutzerspezifische Daten															
3rd party licencing															
Hilfsfunktionalität															
Lizenzprüfungs-API für Zielprogramm															
notwendig	X			X	X					X			X		X
optional		X				X	X		X						
Netzwerklizenzserver					X		X	X							
Dezentrale Netzwerklizenzprüfung															
Internetlizenzserver															
Lizenzgenerierung															
Programm	X			X											X
		⋮													

Produkt	ContraCopy [Alk]	GAPS [GAP]	Guardant [Akt]	HASP SL [Alaa]	KEY-LOCK [Mica]	Matrixlock [TDi]	CRYPTO-BOX [MARc]	Nalpeiron Flash [Nal]	PACE [PAC]	Dinkey Dongles [Rei]	SecuROM [Son]	Sentinel UltraPro [Saf]	SG-Lock [SG]	StarForce Pro [Pro]	Wizzkey [Wiz]
API integrierbar in Webserver															
Lizenzverwaltung Programm															
API															
Blacklist															
Lizenztransfer								X							
Aktivierung							X	X	X					X	
über Webserver							X	X						X	
über Webserver beim Hersteller															
über Datei							X								
über Code (auch Telefon, Fax, usw.)							X	X							
über Programm							X		X						
Online-Lizenzprüfung											X			X	
immer beim Programmstart															
optional beim Programmstart															
optional zyklisch															
bei der Aktivierung															
integrierter Webserver															
Prüfung beim Update															
Anpassung an eigene Applikation															
eigene Dialoge									X						
eigene Texte									X						
Anti-Spoofing							X								
Verschlüsselung des Programms				X			X							X	
Komprimierung des Programms				X			X							X	
Erkennung von Debuggern			X	X	X	X	X								
Erkennung/Behinderung von Disassemblern			X				X								
Erschwerung von Memory Dumps							X								
⋮		⋮													

Produkt	ContraCopy [Alk]	GAPS [GAP]	Guardant [Akt]	HASP SL [Alaa]	KEY-LOCK [Mica]	Matrixlock [TDi]	CRYPTO-BOX [MARc]	Nalpeiron Flash [Nal]	PACE [PAC]	Dinkey Dongles [Rei]	SecuROM [Son]	Sentinel UltraPro [Saf]	SG-Lock [SG]	StarForce Pro [Pro]	Wizzkey [Wiz]
Erschwerung von Memory Patches														X	
Erschwerung von Traces							X								
Polymorphismus															
Erkennung der Rückstellung der Systemzeit															
Erkennung eines Rollbacks oder Klonens des Systems															
Audit/Überwachung der Keys															
Markierung von DLLs															
CRC check von DLLs															
Erkennung von CD Emulatoren														X	
Code auf Dongle		X													
Anmeldung an Dongle mit Key													X		
Plattformen und Umgebungen															
Programmiersprachen und Entwicklungsumgebungen															
können C/C++ DLLs aufrufen	X	X	X	X	X	X	X			X		X	X		X
können COM Objekte aufrufen															
C/C++		X					X					X	X		
Java		X					X					X	X		
Visual Basic 6		X					X					X	X		
Microsoft .NET							X					X	X		
Delphi		X					X					X	X		
Assembler		X													
Betriebssysteme (Zielapplikation)															
Windows 9x	X	X	X	X	X	X	X	X	X	X	98	X	X	X	X
Windows 2000, XP, 2003	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Linux auf i386				X		X	X						X		
andere Unix-Derivate							X								
MacOS				X		X	X					X			
⋮		⋮													

Produkt	ContraCopy [Alk]	GAPS [GAP]	Guardant [Akt]	HASP SL [Alaa]	KEY-LOCK [Mica]	Matrixlock [TDi]	CRYPTO-BOX [MARc]	Nalpeiron Flash [Nal]	PACE [PAC]	Dinkey Dongles [Rei]	SecuROM [Son]	Sentinel UltraPro [Saf]	SG-Lock [SG]	StarForce Pro [Pro]	Wizzkey [Wiz]
16bit Windows		X	X		X	X									
DOS		X	X		X										
sonstige															

Tabelle 3: Leistungsmerkmale der untersuchten Produkte: Bindung an externe Hardware

6.2.4. Bewertung und Zusammenfassung

Die meisten untersuchten Kopierschutz-Produkte konzentrieren sich darauf, eine Software an ein bestimmtes System oder eine andere Hardware-Komponente zu koppeln. Die Übertragung von Lizenzierungsinformationen an den Kunden erfolgt über das Internet oder über (alpha)numerische Schlüssel. Alternativ werden vielfach auch Online-Lizenzprüfungen beim Programmstart vorgesehen. Die meisten Produkte konzentrieren sich auf die aktuellen Betriebssysteme der Firma Microsoft. Viele Produkte bieten APIs an, welche die Entwicklung eigener Konzepte und Erweiterungen ermöglichen.

Nur wenige Produkte bieten überhaupt Unterstützung für Module an (vgl. Tabellen 1, 2 und 3). Bei diesen Produkte gibt es zwar eine beschränkte oder unbeschränkte Anzahl von booleschen Ausdrücken – d. h. von einfachen Schaltern, mit dem sich eine Funktion ein- oder ausschalten läßt – innerhalb der Lizenzierungsinformationen, eine Unterstützung komplexerer Strukturen fehlt jedoch. In diesem Bereich sind daher immer Anpassungen an die eigene Software notwendig. So bieten viele Hersteller das Freischalten einer Software über einen an die Hardware gekoppelten Schlüssel an, während dies für die Erweiterung um einzelne Module nicht vorgesehen ist. Lizenzierung von Erweiterungsschnittstellen an Dritthersteller ist überhaupt nicht vorgesehen. Das in der Tabelle als “3rd Party Licensing” bezeichnete und nur im Produkt PCGuard [Cek] implementierte Leistungsmerkmal bezeichnet nur die Möglichkeit, daß Händler die benötigten Schlüssel selbst generieren und keine Behandlung von Dritthersteller-Modulen.

Die Nutzung eines fertigen Produkts hat den Vorteil, daß es gegenüber eigenen Lösungen länger im Einsatz ist, technisch ausgereift sein sollte und mehr Aufwand in Schutzmechanismen investiert wurde. Ein Nachteil einer solchen Lösung ist, daß sie zwangsläufig auch in verschiedenen anderen Produkten im Einsatz ist und daß, sobald eines dieser Produkt geknackt wur-

de, auch das eigenen Produkt nicht mehr sicher vor unlizenzierter Nutzung ist.

6.3. Schutz vor unberechtigter Verwendung

Im folgenden werden konkrete Maßnahmen beschrieben, die Software vor unlizenzierter Nutzung schützen. Außerdem wird die Vergabe einer Lizenz an einen neues oder ein anderes System beschreiben und dabei dargelegt, wie verhindert wird, daß Lizenzen unrechtmäßig vergeben werden.

6.3.1. Koppelung an Hardware

6.3.1.1. Koppelung an PC-Bauteile

6.3.1.1.1. Abfrage der CPUID

Das Beispiel Quellcode 2 legt dar, wie auf einem Pentium-kompatiblen System die Daten des Prozessors abgefragt werden können. Bei Prozessoren, die dies unterstützen, wird darüber hinaus noch die Seriennummer abgefragt. Diese ist dann eindeutig.

6.3.1.1.2. Abfrage der MAC-Adresse

Wird eine Lizenz an eine eindeutige Kennung eines Geräts im Ethernet-Netzwerk (MAC-Adresse) gekoppelt, so kann diese Lizenz auf Geräten, die im gleichen Netzwerk angeschlossen sind, nur einmal verwendet werden. Das Beispiel Quellcode 3 zeigt, wie diese Kennung bei den Windows-Betriebssystemen ab Windows 2000 geprüft werden kann.

6.3.1.1.3. weitere Hardwareeigenschaften

Die Betriebssysteme ermöglichen oft auch die Abfrage weiterer Werte, wie

- die Hersteller-ID der Festplatte,
- die BIOS-Version,
- die Motherboard-Kennung,
- die Größe des eingebauten Arbeitsspeichers,
- die Anzahl der Prozessoren,
- die Anzahl und Typen der angeschlossenen optischen Laufwerke,
- Informationen über weitere eingesetzte Komponenten wie Graphikchip, SCSI-Controller oder IDE-Controller (vgl. auch [Mich]).

Bei den neueren Windows-Betriebssystemen können viele Werte über die “Windows Management Instrumentation” abgefragt werden (vgl. [Micg; Li]). Ein Beispiel für die Bestimmung der Seriennummer findet sich unter [Isk].

Unter Unix-Systemen sind solche Informationen über das proc-Dateisystem abrufbar. Außerdem kann man von der Verfügbarkeit von gewissen Gerätedateien im dev-Dateisystem auf das Vorhandensein von bestimmter Hardware schließen.

6.3.1.2. Koppelung an logische Einstellungen des Systems

Der Schutz kann an auch im lokalen Netzwerk einmalige Parameter gebunden werden. Auf diese Art und Weise kann eine mehrfache Nutzung im gleichen Netzwerk verhindert werden. Auch eine Übertragung auf andere Rechner wird erschwert, da für eine unberechtigte Nutzung derselben Lizenz zunächst die fremden Einstellungen übernommen werden müßten, die u. U. eine Nutzung mit Verbindung in das lokale Netzwerk verhindern würde. Bei Programmen, die unabhängig verwendet werden können, bietet diese Vorgehensweise wenig Schutz. Folgende Parameter bieten sich für eine Prüfung an:

- die IP-Adresse,
- der Hostname,
- Domänen-Einstellungen,
- der NetBIOS-Name.

Außerdem können auch Eigenschaften der Installation des lokalen Systems geprüft werden, wie

- die Volume ID eines Datenträgers (vgl. [Mice]),
- Installationspfade von Programmen oder Betriebssystembestandteilen,
- Benutzername des Benutzers, der das Programm aufruft,
- die genaue Version des Betriebssystems.

6.3.1.3. Koppelung an andere Hardware

6.3.1.3.1. Koppelung an Datenträger

Um sicherzustellen, daß ein Programm immer nur auf einem System gleichzeitig genutzt wird, kann man es an einen Datenträger wie eine CD-ROM oder Floppy-Diskette koppeln.

Dabei gibt es drei Möglichkeiten, die jeweils unterschiedlichen Schutz bieten:

a) Der Datenträger enthält ein normales Dateisystem und auf diesem ist eine Datei gespeichert, welche die notwendigen Lizenzierungsinformationen enthält. Dieses Verfahren hat den Vorteil, daß diese Datenträger mit Standardwerkzeugen erstellt werden können und den Nachteil, daß jeder, der sich eine Kopie des Datenträgers erstellt, das Programm auch ohne gültige Lizenz nutzen kann. Prüft man nicht zusätzlich noch die Art

des Datenträgers (CD-ROM, Floppy, Festplatte), könnte man die Datei sogar auf die Festplatte kopieren und das Programm unbeschränkt nutzen. Eine solche Prüfung kann beispielsweise unter Windows mit der API-Funktion GetDriveType (vgl. [Micf]) vorgenommen werden. Einfache Lösungen prüfen nicht einmal echte Lizenzinformationen, sondern nur das Vorhandensein der Installations-CD-ROM in einem Laufwerk.

b) Der Datenträger enthält kein Dateisystem und wird direkt gelesen. Bei einigen Betriebssystemen wie den Windows-Versionen kann ein solcher Datenträger nicht mit den mitgelieferten Programmen kopiert werden. Andere Systeme haben diese Möglichkeit jedoch vorgesehen (z. B. kann unter Linux über den Befehl
dd -if=/dev/fd0 -of=/dev/fd1
die Diskette im ersten Laufwerk Byte für Byte auf diejenige im zweiten Laufwerk kopiert werden). Für Windows gibt es Programme, die dies ebenfalls ermöglichen. Außerdem bieten viele virtuelle PCs die Möglichkeiten, virtuelle Datenträger anzugeben, hinter denen sich in Wirklichkeit Dateien verbergen. Auch hier bietet die Technik keinen wirklichen Schutz.

c) Die Datenträger sind mit bestimmten physikalischen Fehlern versehen. Diese lassen sich nicht mit jedem System und jedem Laufwerk einfach übertragen. Bei der Lizenzprüfung müssen dann die fehlerhaften Sektoren festgestellt und mit den in der Applikation gespeicherten Werten verglichen werden. Eine einfachere Variante dieses Vorgehens erstellt bei CD-ROMs nur Dateinamen, die beim Kopieren Fehler verursachen (weil sie nicht normgerecht oder zu lang sind oder ungültige Zeichen enthalten). Alternativ werden im Inhaltsverzeichnis der CD-ROM Dateigrößen angegeben, welche die tatsächliche Größe des Datenträgers überschreiten.

Alle drei Methoden lassen sich dadurch umgehen, daß die Prüffunktionalität im Maschinencode umgangen wird. Daher werden zur Lösung dieser Problematik zusätzliche Maßnahmen benötigt.

6.3.1.3.2. Koppelung an Kopierschutzstecker

Zum Schutz der Software wird ein spezieller Kopierschutzstecker (englisch: “Dongle”) eingesetzt. Er wird üblicherweise mit der seriellen oder parallelen Schnittstelle des Rechners oder mit einer USB-Schnittstelle verbunden. Einfache Dongles enthalten nur einen nichtflüchtigen Speicher, dessen Daten gelesen und ggf. auch geschrieben werden können. Komplexere Modelle enthalten dazu noch einen Mikrocontroller, der es erlaubt, gewisse Berechnungen wie kryptographische Algorithmen durchzuführen.

Bei der Nutzung von Kopierschutzsteckern gibt es folgende Modelle:

a) Die Applikation liest die Lizenzierungsinformationen vom Kopierschutz-

stecker und prüft diese. Dies läßt sich umgehen, indem das Protokoll auf der Schnittstelle abgehört und mit Hilfe eigener Hardware nachgebildet wird. Alternativ läßt sich auch ein eigener Treiber entwickeln, der dieselbe Schnittstelle zur Applikation verwendet, wie diejenige Hardware, an welcher der Kopierschutzstecker angeschlossen ist.

b) Die Applikation sendet Daten an den Dongle. Dieser berechnet daraus entweder einen Hash-Wert oder verschlüsselt sie. Das Ergebnis wird dann an die Applikation zurückgeschickt. Die Applikation vergleicht das Ergebnis dann mit ihren eigenen Berechnungen. Es nützt also nichts, die einmal aufgenommenen Antwortdaten zu protokollieren und danach von einem eigenen Gerät zu versenden. Ein Auslesen des auf dem Dongle verwendeten Codes ist nicht möglich. Umgehen läßt sich ein solcher Schutz, indem die Prüfsummenberechnung in der Applikation analysiert oder umgangen wird. Im Idealfall wird der Maschinencode der Applikation vom Dongle selbst erst entschlüsselt.

c) Der Kopierschutzstecker enthält Teile des Programmcodes der Applikation, der erst zur Laufzeit nachgeladen wird. So kann verhindert werden, daß Demoversionen ohne Besitz eine gültigen Dongles zur unlizenzierten Nutzung manipuliert werden können. Diese Vorgehensweise kann zur Absicherung mit b) kombiniert werden.

Sicher machen läßt sich ein Schutz mit Hilfe eines Dongles, indem – wie es von Krzystof Dyki zur Zeit in seiner Doktorarbeit untersucht wird (vgl. [Dyk06]) – neben der Entschlüsselung des eigentlichen Maschinencodes auch ein Teil des produktiven Codes der Applikation im Dongle untergebracht wird und die Programmierung des Dongles so gestaltet wird, daß sie sich nachträglich nicht mehr auswerten läßt. Dyki untersucht die Verwendung von Field Programmable Grid Arrays (FPGAs), d. h. programmierbaren Logikschaltkreisen mit Möglichkeiten zur Parallelverarbeitung, zu diesem Zweck. In diesem Fall wird der Algorithmus zur Lizenzprüfung bzw. zur Entschlüsselung der Applikation mit Hilfe der Logikschaltkreise implementiert. Vorteil dieser Lösung ist, daß es FPGAs gibt, deren Konfiguration sich nicht von außen auswerten läßt.

6.3.1.3.3. Koppelung an Magnet-, Chip- oder RFID-Karten

Zur Nutzung des Programms wird eine Identifikationskarte verwendet, wie sie auch in Zutrittskontroll- oder Zeiterfassungssystemen üblich ist. Übliche Verfahren zum Lesen der Identifikation sind Magnetstreifen, Chip-Karten (Lesen über Kontakt) oder über Funk zu lesende Karten. Bei einfachen System wie Magnetstreifen und manchen RFID-Chips können die Daten mit einfachen Lesegeräten kopiert werden. Bei fortgeschritteneren Systemen enthält jeder Chip ab Werk eine eindeutige Kennung, die nachträglich nicht mehr umprogrammiert werden kann. Manche Systeme sehen einen Schreib-

schlüssel vor, ohne den eine Karte nicht erstellt werden kann oder kennen Lizenzkarten, mit denen einem der Hersteller des Systems die Berechtigung zum Codieren von Karten gibt, die allerdings alle anhand einer festgelegten Kennung identifiziert werden können. Bei besseren RFID-Verfahren ist die Funkstrecke zwischen Karte und Leser verschlüsselt. Manipulationen können daher nur zwischen Leser und Applikation oder in der Applikation selbst vorgenommen werden. Gängige RFID-Karten enthalten außerdem eine Seriennummer, die bei der Herstellung festgelegt wird und nachträglich nicht mehr geändert werden kann. Wird auf der RFID-Karte noch eine PC-Hardwarekennung gespeichert und mit der Seriennummer der Karte verschlüsselt, so wird die Emulation des Ausweislesers zusätzlich erschwert. Ist darüber hinaus der Hersteller der Software auch Hersteller des Ausweislesegeräts, so kann er es um weitere Funktionen erweitern, welche die Authentizität des Lesers gewährleisten: Zu diesem Zweck werden sensible Teile des Maschinencodes oder der Lizenzierungsinformationen verschlüsselt ausgeliefert und zur Laufzeit an den Leser geschickt. Dieser kennt dann den notwendigen privaten Schlüssel und gibt als Antwort die entschlüsselten Daten an die Applikation zurück. Dies erschwert die Emulation eines solchen Lesers, da der private Schlüssel von außen nicht abgefragt werden kann. Die Entschlüsselung kann darüber hinaus noch abhängig von der Karte und deren Seriennummer gestaltet werden. Verfahren mit Identifikationskarten bieten sich insbesondere für Produkte an, die oft ohnehin mit entsprechenden Lesegeräten ausgeliefert werden; gerade bei Zutrittskontrollsystemen werden derartige Leser gerne zu Kontrollzwecken eingesetzt.

Ist die Lizenz an eine Firma oder Institution gekoppelt, kann die Karte auch zur Aktivierung nach der Installation verwendet werden. Sie wird danach nicht mehr benötigt. Sind die Kartenleser in der Lage, die Karten zu beschreiben, kann ein interner Zähler für die Anzahl installierter System geführt werden. Dieser enthält die Anzahl noch freier Lizenzen und wird bei Installation eines neuen System dekrementiert und bei Deinstallation inkrementiert. Auf diese Art und Weise lassen sich dann auch Lizenzen für eine bestimmte Anzahl Benutzer verwalten. Aus Sicherheitsgründen sollten die Karte auch die maximale Anzahl verfügbare Lizenzen enthalten, damit über manipulierte Deinstallationen nicht mehr Lizenzen entstehen als ursprünglich angenommen. Es ist sinnvoll, die Daten auf dem Ausweis zu verschlüsselt, wobei zumindest ein Teil des Schlüssel aus dem Ausweisleser stammt und von außen praktisch nicht zu manipulieren ist.

Identifikationskarten sind zum Schutz vor unlizenzierter Nutzung von Software bislang weder in Produkten im Einsatz, noch werden sie in der Literatur beschrieben. Aus diesem Grund werden in diesem Absatz Lösungen für deren Verwendung erarbeitet.

6.3.1.3.4. Koppelung an Biometrie des Benutzers

Ist eine Lizenz an die Person eines einzelnen Benutzers gekoppelt, kann man die Berechtigung zur Nutzung des Programms auch mit biometrischen Systemen prüfen. Gängige Verfahren sind Fingerabdruckprüfung, Handgeometrieprüfung, Gesichtserkennung oder Iriserkennung. Nachteil dieser Verfahren ist, daß es selbst bei optimaler Einstellung des Systems jeweils eine kleine Gruppe von Personen gibt, die recht oft nicht richtig identifiziert werden können. Außerdem sind Erkennungsfehlerquoten von bis zu 2% wegen Handhabungsproblemen normal. Schließlich haben manche Benutzer Angst vor diesen Verfahren, weil sie fürchten, daß ihre körperlichen Eigenschaften erfaßt und weitergegeben werden.

Soll die Lizenz an eine Firma oder Institution gekoppelt werden, kann die biometrische Prüfung zur Aktivierung der Software auf dem neuen System verwendet werden. Auf diese Art und Weise kann ein Administrator beliebig vielen Mitarbeitern eine Lizenz freischalten. Dieses Verfahren ist nur für Lizenzen geeignet, die an eine Institution unabhängig von der Anzahl der tatsächlichen Systeme gebunden sind. Biometrische Merkmale lassen sich darüber hinaus auch auf dieselbe Art und Weise wie Identifikationskarten einsetzen, sofern man von den Anwendungen, die eine Schreibfunktion benötigen, absieht.

Zur Zeit sind keine Systeme auf dem Markt, welche die Lizenz zur Nutzung einer Software mittels biometrischer Merkmale des Berechtigen prüfen. Daher wird in diesem Absatz herausgearbeitet, auf welche Art und Weise diese eingesetzt werden können.

6.3.2. Koppelung an andere Software

Ein Programm kann sich auch den bereits vorhandenen Kopierschutz eines anderen Programms zunutzemachen. Wenn beispielsweise das Betriebssystem schon an die konkrete Hardware gekoppelt ist und dazu noch eine eindeutige Kennung enthält, die von anderer Software geprüft werden kann, läßt sich damit leicht ein gleich mächtiger Schutz in das eigene Programm einbauen. Ein Beispiel für eine solche Technik ist die Nutzung der Betriebssystem-ID bei den Betriebssystemen Windows 2000 und XP der Firma Microsoft, wie im Beispielprogramm Quellcode 1 dargestellt ist.

Ein Vorteil dieser Lösung ist, daß sich die Daten leicht abfragen lassen. Von Nachteil ist, daß jemand, der bereits das Betriebssystem ohne gültige Lizenz betreibt, an dieser Stelle auch eine beliebige ID eintragen kann.

6.3.3. Prüfsummen

Prüfsummen lassen sich an verschiedenen Stellen zum Schutz vor unlizenzierter Nutzung eines Programms einsetzen. Am besten geeignet sind

Berechnungsverfahren, die bei kleinen Änderungen an den Daten ein möglichst unterschiedliches Ergebnis liefern. Dies kann z. B. mit dem Verfahren zur zyklischen Redundanzprüfung (CRC) erreicht werden. Besser geeignet sind kryptographische Hashfunktionen wie der Message Digest Algorithm 5 (MD5-Algorithmus) oder der Secure Hash Algorithm 1 (SHA-1); diese haben die Eigenschaft, daß andere Daten als diejenigen über welche die Prüfsumme berechnet wurden, die denselben Hashwert haben, nicht oder nur schwer zu finden sind.

Es gibt folgende Ansätze für die Verwendung von Prüfsummen:

a) Lizenzierungsinformationen werden in einer externen Datei gespeichert, über diese wird die Prüfsumme berechnet.
b) Für jede externe Bibliothek wird eine Prüfsumme im eigentlichen Programm gespeichert. Bevor die externen Funktionen aufgerufen werden, wird diese dann mit der neu berechneten verglichen und ggf. die weitere Ausführung verweigert.
c) Das Programm (oder ggf. auch eine Bibliothek) selbst hat die eigene Prüfsumme in seinem Maschinencode gespeichert.
d) Červeň beschreibt in [Čer02, Kapitel 8] einen Schutzmechanismus, bei dem beim Programmstart eine Prüfsumme (in diesem Fall ein CRC-Wert) über einen bestimmten Codeblock gebildet wird. Wird dieser Block zur Laufzeit verändert, kann dies mit einer späteren Neuberechnung der Prüfsumme erkannt werden. Um eine solche Maßnahme zu umgehen, müßte die Prüfsumme ebenfalls manipuliert werden.

Alle drei Verfahren haben dasselbe Problem: Die Prüfsumme muß an irgendeiner Stelle gespeichert sein. Spätestens bei der Prüfung ist diese im Speicher und wird verglichen. Möchte nun jemand den Maschinencode eines Programms manipulieren, kann er entweder die Summe am Speicherort mit der neu berechneten überschreiben oder den Code zum Vergleich des Prüfsumme so verändern, daß immer Gleichheit festgestellt wird. Außerdem muß bei einer Checksumme, die in der berechneten Datei liegt, diese Speicherstelle in irgendeiner Form markiert, von der Berechnung ausgenommen und schließlich nach Erstellung der Datei manipuliert werden, damit sie die richtige Prüfsumme enthält, da ein normaler Compiler diese Aufgabe nicht übernimmt. Würde ein Algorithmus verwendet, der es erlaubt, die Prüfsumme so zu berechnen, daß sie selbst Bestandteil der Prüfsumme wird (z. B. bei Summe über alle Bytes einschließlich der Prüfsumme selbst modulo 256 ergibt ein festen Wert), könnte ein Manipulierender diese Schritte nach Änderung des Maschinencodes auch selbst nachvollziehen. Solche Verfahren werden zur Absicherung der Übertragung auf Hardware-Bus-Systemen angewandt.

6.3.4. Starke Verschlüsselung

Viele auf dem Markt erhältliche Produkte verschlüsseln Lizenzierungsinformationen oder Programmcode mit „starker Verschlüsselung“. Dabei werden Techniken wie das nach den Initialen seiner Erfinder benannte asymmetrische Verschlüsselungsverfahren RSA oder das symmetrische Verschlüsselungsverfahren Advanced Encryption Standard (AES) mit hohen Schlüssellängen eingesetzt, die nach heutigem Stand nicht geknackt werden können. Bei symmetrischer Verschlüsselung werden die Daten mit demselben Schlüssel ver- und entschlüsselt, bei asymmetrischer Verschlüsselung werden zur Ver- und Entschlüsselung unterschiedliche Schlüssel verwendet.

Es gibt zwei Ansatzpunkte, derartige Verschlüsselung zu umgehen, ohne die eigentliche Verschlüsselung zu brechen.

a) Ein Benutzer mit gültiger Lizenz wartet, bis der Programmcode im Speicher entschlüsselt ist und erzeugt aus dem Speicherinhalt ein neues Maschinenprogramm. Dieses läßt sich dann auch ohne gültige Lizenz verwenden. Diese Vorgehensweise kann dadurch erschwert werden, daß man immer wieder nur Teilstücke des Codes zur Laufzeit entschlüsselt und auch wieder verschlüsselt.
b) Egal ob symmetrische oder asymmetrische Verschlüsselung verwendet wird, spätestens zum Zeitpunkt der Entschlüsselung ist der notwendige Schlüssel auf dem System bekannt. Durch Analyse des Maschinencodes gilt es, diese Stelle im Programm zu finden und den Ort zu lokalisieren, an dem der Schlüssel gespeichert ist. Selbst wenn der Schlüssel anhand der aktuellen Hardware berechnet wird, kann man entweder den Wert nach erfolgter Berechnung überschreiben oder das verschlüsselte Programm auf einem lizenzierten System anhand des gefundenen Schlüssels entschlüsseln und daraus ein neues Maschinenprogramm generieren. Diese Vorgehensweise läßt sich durch Techniken erschweren, wie sie im Abschnitt 6.3.5 beschrieben sind. Aber selbst wenn die Analyse mit Debuggern, Disassemblern und Traces ausgeschlossen werden kann, läßt sich nicht verhindern, daß jemand den Code mit Hilfe eines klassischen Schreibtischtests analysiert. Der Aufwand wird dann jedoch beträchtlich. Das Kernproblem der Verschlüsselung des Programmcodes ist, daß zu Beginn der Ausführung immer ein unverschlüsselter Code stehen muß, der dann die Entschlüsselung vornimmt. Man kann nur versuchen, diesen Bereich mit Mechanismen wie Polymorphismus (vgl. Abschnitt 6.3.5.2) so stark wie möglich zu verschleiern, so daß der Aufwand für den “Cracker” zu hoch wird.

6.3.5. Anti-Spoofing

6.3.5.1. Verhindern, daß das Programm gedebuggt oder getraced wird

Diese Schutzmaßnahme basiert darauf, daß das Vorhandensein eines Debuggers erkannt wird. Dann kann wahlweise das Programm mit oder ohne Meldung beendet werden oder es können andere Maßnahmen ergriffen werden. Eine Möglichkeit ist es, den Debugger-Prozeß zu beenden. Dies funktioniert allerdings nicht bei sogenannten Kernel-Debuggern, da diese nicht durch Benutzerprogramm beendet werden können. In diesem Fall sorgen manche Schutzprogramme dafür, daß durch illegale Maschinencodes oder Manipulationen an Strukturen des Betriebssystems letzteres zum Absturz gebracht wird.

Pavol Červeň beschreibt in [Čer02, Kapitel 7] diverse Möglichkeiten, um Debugger zu erkennen. Debugger im Applikationsmodus lassen sich i. allg. über die Liste der aktiven Prozesse identifizieren, außerdem auch über das gesetzte Trap Flag. Das Trap Flag wird allerdings nur im Einzelschrittmodus verwendet. Beim Debuggen mit Haltepunkten wird der Code so modifiziert, daß die Anweisung, an der unterbrochen werden soll, durch den Aufruf des Unterbrechungsinterrupts (bei i386: INT 3) überschrieben wird. Bevor das Programm weiter ausgeführt wird, wird hier wieder der Originalcode eingefügt. Haltepunkte können also über das Vorhandensein dieser Interrupts erkannt werden (vgl. [Eil05]).

Die Bedienoberfläche von Kernel-Debuggern wird üblicherweise als normaler Prozeß gestartet. Sie kommuniziert mit dem eigentlichen Debugger über definierte Schnittstelle (wie auch eine normale Applikation über solche Schnittstellen mit den Treibern und letztendlich mit den angeschlossenen Geräten kommuniziert). Die Erkennung dieser Typen von Debuggern erfolgt dadurch, daß das Vorhandensein dieser Schnittstellen geprüft wird.

Mit ähnlichen Mitteln wie Debugger arbeiten Trace-Programme; sie erstellen eine Liste, in welcher Reihenfolge die einzelnen Funktionen oder Befehle aufgerufen worden sind. Červeň beschreibt in [Čer02, Kapitel 8], wie sich ein Programm daraufhin testen läßt, ob es getraced oder gedebuggt wird. Das Programm installiert einen eigenen Exception-Handler für die Trace-Exception. In diesem wird nur eine bestimmte Variable inkrementiert. Nun wird das Trap Flag manuell gesetzt. Ist kein Debugger installiert, wird die eigene Routine aufgerufen und die Variable inkrementiert, läuft ein Debugger, wird dessen Behandlungsroutine aufgerufen, die auf diese Variable keinen Einfluß hat. Anhand des Wertes kann nun bestimmt werden, ob ein Debugger läuft oder nicht.

Der Kernel-Debugger SoftICE der Firma NUMega (vgl. [Com]) wird dabei sowohl von Červeň, als auch den meisten Herstellern von Schutzprogrammen als wichtigster „Feind“ unter den Debuggern gesehen.

Eine weitere Möglichkeit zur Analyse eines Programms sind API Hooks,

d. h. eine Schnittstellen-gleiche API wie diejenige des Betriebssystems, die aber neben der eigentlichen Funktionalität in diesem Fall noch eine Protokollierung vornimmt. Sie lassen sich erkennen, indem Prüfsummen über den Beginn einer solchen API-Funktion auf dem Originalbetriebssystem gebildet werden und zur Laufzeit verglichen werden. Alternativ gibt es auch die Möglichkeit, die ersten Befehle der API-Funktion bereits im eigenen Code aufzurufen und dann ein paar Byte weiter mitten in die Funktion zu springen.[4] Beide Methoden haben den Nachteil, daß sie sehr stark von einer bestimmten Betriebssystemversion abhängig sind. Es muß eine Prüfung auf die aktuelle Version vorgenommen werden und eine entsprechende Tabelle für jede einzelne Version (einschließlich Service Packs und Patches für einzelne Bibliotheken) zur Verfügung stehen. Dieses Problem läßt sich vermeiden, indem die Angreifer mit ihren eigenen Waffen geschlagen werden. Die Applikation erstellt selbst eine eigene Version einer nicht benötigten API-Funktion, die einen unerwarteten Rückgabewert liefert. Ist nun auf dem System ein API Hook vorhanden, würde eine Funktion aufgerufen, die dem Originalverhalten des Betriebssystems entspricht und nicht dem der manipulierten.

6.3.5.2. Verschleierung des Maschinencodes

Maßnahmen, die dazu dienen, den Maschinencode im Speicher schlecht lesbar zu machen, erfüllen gleich mehrere Zwecke:

- Die Arbeit von automatischen Disassemblern wird erschwert; diese können im Idealfall die Grenzen der einzelnen Instruktionen und der Daten nicht mehr unterscheiden.
- Memory Dumps, d. h. Ausgaben des Speichers einer Applikation helfen nicht weiter, da der Maschinencode nicht eindeutig zu identifizieren ist.
- Memory Patches, d. h. das gezielte Überschreiben bestimmter Speicherstellen, um die Funktion zu manipulieren, wird erschwert, wenn der Code unklar ist.
- Die Entwicklung von Keygeneratoren wird erschwert. Vielfach werden diese Programme sogar durch Kopieren des Codes für die Berechnung der Schlüssel aus dem geschützten Programm erstellt.

Folgende Maßnahmen erschweren die Lesbarkeit eines Maschinenprogramms:

- mehrdeutiger Code, d. h. es wird mitten in eine (aus mehreren Bytes bestehende) Maschinenanweisung gesprungen, die dadurch eine neue Semantik erhält. Wird diese Technik mehrfach hintereinander angewandt, ist sie schwer zu durchschauen.

[4] Der Stack muß zu Beginn der kopierten Instruktionen natürlich wie bei einem Funktionsaufruf vorbereitet werden.

- Polymorphismus, d. h. der Maschinencode verändert sich während des Programmablaufs selbst; wird an einer Stelle der Code manipuliert, so kann dies Auswirkungen auf spätere Durchläufe haben.
- Gerade nicht benötigte Codeblöcke werden verschlüsselt und erst bei Bedarf wieder entschlüsselt; diese Abschnitte lassen sich zu diesem Zeitpunkt nicht analysieren.

Wie im Abschnitt 6.3.4 gilt auch hier: Mit genügend großem Aufwand und einem Schreibtischtest lassen sich alle diese Schritte nachvollziehen. Abhilfe schafft hier nur, einen Teil der Verarbeitung auf einem externen Gerät durchzuführen, dessen Verhalten nicht bekannt ist, z. B. auf einem Kopierschutzstecker (vgl. Abschnitt 6.3.1.3).

Etwas schwieriger ist diese Aufgabe, wenn ein ausgeliefertes Programm nicht aus Maschinencode, sondern aus einem vorcompilierten Byte-Code besteht (wie bei Java, Microsoft Visual Basic oder Microsoft .NET). Vielfach sind für die Laufzeitumgebungen bereits Decompiler erstellt oder sie gehören wie der “ILDASM” beim .NET Framework SDK bereits zum Standardlieferumfang. In [Mas05b] wird als einziger Weg die Verschleierung des Codes durch nichtssagende oder falsch bezeichnete oder auch durch zusätzliche sinnlose Variablen- und Funktionsnamen vorgeschlagen.

6.3.5.3. Kundenspezifischer Maschinencode

Es wird ein ein neuer Compiler (ggf. auch Präprozessors) entwickelt, der für jeden Kunden gesteuert über Parameter wie dessen Seriennummer oder Modulkonfiguration jeweils anderen Code generiert (zumindest in Bezug auf Lizenzkontrolle). Auf diese Art und Weise ließe sich auch ein Wasserzeichen (mehr dazu im Abschnitt 6.3.9 weiter unten) des Kunden unterbringen.

6.3.5.4. Verstecken von Schlüsseln

In [Čer02] werden bereits verschiedene Möglichkeiten zum Verstecken von Schlüsseln von kryptographischen Verfahren vorgeschlagen, wie dynamische Bibliotheken (DLLs) im Systemverzeichnis oder in den Daten.

Darüber hinaus bietet sich das Verstecken in Bild- oder anderen Mediendateien mittels Steganographie (weitere Informationen zum Verfahren unter [Wei]) an. Dabei werden Bilder mit den zu versteckenden Informationen überlagert, so daß das Rauschen des Bilds geringfügig zunimmt. Ein Betrachter erkennt den Unterschied, selbst wenn er das Originalbild zum Vergleich hat, nur schwer. Anhand diese Unterschieds wird dann die versteckte Information wieder extrahiert.

Ein weiterer neuer Ansatz ist es, feste Schlüssel so zu wählen, daß sie für einen geübten Leser wie Maschinencode aussehen. Am besten enthält dieser “Maschinencode” dann noch einen Aufruf einer sonst nirgends aufgerufenen Dummy-Funktion, die sehr umfangreich ist und so aussieht, als

ob in ihr umfangreiche Prüfungen vorgenommen würden. Die beiden zuletzt angeführten Methoden stellen einen eigenen Beitrag zu dieser Arbeit dar, da derartige Lösungen bislang nicht zu finden waren.

6.3.5.5. weitere Maßnahmen

Die beiden folgenden Maßnahmen erschweren gewisse Manipulationen am System als Ganzem.

a) Es wird erkannt, daß die Systemzeit zurückgestellt wurde, um die Nutzungsdauer von zeitbeschränkten Demonstrationsversionen zu verlängern. Folgende Maßnahmen sind dabei hilfreich:
   - Es wird geprüft, ob bereits Dateien mit neuerem Zeitstempel als die jetzige Systemzeit auf dem System zu finden sind; alternativ kann diese Prüfung aus Performancegründen auch auf einige wenige Systemdateien beschränkt werden.
   - Die Applikation führt, während sie ausgeführt wird ein Protokoll über verschiedene Zeitpunkte; liegt bei einer neueren Ausführung die Systemzeit vor dem letzten Eintrag im Protokoll, so wurde die Systemzeit zurückgestellt. Als Verbesserung kann darüber hinaus ein ständig laufender Dienst installiert werden, der diese Protokollierung auch dann vornehmen kann, wenn die eigentliche Applikation nicht gestartet ist. Ein solches Verfahren wurde bisher offenbar nicht verwendet oder zumindest so nicht veröffentlicht.
   - Die Zeit wird mit anderen Zeiten im lokalen Netzwerk oder im Internet verglichen.

b) Es wird verhindert, daß das System mit Hilfe eines Images der Festplatte restauriert wird. Dies kann anhand der Zeitstempel der verschiedenen Dateien festgestellt werden.

Diese beiden Maßnahmen bieten zumindest bei zeitbeschränkten Versionen von Software, die auf eine Konsistenz und Richtigkeit der Systemzeit angewiesen sind, einen gewissen Schutz. So läßt sich ein Zeitwirtschaftssystem mit einer manipulierten Systemzeit nicht produktiv betreiben.

6.3.5.6. Kryptoprozessoren und Trusted Computing

Eldad Eilam beschreibt in [Eil05, S. 318 ff und S. 322 ff] Kopierschutzmechanismen, die auf einer veränderten, nicht mehr so offenen, Systemarchitektur basieren. Künftige Systeme erhalten dann jeweils ein Paar aus öffentlichen und privatem Schlüssel, wobei letztere nur im System gespeichert ist und auch über Software nicht ausgelesen werden kann. Ein Endkunde, der eine Software bestellt, würde seinen öffentlichen Schlüssel bei der Bestellung an den Hersteller übermitteln. Dieser verschlüsselt dann die Applikation mit dem öffentlichen Schlüssel und sendet sie an den Endkunden. Wenn

jener diese dann ausführt, sorgt der Prozessor dafür, daß diese nur verschlüsselt im Speicher gehalten wird. Der Programmcode wird zur Laufzeit mit dem privaten Schlüssel entschlüsselt und dann ausgeführt. So hätte der Benutzer selbst, wenn er mit allen Systemprivilegien ausgestattet wäre, keine Möglichkeit mehr, die Applikation unverschlüsselt zu sehen. Allerdings gibt es bereits Forschungsarbeiten, bei denen der private Schlüssel über eine Untersuchung den Stromverbrauch des Chips ermittelt werden konnte. Mit diesem Schlüssel könnte die Applikation dann dekodiert und verbreitet werden. Neben der mangelnden Verbreitung solcher Systeme steht dieser Vorgehensweise auch ein großes Mißtrauens seitens der Nutzer entgegen, da viele befürchten, daß sie die Kontrolle über ihr System an ein Kartell von Industrie-Konzernen abgeben (vgl. [Hei03b]). Auch werden Sicherheitrisiken befürchtet, wenn der Benutzer nicht mehr erkennen und kontrollieren kann, was auf seinem System abläuft, denn auch ein Virenscanner könnte derart verschlüsselte Programme nicht mehr kontrollieren. Außerdem ist die Sicherheit von Transaktionen dadurch gefährdet, da die Hersteller der Systeme den privaten Schlüssel kennen würden.

6.3.6. Code zurückhalten

Code für Module, die nicht freigeschaltet sind, wird (z. B. in Form einer DLL) erst bei Freischaltung ausgeliefert. Alternativ kann auch nur ein wesentlicher Bestandteil des Moduls entfernt werden, um die Übertragung zu beschleunigen. Außerdem kann dieser Code auch auf einem Kopierschutzstecker gespeichert werden. Ein Vorteil dieser Vorgehensweise ist, daß Personen, die nur die beschränkte Version zur Verfügung haben, daraus auch mit Manipulationen am Maschinencode keine Vollversion erstellen können. Das Verfahren ist allerdings nicht geeignet, wenn Testversionen mit vollem Leistungsumfang und reinen Zeitbeschränkungen vertrieben werden sollen.

6.3.7. Online-Lizenzprüfungen

Online-Lizenzprüfung setzt eine Netzwerkverbindung zwischen dem Endkunden und dem Hersteller voraus. I. allg. wird diese über das Internet aufgebaut. Die beiden folgenden Wege bieten sich an:

- Die Verbindung wird über ein transportorientiertes Protokoll wie TCP aufgebaut. Ggf. läßt sich der Datenverkehr verschlüsseln. Ein Vorteil dieser Lösung ist, daß sich die Funktionalität leichter vor dem Nutzer verbergen läßt. Allerdings sind solche Verbindungen in vielen Firmennetzen so nicht zulässig.
- Die Verbindung wird über das Protokoll HTTP aufgebaut. Bei Bedarf kann dies Übertragung noch mit Secury Sockets Layer (SSL) verschlüsselt werden; dies wird dann als HTTPS bezeichnet. Dieser Weg ist in

den meisten Netzwerkumgebungen möglich, auch die erzwungene Nutzung eines Proxy-Servers schadet hier nicht. Außerdem sind die Protokolle standardisiert und die Implementierung ist vielfach getestet. Ein Nachteil dieser Vorgehensweise ist, daß der Nutzer den gesamten Datenverkehr protokollieren und analysieren kann. Bei einfach gewählten Daten läßt sich dann leicht ein eigener Webserver so einrichten, daß er dieses Protokoll nachbildet und die Applikation ohne entsprechende Lizenz nutzbar ist.

6.3.8. Viele Updates

Bei Programmen, deren unlizenzierte Nutzung durch Manipulation der Binärdateien ermöglicht wird, ist es ratsam, für diese recht oft neuere Versionen freizugeben. Bei jeder neuen Version kann dann der Kopierschutz modifiziert werden. Dies hat zur Folge, daß die “Cracker” diesen Schutz erneut komplett analysieren müssen.

Ein Hersteller könnte auch bewußt Fehler in das Programm einbauen, um die Nutzer zu Updates zu zwingen. Dabei würden die Nutzer unlizenzierter Version die Software bis zur neuerlichen Analyse des Schutzes nicht mehr nutzen zu können. Neben solchen fest eingebauten Fehlern wäre es auch möglich dafür zu sorgen, daß der absichtlich eingebaute Fehler erst nach einer gewissen Zeit auftritt; diese Vorgehensweise wird auch als “Timebombing” bezeichnet. Treten diese Fehler erst dann auf, wenn sich der Hersteller sicher sein kann, daß alle lizenzierten Benutzer die neuere Version bereits eingespielt haben, so würden diese nicht in Mitleidenschaft gezogen. In der Praxis läßt sich dieser Zeitpunkt im Voraus allerdings nicht bestimmen. Außerdem muß festgestellt werden, daß das absichtliche Integrieren von Fehlfunktionen in der Software in der Beziehung zum Kunden, der eine gültige Lizenz erworben hat, durchaus als Betrug angesehen werden kann, da der Hersteller mangelhafte Ware liefert und dies dem Kunden verschweigt, obwohl er Kenntnis davon hat.

Nachteile dieser Vorgehensweise sind der hohe Aufwand für die häufige Freigabe von Software, für die jedesmal auch alle vorgesehenen Tests durchgeführt werden sollten. Auch beim Kunde wird zumindest die Arbeitszeit für die Prüfung und Installation der neuen Version benötigt; dazu kommen im Fehlerfall noch dadurch bedingte Aufwände.

Der häufige Wechsel des Lizenzschutzes ohne die bewußte Integration von Fehlern eignet sich gut für Systeme, die ohnehin regelmäßig upgedatet werden müssen, um ihre Funktion zu erfüllen. Dazu gehören Programme wie Virenscanner, die üblicherweise täglich auf den neusten Stand gebracht werden, oder Lohnbuchhaltungssoftware, die jährlich um geänderte gesetzliche Grundlagen und Tabellen ergänzt werden muß.

6.3.9. Ergebnisse sind mit Daten des Kunden markiert

Die Arbeitsergebnisse sind mit gewissen unveränderlichen Daten des Kunden fest verbunden. Ein Nutzer, der mit der Lizenz eines anderen arbeitet, würde seine Ergebnisse also unter anderem Namen verbreiten.

Diese Vorgehensweise läßt sich beispielsweise beim einem Fax-Programm nutzen, das als Absenderkennung immer die Kennung des Absenders verschickt, welche fest in den Lizenzierungsinformationen eingebettet ist oder bei einem CAD-Programm, wobei die damit erstellten Zeichnung als Ersteller den in der Lizenz angegebenen Namen enthält. Ein Nachteil dieser Lösung für den Endkunden ist, daß bei einer Umfirmierung oder einem Wechsel des Sachbearbeiter immer die Lizenzierunginformationen durch den Hersteller geändert werden müssen.

Alternativ läßt sich auch das Maschinenprogramm mit einem Wasserzeichen mit den Informationen des jeweiligen Kunden versehen. Dieses kann vom Compiler mit erzeugt werden oder im Nachhinein aufgebracht werden (vgl. dazu Ansätze in [Eil05, S. 321 f]).

6.3.10. Vorgehensweise bei Erkennung unlizenzierter Nutzung

Erkennt eine geschützte Software, daß sie ohne gültige Lizenz genutzt wird, gibt es verschiedene Möglichkeiten, darauf zu reagieren.

a) Es wird eine Meldung anzeigt und das Programm wird beendet. Nachteil dieser Lösung ist, daß die Funktion, welche diese Meldung ausgibt, relativ leicht im Maschinencode lokalisiert werden kann und dem “Crakker” einen Ansatz für die Umgehung des Schutzes liefert.

b) Das Programm wird sofort und ohne Meldung beendet. Alternativ kann auch Code aufgerufen werden, der das System zum Absturz bringt. Im Maschinencode sind solche Stellen u. U. schwer zu finden, mit einem Debugger ist es dagegen eher einfach. Um anhand des Stacks der Applikation bei der Beendigung keinen Rückschluß auf den Ort und die Funktion der Lizenzprüfung zu erhalten, gibt es auf manchen Systemen die Möglichkeit, den Inhalt des Stacks irgendwo am Anfang des Programms zu speichern, danach die eigentliche Lizenzprüfung durchzuführen und, falls diese fehlschlägt, die entsprechende Funktion zur Restaurierung des Stacks aufzurufen. Ab diesem Zeitpunkt kann der Debugger anhand des Stacks nicht mehr erkennen, welche Funktion den Anlaß zur Beendigung des Programms gab. In der Programmiersprache C können beispielsweise die Funktion setjmp und longjmp (vgl. [Her99, Kapitel 8: Nicht-lokale Sprünge]) dazu verwendet werden. Die Verwendung dieser Funktionen wird in der Literatur ausführlich beschrieben und auch oft in Softwareprojekten angewandt; die Nutzung dieser Funktionen um ein Programm immer Falle nicht-lizenzierter Nutzung so zu

beenden, daß sich bei Nutzung eines Debuggers die Ursache nicht mehr erkennen läßt ist ein eigener Beitrag in dieser Arbeit.

c) Das Programm wird erst mit zeitlicher Verzögerung beendet. Eine zeitliche Verzögerung hat den Nachteil, daß das Programm eine gewisse Zeit ohne gültige Lizenz verwendet werden kann. Sie hat auch Vorteile, wie daß der Zusammenhang zwischen der Lizenzprüfungsfunktion und dem Abbruch nur schwer zu finden ist. Außerdem kann auf diese Art und Weise an mehreren Stellen Lizenzschutz integriert werden. Wird die erste Prüfung umgangen, meint der “Cracker”, er sei bereits am Ziel und verbreitet diese Version weiter. Wenn dann im Laufe der Zeit immer wieder neue Prüfungen durchgeführt werden, müssen diese erst mühsam wieder einzeln gefunden werden. Würden allerdings alle Prüfungen ohne zeitliche Verzögerung durchgeführt, so könnte man diese der Reihe nach durcharbeiten und wäre dann sicher, eine komplett lauffähige Version erstellt zu haben.
d) Das Programm funktioniert zunächst einmal weiterhin wie gewünscht. Nach einiger Zeit werden aber immer wieder bestimmte Fehler angezeigt, die zu einem Abbruch des Programms und ggf. auch zum Verlust von Daten führen. Erkundigt sich ein Nutzer beim Hersteller oder in öffentlichen Foren nach einer Fehlerbehebung für diesen speziellen Fehler, enttarnt er sich damit als unlizenzierter Nutzer des Programms und setzt sich strafrechtlicher Verfolgung oder Schadensersatzansprüchen des Herstellers aus.
e) Eine Variante zu d) ist es, keinen echten Fehler zu erzeugen, sondern das Verhalten des Programms zu verändern. So hat beispielsweise ein Spielehersteller einmal die Figuren in seinem Spiel bei unlizenzierten Versionen im Laufe der Zeit optisch verändert. Viele Nutzer haben dann in einschlägigen Foren oder sogar beim Hersteller nach dem Sinn dieser Veränderung nachgefragt.

Maßnahmen, die erst mit einer gewissen Verzögerung durchgeführt werden (wie in c, d oder e), erfordern einen gewissen Aufwand in der Implementierung und müssen beim Design des Systems berücksichtigt werden. Sofort wirkende Maßnahmen lassen sich im Zusammenhang mit Kopierschutz leicht integrieren und können auch bei Schutzmechanismen verwendet, die auf bereits existierende Binärprogramme angewandt werden und dabei beispielsweise den Maschinencode des Ursprungsprogramms verschlüsseln und in ein neues Programm einpacken, welches auch die Prüfung auf das Vorhandensein der Lizenz vornimmt.

6.3.11. Lizenzvergabe

Bei Systemen, die Lizenzinformationen prüfen, können diese auf verschiedenen Wege zum Kunden gelangen. Entsprechendes gilt auch für bei der Produktaktivierung. Diese Wege sind:

6.3.11.1. Freischaltung über Lizenzschlüssel oder -dateien

Die Lizenzinformationen werden entweder in Form einer Datei oder einer Schlüssel-Zeichenkette vom Hersteller zum Kunden versandt. Lizenzdateien können entweder auf Datenträgern gespeichert werden, die dann zum Kunden verschickt werden, oder sie werden auf dem elektronischen Wege, z. B. per E-Mail oder über das Internet versandt.

Diese Möglichkeiten gibt es auch für Lizenzschlüssel. Da manche Firmen die Nutzung von Datei-Anhängen in E-Mails auf bekannte Typen beschränken oder ganz verbieten, bietet sich ein im Klartext lesbarer Lizenzschlüssel auch für diesen Fall an. Lizenzschlüssel können darüberhinaus auch auf Papier, per Telefon oder per Telefax weitergeben werden und sind somit flexibler in ihrer Weitergabe. Ein Nachteil ist, daß die Lizenzschlüssel im Gegensatz zu Lizenzdateien nicht zu lange werden dürfen, damit ein Nutzer diese Daten in angemessener Zeit fehlerfrei eingeben kann. Werden aber aufgrund komplexer Software mit vielen Modulen, Hardwarekoppelung und in den Lizenzierungsinformationen hinterlegtem Kundenname eigentlich längere Daten benötigt, kann dies folgendermaßen gelöst werden. Der Kunde kann anhand der ihm bekannten Daten die Lizenzierungsdatei selbst erstellen. Er gibt diese Daten an den Hersteller weiter und dieser berechnet daraus einen Schlüsselwert. Das Programm prüft mit diesem Schlüsselwert die Plausibilität der Lizenzierungsdatei und wendet deren Eintragungen dann an. Zur Berechnung des Schlüsselwerts wird dann eine Prüfsumme oder ein Hash-Wert verwendet. Nachteil dieses Verfahrens ist, daß sobald jemand die Berechnung der Prüfsumme herausgefunden hat, beliebige Lizenzierungsinformationen erzeugt werden können. Wer schon einmal im Internet nach Seriennummerngeneratoren (auch als “Keygen” bezeichnet) sucht, kann sich davon überzeugen, daß dies bei vielen Programmen so schon geschehen ist (vgl. die 10 Millionen gefundenen Einträge bei Google [Goob]).

6.3.11.2. Online-Freischaltung

Das Programm wird freigeschaltet, in dem über das Internet eine Verbindung zu einem Server des Herstellers aufgenommen wird. Wird dabei das HTTP-Protokoll verwendet, so wird die Server-Funktionalität über das Common Gateway Interface (CGI) oder eine integrierte Skriptsprache implementiert. Dabei muß geprüft werden, ob für das Kontakt aufnehmende System vom Kunden eine Lizenz erworben wurde. Dazu werden die übermittelten

Daten mit einer Datenbank verglichen, wobei kontrolliert wird, ob vom angegebenen Kunde bereits eine Zahlung erfolgt ist und er nicht bereits schon einmal eine Online-Freischaltung vorgenommen hat.

6.3.11.3. Freischaltung durch Service-Techniker

Möchte ein Service-Techniker des Herstellers für einen Kunden eine neue Lizenz vergeben, kann er im einfachsten Fall diese Lizenzinformation vor Ort komplett neu erstellen. Der Hersteller sollte eine große Verbreitung der Lizenzerstellungsinfrastruktur vermeiden, da jemand, der zur ihr Zugang erhält, für das System gültige Lizenzinformationen erstellen kann, ohne eine Lizenz erworben zu haben. Im Gegensatz dazu kann es Sinn machen, wenn der Techniker vor Ort temporär Module freischalten kann, bis die angepaßten Lizenzierungsinformationen zum Hersteller übermittelt und berechnet sind. Diese Vorgehensweise kennt keines der untersuchten Produkte.

Sollen nur Erweiterungen vorgenommen werden, kann der Techniker vom Hersteller entweder die veränderte Lizenzdatei oder einen Freischaltschlüssel (s. o.) mitbringen und auf dem Kundensystem einspielen.

6.3.12. Übertragung einer Lizenz auf ein anderes System

Manchmal ist es notwendig, eine an einen Rechner gekoppelte Lizenz auf ein anderes System zu übertragen. Dies kann beispielsweise bei defekter Hardware oder dem Umstieg auf ein leistungsfähigeres System notwendig werden. In diesem Zug soll das Programm nicht mehr auf dem bisherigen und nur noch auf dem neuen System nutzbar sein. Je nach Art der Lizenzprüfung gibt es hier verschiedene Vorgehensweisen:

6.3.12.1. Deaktivierung und Neu-Aktivierung

Das Programm wird auf dem bisherigen System deinstalliert. Bei der Deinstallation wird die Hardwarekonfiguration des neuen Systems eingegeben und es werden Lizenzierungsinformationen für das neue System generiert. Diese Vorgehensweise hat mehrere Nachteile: Mittels einer Datensicherung läßt sich das bisherige System wieder auf den vorhergehenden Stand bringen und weiter verwenden. Außerdem setzt dieses Verfahren voraus, daß das bisherige System noch funktionstüchtig ist. Sofern dies nicht der Fall sein sollte (z. B. bei Hardwaredefekten), ist ein Umzug der Lizenz nicht mehr möglich.

6.3.12.2. Lizenzierung über Datenträger oder Kopierschutzstecker

Wird ein System über einen Datenträger oder einen Dongle abgesichert, so

wird dieses Medium einfach aus dem bisherigen System entfernt und mit dem neuen verbunden. Dann ist das neue System sofort nutzbar.

Werden beschreibbare Datenträger eingesetzt, so können diese auch bei der Deinstallation des alten Systems markiert werden, so daß eine erneute Installation nur möglich ist, wenn diese Markierung vorhanden ist. Dies setzt allerdings voraus, daß das alte System noch funktioniert, wodurch diese Vorgehensweise bei Hardwareausfällen nicht praktikabel ist.

6.3.12.3. Umzug mittels einer direkten Netzwerkverbindung

Auf dem neuen System wird die Software installiert. Sie enthält eine Funktion, die eine Verbindung zum bisherigen System aufbaut und die Lizenzierungsinformationen überträgt. Danach wird das alte System deaktiviert und nach erfolgreicher Deinstallation das neue aktiviert. Auch dieses Verfahren ist bei defekter Hardware nicht mehr durchführbar. Außerdem könnte das bisherige System mit einem geeigneten Festplattenimage auf den vorherigen Stand zurückgebracht werden und der Kunde hätte zwei lauffähige Systeme.

Ein nur teilweise sicherer Ansatz sieht folgendermaßen aus: Jedes System modifiziert die Lizenzierungsinformationen in Abhängigkeit von der Systemzeit. Wird nun die Übertragungsfunktion aufgerufen und das bisherige System über ein Festplattenimage zurückgesetzt, kann festgestellt werden, daß, wenn das Image zu alt ist, die Lizenz bereit erloschen ist. Diese Vorgehensweise hat aber mehrere Nachteile:

- Der Kunde kann keine Sicherungskopien seiner Lizenzinformationen erstellen, die nach einer gewissen Zeit noch funktionsfähig sind.
- Wird das Image unmittelbar vor dem Übertragungsvorgang erstellt, so läßt es sich dennoch weiterverwenden.
- Wird ein System längere Zeit nicht genutzt, fehlt für diesen Zeitraum die notwendige Modifikation der Lizenzinformationen und das System ist beim nächsten Start nicht mehr nutzbar.

6.3.12.4. Erkennung eines Zweitsystems im Netzwerk

Praktikabel ist es daher, einen Umzug im Netzwerk zuzulassen und dann im laufenden Betrieb zu prüfen, ob ein weiteres System mit derselben Lizenz im Netzwerk gefunden wird. Danach würden dann beide Systeme deaktiviert. Diese Prüfung läßt sich leicht umgehen, wenn zwischen dem lizenzierten und dem nicht-lizenzierten System keine Netzwerkverbindung besteht. Daher ist dieses Verfahren nur anzuwenden, wenn aus anderen Gründen (wie z. B. mit dem Namen des Kunden markierten Arbeitsergebnissen) sichergestellt wird, daß nur ein Kunde diese Lizenz nutzen kann und das Programm ohne eine Verbindung zum lokalen Netzwerk keinen Nutzen hat.

6.3.12.5. Umzug über Online-Verbindung

Bei Programmen, welche zur Lizenzprüfung eine Online-Verbindung nutzen, ist ein Transfer der Lizenz auf ein anderes System einfacher. Das bisherige System teilt bei der Lizenzprüfung dem Server die Daten des neuen Systems mit und beendet sich. Wird trotzdem noch einmal das bisherige System aufgerufen, so lehnt der Lizenzprüfungsdienst auf dem Server dieses ab. Jetzt wird nur noch das neue System freigeschaltet.

6.3.13. Schritte nach Fertigstellung der Software

Jeder auch noch so ausgeklügelte Kopierschutz bewahrt den Hersteller nicht davor, sich während des gesamten Lebenszyklus des Produkts darüber zu informieren, ob und wie der Schutz umgangen wird. Wer möchte, daß seine Software nur von denjenigen Personen genutzt wird, welche sie auch lizenziert haben, sollte in seinen Entwicklungsplänen nicht nur Kapazitäten für nachträgliche Fehlerbehebungen, sondern auch für Recherchen nach “Cracks” und Anpassungen des Lizenzschutzes berücksichtigen.

7. Entwicklung und flexible Lizenzierung modularer und erweiterbarer Software

Dieser Abschnitt macht Vorschläge zur Modellierung komplexer modularer und erweiterbarer Software und stellt ein Modell für die flexible Lizenzierung modularer Software vor. Dies ist ein eigener, neuer Beitrag dieser Arbeit.

7.1. Entwicklung modularer Software

Im Abschnitt 3.8 wurde bereits dargelegt, wie modulare Software modelliert werden kann; dabei wurde auch die Implementierung mittels aspektorientierter Programmiersprachen vorgeschlagen. Dieser Abschnitt beschäftigt sich mit konkreten Maßnahmen bei der Implementierung.

Die Modellierung modularer Software in Klassen und Objekten ist heute weit akzeptiertes Vorgehen. Die Lizenzprüfung kann dann als Aspekt implementiert werden. In Programmiersprachen, die Aspekte bereits unterstützen, ist die Einbindung eines solchen trivial. In herkömmlichen objektorientierten Sprachen bietet sich die Verwendung von Interfaces an.

Für jedes komplexe Produkt wird eine geeignete Beschreibung benötigt, welche die Beziehungen und Abhängigkeiten zwischen den Modulen darlegt. Dies kann in Form eines Feature Diagramms (vgl. [CE00, Abschnitt 4.4.1] und Abschnitt 3.8) geschehen, wobei anstelle der Feature die einzelnen Module treten. In diesem Zug werden dann auch optionale, alternative und Oder-Features definiert. Insbesondere für die Lizenzierung von Erweiterungen an Dritthersteller werden auch alle Module betreffende Features (“common features”) benötigt, um gewisse Arten von Schnittstellen, wie das Recht, Berichte zu erzeugen, zu lizenzieren.

Modulare und erweiterbare Software kann dazu führen, daß Module aus verschiedenen Quellen zu einem Gesamtprodukt zusammengefügt werden. Es läßt sich oft nicht vermeiden, daß sich dabei die Schnittstellen zwischen den Modulen bzw. zwischen dem Gesamtprodukt des Herstellers und den Erweiterungen von Drittherstellern ändern. Selbst wenn sich die Schnittstellen als solche nicht ändern, können Seiteneffekte und Fehler, die auf die Kernversion keine Auswirkungen haben, in den Erweiterungen von Drittherstellern dazu führen, daß sich das Verhalten ändert.

Das nötige Konfigurationsmanagement läßt sich mittels unterschiedlicher Ansätze durchführen:

- **Einheit aller Versionen:**
  Der Hersteller erstellt und testet eine Version des Systems mit allen Modulen, wobei alle möglichen Kombinationen von Modulen im Test berücksichtigt werden. Die Dritthersteller erhalten die neue Version mit

allen Änderungen und der notwendigen Dokumentation, passen ihre Erweiterungen darauf an und testen die Gesamtlösung. Da die Endkunden, welche ein Produkt des Drittherstellers nutzen, grundsätzlich auch beliebige gültige Kombinationen von Modulen lizenziert haben können, ist der Test beim Dritthersteller mindestens so komplex wie derjenige des Herstellers. Vorteil dieser Lösung ist, daß alle Module sicher eingesetzt werden können, ohne daß im größeren Rahmen Fehlfunktionen zu erwarten sind. Nachteil ist der beträchtliche Testaufwand, der v. a. Hersteller kleinerer Erweiterungen überfordern würde.

- **Haupt- und Nebenversionen:**
  Der Hersteller unterscheidet seine Versionen in Haupt- und Nebenversionen, wobei innerhalb einer Hauptversion die Schnittstellen unverändert bleiben. Dadurch ist es möglich, innerhalb einer Hauptversion die Module unterschiedlicher Nebenversionen zu kombinieren. Ein Dritthersteller kann seine Erweiterungen mit dem ersten Stand einer Hauptversion entwickeln und testen und ist sicher, daß die Schnittstellen sich in den künftigen Nebenversionen gleich verhalten. Vorteil dieser Lösung ist ein deutlich geringerer Testaufwand, von Nachteil ist eine gewisse Unsicherheit, wenn trotz gleicher Schnittstellen durch unterschiedliche Implementierungen und Fehler Seiteneffekte auftreffen.
- **Projektbezogenes Management:**
  Für jeden Endkunden und die von ihm gewünschte Konfiguration wird eine Version speziell zusammengestellt und mit diesen Merkmalen getestet. Dabei wird die Software von Drittherstellern mit einbezogen. Diese Lösung bringt die höchste Sicherheit, wobei der Aufwand nur dann praktikabel ist, wenn der Hersteller relativ wenige Kunden hat, die ziemlich große Systeme erwerben und lizenzieren.

Bei allen diesen Methoden muß beispielsweise durch Quellcodeverwaltungssysteme sichergestellt sein, daß die Versionen aller Module markiert werden und nachvollziehbar sind. Auf geeignete Maßnahmen zum Schutz vor unlizenzierter Nutzung wurde bereits auf den vorherigen Seiten eingegangen.

7.2. Modellierung und Veränderung von Lizenzierungsinformationen

7.2.1. Endbenutzermodule

Im folgenden wird o. E. d. A. davon ausgegangen, daß Lizenzierungsinformationen in einer Datei gespeichert werden (dieselben Inhalte lassen sich beispielsweise auch in Schlüsselzeichenketten codieren).

Die Lizenzierungsinformationen werden am sinnvollsten in mehrere eigen-

ständige Blöcke aufgeteilt. Dazu gehören zu Beginn ein allgemeiner Block mit kundenspezifischen Informationen (im folgenden als Kundenblock bezeichnet) wie dem Kundennamen oder einer Seriennummer. Auch Informationen über eine Koppelung an eine konkrete Hardware fallen darunter. Danach wird für jedes Modul ein eigener Block erstellt, unabhängig davon, ob es auch lizenziert wurde oder nicht. Nicht-lizenzierte Module sind anhand des Inhalts ihres zugehörigen Blocks erkennbar. Jeder Block erhält die Information, ob das zugehörige Modul lizenziert wurde und in welchem Umfang (z. B. lizenziert für bis zu 10 Benutzer gleichzeitig). Abbildung 3 stellt diese Aufteilung dar.

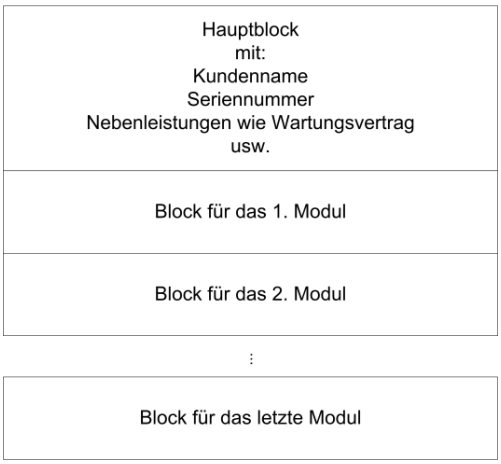

Abbildung 3: Blöcke in einer Lizenzierungsdatei

Jeder Block enthält folgende Informationen:

- eine eindeutige Identifikation des Moduls (i. d. R. Name oder Nummer),
- einen Unterabschnitt, der angibt, ob das Modul dauerhaft freigeschaltet ist; in diesem Bereich werden dann auch Informationen über die zeitbeschränkte Freischaltung für den Evaluierungsbetrieb gespeichert (wie Ablaufdatum oder Anzahl der noch möglichen Nutzungen),
- ggf. einen Abschnitt für weitere Beschränkungen des Moduls (z. B. maximal 100 MB Datenbankgröße),
- optional: Informationen darüber, inwieweit diese Lizenz auch nach einem Versionsupdate noch Bestand haben soll,
- Prüfsummen, welche die Integrität der Daten garantieren sollen; dazu werden mittels gängiger kryptographische Verfahren (vgl. Abschnitt 6.3.4) die Daten verschlüsselt und Hashwerte darüber gebildet. Die Prüfsumme eines jeden Abschnitts muß aber auch Daten aus dem Kundenblock enthalten, damit er nicht bei anderen Kunden genutzt werden kann. Schließlich sollte noch ein geheimer Schlüssel des Herstellers mit eingerechnet werden, um die Nachbildung dieser Dateien zu erschwe-

ren. Kennt ein Hersteller mehrere dieser Schlüssel, die dann zufällig einsetzt werden, ist dies eine weitere Hürde für eine Analyse.
- ggf. zusätzliche Fülldaten, die eine Analyse durch “Cracker” erschweren sollen.

Veränderliche Daten wie die Anzahl der noch übrigen Nutzungen bei Evaluierungsversionen dürfen nicht Bestandteil der normalen Prüfsumme sein; diese enthält nur den separat gespeicherten Startwert. Diese Daten benötigen eigenen Prüfsummen, die von der Applikation immer wieder neu berechnet werden.

Der beschriebene Aufbau eines Blocks wird in Abbildung 4 dargestellt. Die Anordnung der einzelnen Abschnitte ist dabei frei. Wer möglichst großen Schutz vor unlizenzierter Nutzung benötigt, wird dafür sorgen, daß die Daten möglichst schlecht lesbar sind. Spielt dies keine so große Rolle, können die Daten beispielsweise auch in einer XML-Struktur gespeichert werden. Diese hat den Vorteil, daß sie leicht um neue Merkmale erweitert werden kann und so ältere Versionen einer Software u. U. auch mit neueren Lizenzierungsdateien arbeiten können. Die Prüfsummen werden dann in eigenen Inhalten oder Attributen gespeichert.

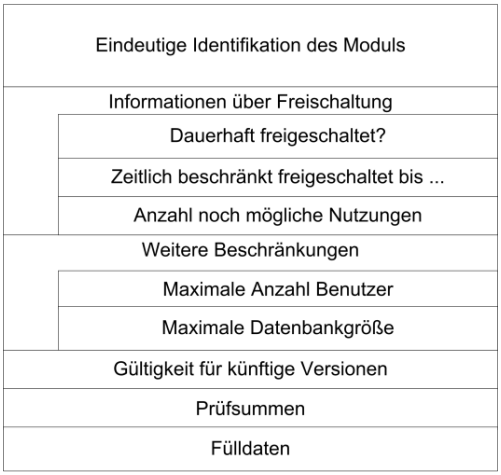

Abbildung 4: Aufbau eines einzelnen Modulblocks

Diese selbständigen Blöcke für jedes Modul haben den Vorteil, daß sie beim Zukauf eines weiteren Moduls auch einzeln freigeschaltet werden können, ohne daß die Lizenzierungsinformation als ganze ausgetauscht werden muß. Zu diesem Zweck berechnet der Hersteller aus den ihm bekannten Kundendaten und den bisherigen Inhalten des Blocks eine Differenz. Diese wird in einen alphanumerischen Code umcodiert und an den Kunden versandt. Ein im Softwaresystem enthaltenes Dienstprogramm berechnet aus dem bisherigen Block und dem alphanumerischen Code dann den neuen Block. Das Dienstprogramm kann auch so erweitert werden, daß es den

Kunden bei der Bestellung unterstützt, indem es die notwendigen Daten für diese aufbereit und – falls möglich – an den Hersteller verschickt. Dieser Ablauf ist in Abbildung 5 nochmals verdeutlicht.

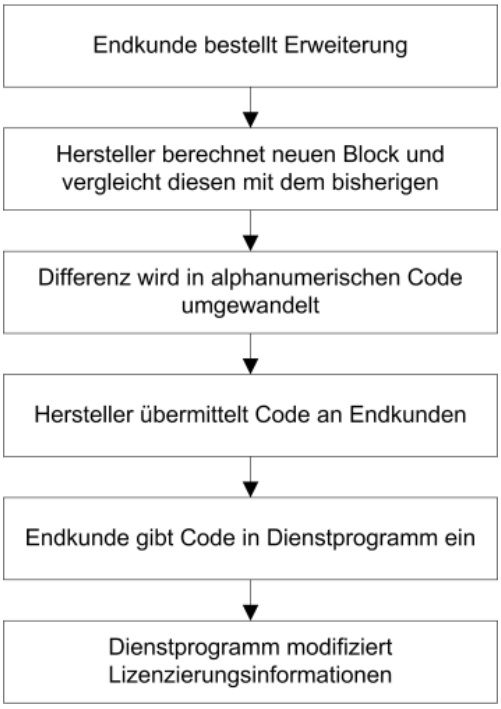

Abbildung 5: Freischaltung eines einzelnen Moduls

Einzelne der bis hierher beschriebenen Funktionalitäten (wie z. B. die Möglichkeit, die Anzahl der Benutzer zu beschränken) können dann weggelassen werden, wenn sie in einer bestimmten Applikation nicht benötigt werden, ohne daß die Gesamtfunktion dieser Art von Lizenzierungsinformationen beeinträchtigt wird.

Dieser Aufbau der Lizenzierungsinformation erlaubt es auch, sie in mehrere Dateien aufzuteilen. Auf diese Art und Weise könnten dann Erweiterungen auch über zusätzliche Dateien realisiert werden. Dabei ist zu beachten, daß alle diese Dateien denselben Hauptblock enthalten müssen, da nur dieser es erlaubt festzustellen, ob die in der Datei enthaltenen Module auch tatsächlich an denjenigen Endkunden lizenziert sind, der sie verwendet. Sind Modulblöcke mehrfach vorhanden, müßte festgelegt werden, ob der weitestgehende bewertet wird oder ob diese aufsummiert werden (z. B. ein Block mit zehn Benutzern, ein zweiter mit fünf Benutzern, kann entweder fünfzehn oder zehn Benutzer im Ergebnis ergeben).

Selbst wenn die Lizenzierungsinformationen einen Block bislang noch nicht enthalten (weil diese Funktionalität beispielsweise erst in einer neueren Version implementiert wurde), ist eine Freischaltung über alphanumerische Co-

des wie folgt möglich: Das Dienstprogramm erstellt selbständig einen Standardblock, der das Modul sperrt. Danach kann wie oben beschrieben der Hersteller einen Differenzschlüssel erstellen und an den Endkunden liefern.

7.2.2. Vertrieb von Mengenlizenzen an Händler

Werden Lizenzen in Mengen (vgl. Abschnitt 5.4.2.2) an Händler, Vertriebsniederlassungen oder OEMs vertrieben, so enthalten diese Lizenzen zunächst keine Daten des Kunden. Die Lizenzierungsinformationen müssen von einem Dienstprogramm auf den Endkunden angepaßt werden. Ggf. kann der Händler auch Lizenzen für bestimmte Module in Lizenzen für andere Module desselben Preises umwandeln.

Bei all diesen Vorgängen ist sicherzustellen, daß eine Lizenz nur einmal vergeben wird. Bei Dateien, die auf einem Datenträger gespeichert sind, ist dies nur schwer möglich, da der Händler jederzeit eine Sicherungskopie zurückspielen kann. In diesem Falle wäre es allerdings möglich festzuschreiben, daß der Kunde das Modul nach Freischaltung durch den Händler zusätzlich beim Hersteller aktivieren muß, bevor es wirksam wird. Käme eine Lizenz bei einem anderen Kunden erneut zum Einsatz, würde die Aktivierung verweigert. Eine weitere Alternative wäre die Nutzung von speziellen externen Geräten (vergleichbar mit Kopierschutzsteckern), welche mittels zugekaufter Lizenzen upgedatet werden können und die dann jeweils kundenspezifische Lizenzen erstellen. Diese spezifische Lizenz kann dann zwar vom Händler beliebig kopiert werden, ist jedoch an einen Endkunden (und ggf. dessen Hardware) gebunden. Anstelle dieses externen Geräts kann auch ein Webserver des Herstellers treten, wobei in diesem Fall wohl eher die Lizenzen nicht in Mengen verkauft, sondern nach Abruf in Rechnung gestellt werden.

7.2.3. Module für Dritthersteller

Die Implementierung der Lizenzen für Erweiterungen von Dritthersteller hängen vom gewählten Vertriebsmodell ab. Werden Generallizenzen vergeben, genügt es, wenn diese parallel zu den normalen Lizenzierungsinformationen geprüft werden, wobei diese Generallizenzen nicht die spezifischen Daten des Endkunden, sondern diejenigen des Drittherstellers enthalten. Wird das Recht für diese Erweiterung direkt vom Hersteller an den Endkunden versandt, kann es vom Ablauf her wie ein Modul des Herstellers behandelt werden.

Ein Dritthersteller kann das oben beschriebene Prinzip für sich übernehmen und eigene Blöcke mit eigenen Schlüsselwerten nutzen, um seine Module zu schützen.

8. Abschluß, Bewertung und Ausblick

Die Lizenzierung modularer und erweiterbarer Software erfordert je nach Produkt, Markt und technischen Möglichkeiten angepaßte Vorgehensweisen. Daher kann eine solche Arbeit kein Kochrezept formulieren, das einem Entwickler vorgelegt wird und diesen über einen starren Prozeß zur idealen Lizenzierung seiner Produkte führt.

Diese Arbeit beschreibt Gründe für die Erstellung modularer Software und bietet auf diese Art und Weise Kriterien für Entwickler, die sich mit der Frage beschäftigen, ob ihr Produkt auch modular gestaltet werden soll. Sie stellt Lösungsansätze vor, wie modulare und erweiterbare Software realisiert und weiterentwickelt werden kann. Außerdem beschreibt sie die Bedeutung der Lizenzierung von Software und nennt Gründe dafür, Software gegen unlizenzierte Nutzung zu schützen. Dabei werden Kriterien beschrieben, die einem Entwickler oder Produktmanager helfen, festzusetzen, in welchem Umfang und in welcher Qualität Schutzmaßnahmen benötigt werden. Es wird dargelegt, auf welche Art und Weise Software kommerziell vertrieben werden kann und auf welche Art und Weise der Endkunde mit den zum Betrieb notwendigen Lizenzierungsinformationen versorgt wird. Für die Integration eines Lizenzschutzes in ein eigenes Produkt kann ein Softwarehersteller ein bestehendes Kopierschutz-Produkt einsetzen, eigene Lösungen entwerfen oder beides kombinieren. Aus diesem Grund beschreibt und bewertet diese Arbeit zunächst einmal die auf dem Markt erhältlichen und in der Literatur beschriebenen Lösungen. Im Anschluß werden auf Grundlage der bekannten Lösungen im Kombination mit neuen Vorschlägen Vorgehensweisen beschrieben, die bei der Sicherung gegen unlizenzierte Nutzung eingesetzt werden können. Schließlich wird eine neue Lösung vorgestellt, die beschreibt, wie Lizenzierungsinformationen derart gestaltet werden können, daß diese flexibel verändert und erweitert werden können.

Neben der Zusammenfassung, Klassifizierung und Bewertung bestehender Erkenntnisse und der Schaffung eines breiten Überblicks über die Entwicklung und den Vertrieb modularer Software, unter Berücksichtigung des Schutzes vor unlizenzierter Nutzung, werden in dieser Arbeit, im Rahmen meines persönlichen Beitrags, auch neue innovative Erkenntnisse eingearbeitet. Dies sind insbesondere Verfahren unter Nutzung von Identifikationskarten oder biometrischer Merkmale, zum Verstecken von Schlüsseln mittels Steganographie oder als Maschinencode getarnt oder zum Abbruch von Programmen mittels nicht-lokaler Sprünge. Dazu kommen neue Abläufe bei der Freischaltung von Modulen durch Servicetechniker. Es werden Ansätze vorgeschlagen, mittels derer ein Konfigurationsmanagement bei komplexen modularen Projekten mit möglichen Erweiterungen durch Dritthersteller durchgeführt werden kann. Schließlich wird noch ein neues Modell vorge-

stellt, mit dem Lizenzierungsinformation so gestaltet werden, daß das Produkt modular und erweiterbar ist und darüber hinaus einzelne Module leicht freigeschaltet werden können.

Alles in allem läßt sich feststellen, daß Software bei derzeit verbreiteter Computerausstattung zwar nicht hundertprozentig vor unlizenzierter Nutzung geschützt werden kann, aber mit an die tatsächlichen Risiken angepaßtem Aufwand Lösungen gefunden werden können, welche die wirtschaftliche Überlebensfähigkeit des Softwareunternehmens erhalten. Wie in vielen Dingen sind hier Maß und Ziel entscheidend. So ist es bei Produkten mit begrenztem Markt sinnvoll, Kunden durch begleitende Maßnahmen wie Dienstleistungen an seine Software zu binden und Kopierschutzmechanismen nur als Ergänzung zu sehen.

Sowohl für die Modellierung komplexer Modulstrukturen, als auch für die Integration von Lizenzverwaltung und -prüfung, bieten die Ansätze der komponentenorientierten Programmierung und des Feature Modelling interessante Perspektiven, da dabei auch modulübergreifende Aspekte modelliert werden können.

Meiner Meinung nach wird sich der Kampf zwischen Herstellern und Crakkern gerade in Märkten mit breitem Zielpublikum (so z. B. bei Computerspielen) in den nächsten Jahren noch verschärfen, während die Firmen mit Nischenprodukten immer mehr auf begleitende Dienstleistungen setzen werden. In manchen Bereichen wandeln sich schon jetzt die bisherigen Hersteller von Softwaresystemen in Dienstleistungsunternehmen, welche freie Software, die beispielsweise unter der GNU General Public License (GPL) (vgl. Abschnitt 2.1.2) lizenziert ist, betreuen und Anpassungen im Rahmen von Werkverträgen durchführen.

Anhang

A. Quellcodebeispiele

Dieser Abschnitt enthält Beispielquellcodes für die im Text beschrieben Techniken.

```
// Ergänzt nach http://www.c-plusplus.de/forum/viewtopic-var-t-is-104590-and-start-is-0-and-postdays-is-0-and-postorder-is-asc-and-highlight-is-.html
#include <iostream>
#include <windows.h>

int main ()
{
  const int BUFSIZE = 32000;
  //Zeichenvorrat Buchstaben,Ziffern ohne AEIOU01 (24)
  UCHAR digits[] = {'B','C','D','F','G','H','J','K','M','P',
    'Q','R','T','V','W','X','Y','2','3','4','6','7','8','9'};
  PUCHAR strresult = new UCHAR[26];
  PUCHAR buf = new UCHAR[BUFSIZE];

  HKEY key = NULL;
  DWORD datasize = BUFSIZE ;
  DWORD dwRet = 0;
  int i = 0, j = 0;
  int x = 0;

  ZeroMemory((PVOID)strresult,26);

  dwRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
       "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",
       0,KEY_READ,&key);

  // Product ID bestimmen
  dwRet = RegQueryValueEx(key,"ProductID",NULL,NULL,
                          (LPBYTE)buf,&datasize);
  if (dwRet != ERROR_SUCCESS)
  {
    std::cerr << "Failed to read ProductID: Error Code: "
              << dwRet << std::endl;
  }
  else
  {
    std::cout << "ProductID: " <<  buf << std::endl;
  }

  // Digital Product ID bestimmen (ab Windows XP)
  dwRet = RegQueryValueEx(key,"DigitalProductID",NULL,NULL,
                          (LPBYTE)buf,&datasize);
```

```
42
43    if ( (dwRet != ERROR_SUCCESS) &&
44         (dwRet != ERROR_MORE_DATA))
45    {
46       std::cerr << "Failed to read DigitalProductID: "
47                 << dwRet << std::endl;
48    }
49    else
50    {
51
52      if (dwRet == ERROR_MORE_DATA)
53        std::cerr << "More data available" << std::endl;
54
55      for (i=24;i>=0;i--)
56      {
57        x=0;
58
59        for (j=14;j>=0;j--)
60        {
61          x = (x<<8) + (buf+0x34)[j];
62          (buf+0x34)[j] = x / 24;
63          x = x % 24;
64        }
65        strresult[i]=digits[x];
66      }
67      std::cout << strresult  << std::endl ;
68    }
69
70    RegCloseKey(key);
71
72    std::cin.get();
73    return 0;
74  }
```

Quellcode 1: Beispiel für Kopplung an Betriebssystem nach [Mara]

```
1  /*********************************************************
2   * Test program for determine CPUID information on       *
3   * Windows systems. Works only on processors that support *
4   * the CPUID instrution (Pentium compatible and latest    *
5   * 486).                                                 *
6   * Author:         Daniel Dreher  pentad@pentad.de       *
7   * Derived from an example published by Intel on URL     *
8   * http://www.intel.com/cd/ids/developer/asmo-na/eng/    *
9   * 257129.htm?prn=Y                                      *
10  * at reference date: 2006-02-04                         *
11  *********************************************************/
12 #include <stdio.h>
13
14 typedef struct cpuid_args_s     // Structure for registers
15 {
16         long eax;
```

```
        long ebx;
        long ecx;
        long edx;
} CPUID_ARGS;


void _CPUID(CPUID_ARGS* p)      // Wrapper function for
                                // CPUID instruction
{
    __asm
    {
        mov     edi, p
        mov eax, [edi].eax
        mov ecx, [edi].ecx // for functions such as eax=4
        cpuid
        mov [edi].eax, eax
        mov [edi].ebx, ebx
        mov [edi].ecx, ecx
        mov [edi].edx, edx
    }
}


int main()
{
        CPUID_ARGS ca;
        int maxlevel;
        char * p;

        ca.eax = 0;
        ca.ebx = 0;
        ca.ecx = 0;
        ca.edx = 0;
        _CPUID(&ca);    // call CPUID to get maximum level and
                        // vendor ID

        maxlevel = ca.eax;
        printf("Maximum support level:          %1i\n",
                maxlevel);
        p = (char*) &ca.ebx;
        printf("Vendor ID string:               "
           "%c%c%c%c%c%c%c%c%c%c%c%c\n",
           p[0], p[1], p[2], p[3], p[8], p[9], p[10], p[11],
           p[4], p[5], p[6], p[7]);


        if (maxlevel >= 1)
        {
                ca.eax = 1;
                ca.ebx = 0;
                ca.ecx = 0;
```

```
                ca.edx = 0;
                _CPUID(&ca);    // call CPUID to get processor
                                // type/family/model/stepping

                printf("Family:                          "
                       "%1i\n", (ca.eax >> 8) & 0xF);
                printf("Model:                           "
                       "%1i\n", (ca.eax >> 4) & 0xF);
                printf("Stepping:                        "
                       "%1i\n", (ca.eax) & 0xF);
                printf("Brand ID:                        "
                       "%1i\n", (ca.ebx) & 0xF);
                printf("Flags 1:                         "
                       "%08x\n", (ca.ecx) & 0xF);
                printf("Flags 2:                         "
                       "%08x\n", (ca.edx) & 0xF);
        }

        if (maxlevel >= 3)
        {
                ca.eax = 3;
                ca.ebx = 0;
                ca.ecx = 0;
                ca.edx = 0;
                _CPUID(&ca);    // call CPUID to get processor
                                // serial number
                printf("Serial number:                   "
                       "%08x%08x\n", ca.ecx, ca.edx);
        }

        return 0;
}
```

Quellcode 2: Beispiel für Kopplung an CPU nach [Pal] und [Lud]

```
#include <windows.h>
#include <wincon.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

typedef struct _ASTAT_
{

  ADAPTER_STATUS adapt;
  NAME_BUFFER    NameBuff [30];

}ASTAT, * PASTAT;

ASTAT Adapter;

void main (void)
```

```
{
  NCB Ncb;
  UCHAR uRetCode;
  char NetName[50];
  LANA_ENUM   lenum;
  int i;

  memset( &Ncb, 0, sizeof(Ncb) );
  Ncb.ncb_command = NCBENUM;
  Ncb.ncb_buffer = (UCHAR *)&lenum;
  Ncb.ncb_length = sizeof(lenum);
  uRetCode = Netbios( &Ncb );
  printf( "The NCBENUM return code is: 0x%x \n", uRetCode );

  for(i=0; i < lenum.length ;i++)
  {
          memset( &Ncb, 0, sizeof(Ncb) );
          Ncb.ncb_command = NCBRESET;
          Ncb.ncb_lana_num = lenum.lana[i];

          uRetCode = Netbios( &Ncb );
          printf( "The NCBRESET on LANA %d return code is: "
                  "0x%x \n", lenum.lana[i], uRetCode );

          memset( &Ncb, 0, sizeof (Ncb) );
          Ncb.ncb_command = NCBASTAT;
          Ncb.ncb_lana_num = lenum.lana[i];

          strcpy( Ncb.ncb_callname,  "*               " );
          Ncb.ncb_buffer = (char *) &Adapter;
          Ncb.ncb_length = sizeof(Adapter);

          uRetCode = Netbios( &Ncb );
          printf( "The NCBASTAT on LANA %d return code is: "
                  "0x%x \n", lenum.lana[i], uRetCode );
          if ( uRetCode == 0 )
          {
                 printf( "The Ethernet Number on LANA %d is: "
                         "%02x%02x%02x%02x%02x%02x\n",
                lenum.lana[i],
                          Adapter.adapt.adapter_address[0],
                          Adapter.adapt.adapter_address[1],
                          Adapter.adapt.adapter_address[2],
                          Adapter.adapt.adapter_address[3],
                          Adapter.adapt.adapter_address[4],
                          Adapter.adapt.adapter_address[5] );
          }
   }

```

```
67 }
```

Quellcode 3: Beispiel für Abfrage der MAC-Adresse laut [Micc]

B. Beispielabläufe

Dieser Abschnitt enthält einige beispielhafte Abläufe für die Auslieferung von Lizenzen an den Endkunden.

Die Abbildung 6 zeigt den Vertrieb von Lizenzen über einen Händler sowohl für neue als auch Erweiterungslizenzen.

Abbildung 7 zeigt den Ablauf für den Betrieb einer Applikation mittels ASP.

Abbildung 8 zeigt den Ablauf, wenn ein Dritthersteller für sein Erweiterungsprodukt eine generelle Lizenz erhält. Abbildung 9 zeigt den Ablauf, wenn ein Dritthersteller die Lizenzen beim Hersteller für jeden Kunden einzeln freischalten läßt.

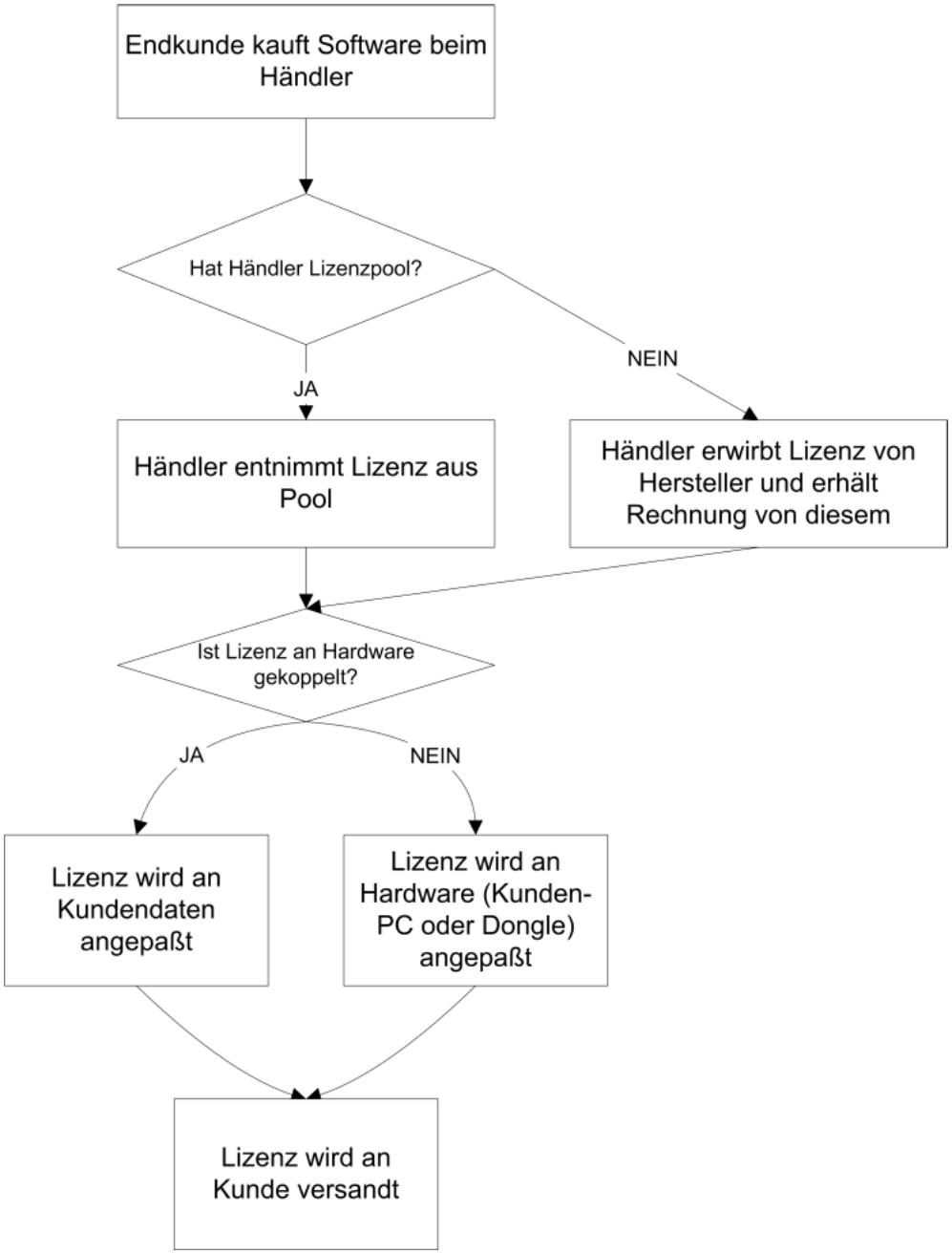

Abbildung 6: Ablauf: Lizenzvertrieb über einen Händler

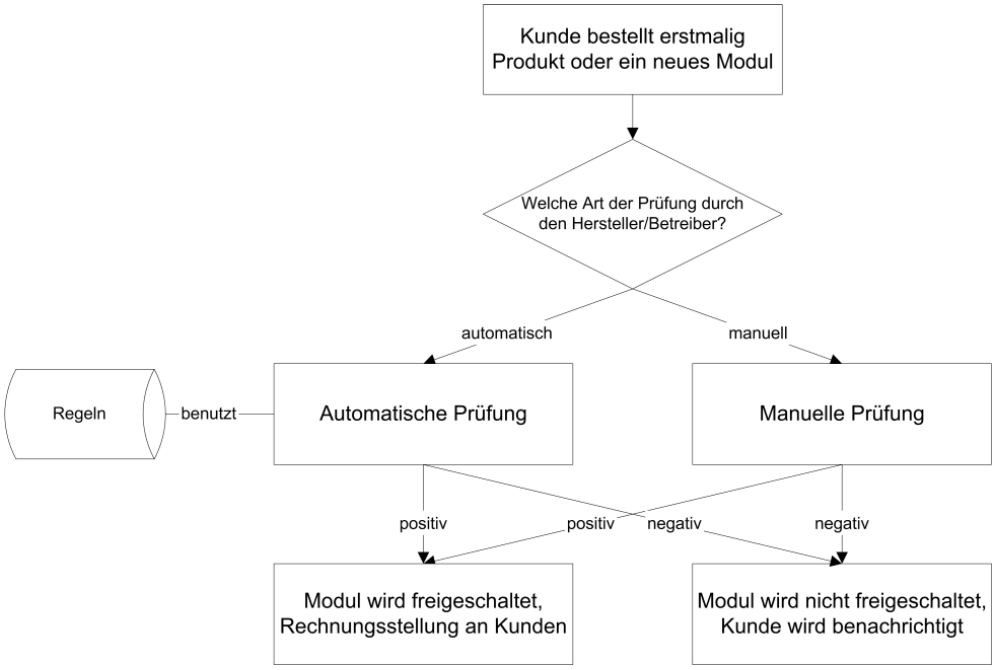

Abbildung 7: Ablauf: Lizenzierung bei ASP

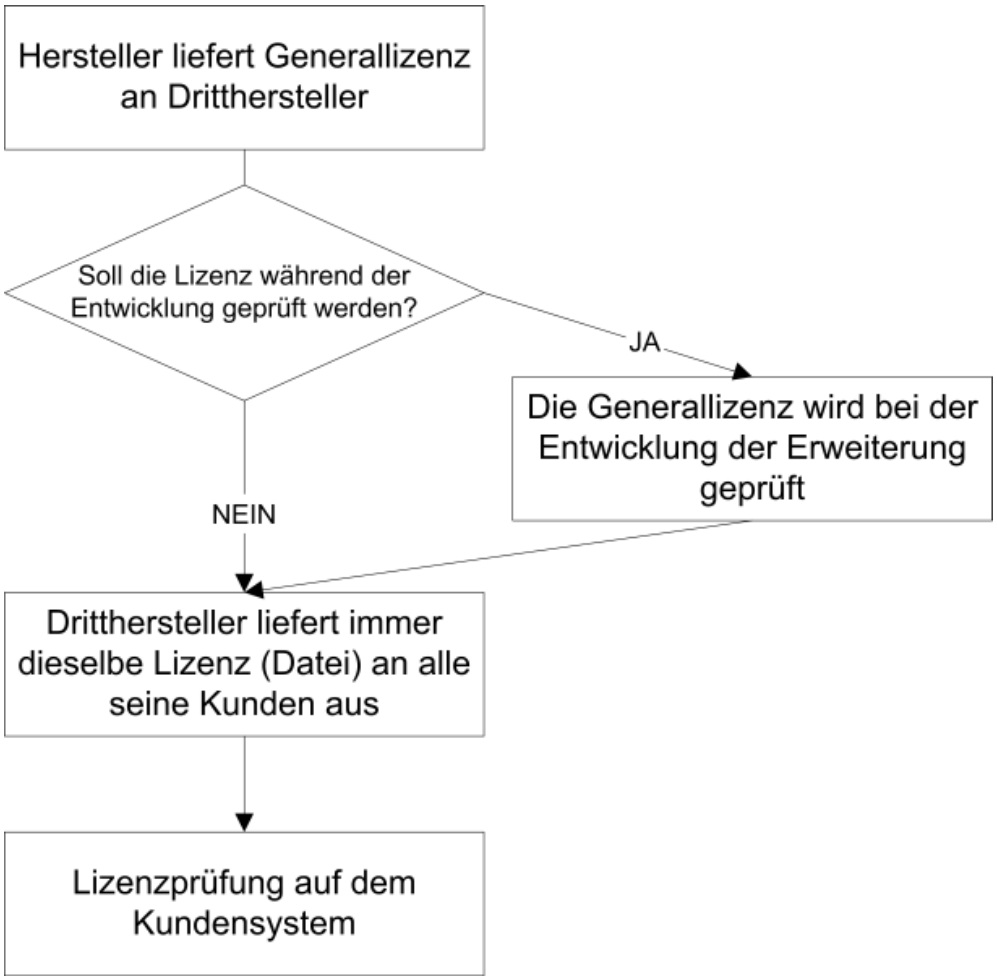

Abbildung 8: Ablauf: Lizenzierung an Dritthersteller über Generallizenz

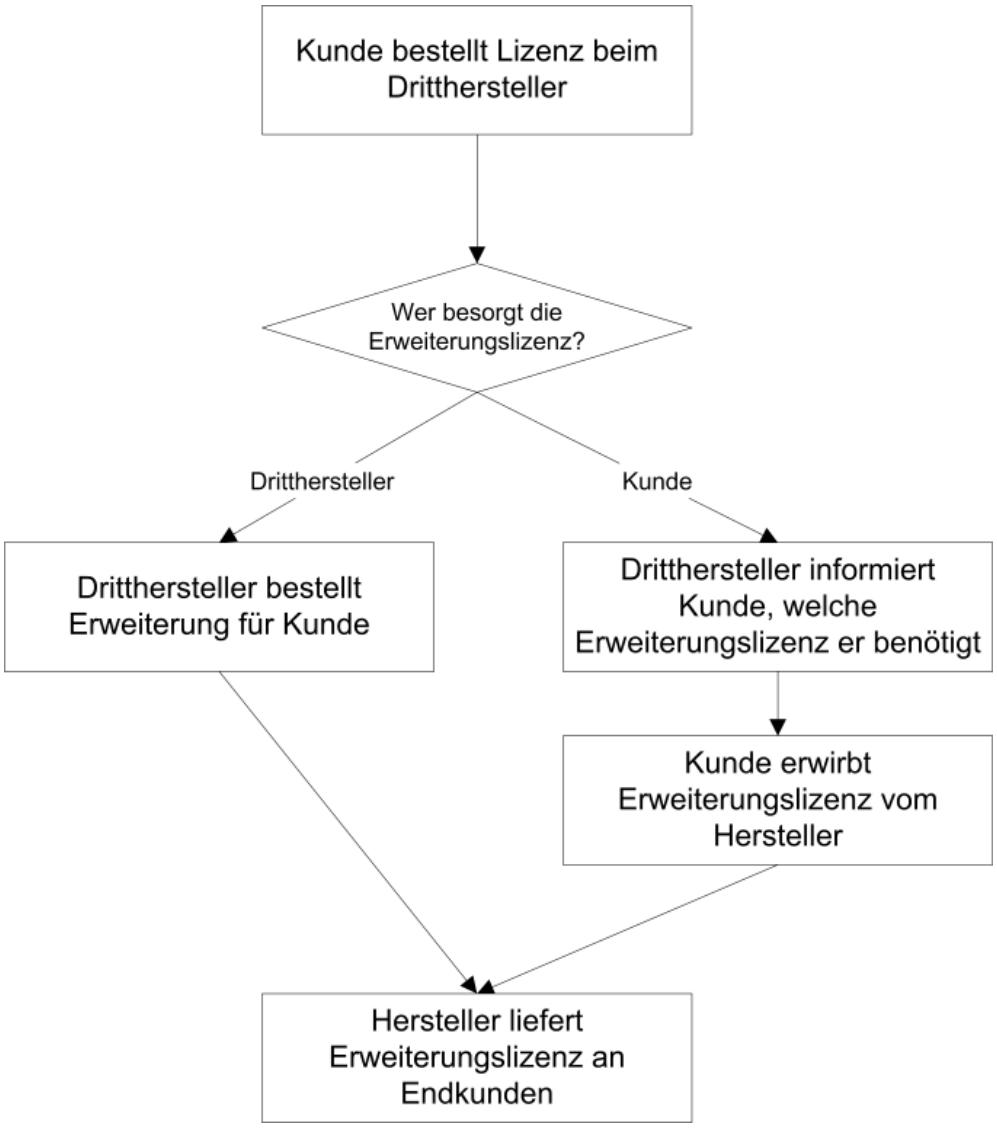

Abbildung 9: Ablauf: Lizenzierung an Dritthersteller über Einzellizenz

C. Glossar

AES	AES steht für “Advanced Encryption Standard” und ist ein symmetrisches Verschlüsselungsverfahren. Es wird nach seinen Entwicklern Rijmen und Daemen auch als Rijndael-Verfahren bezeichnet. Symmetrische Verfahren nutzen für die Ver- und Entschlüsselung von Daten denselben Schlüssel.	75
Anti-Spoofing	Umgehung von Kopierschutzmechanismen, von englisch “to spoof” = beschwindeln, verulken	41
API	API steht für “Application Programming Interface”. Es handelt sich dabei ursprünglich um eine Schnittstelle zum Betriebssystem, die zur Verwendung durch Anwendungsprogramme vorgesehen ist. Der Name wird allerdings auch für allgemeine Schnittstellen zu beliebigen Bibliotheken verwendet. So ist eine Lizenzierungs–API eine Schnittstelle zu Funktionen im Zusammenhang mit Lizenzprüfung.	44
API Hook	Hook heißt wörtlich übersetzt Haken; unter einem API Hook versteht man den Einzug einer Zwischenschicht zwischen API und Anwendung, die in Richtung Anwendung dieselben Schnittstellen verwendet, jedoch zusätzliche oder andere Funktionen ausführt.	77
ASP	Application Service Providing, Anbieten von gewissen Diensten zentral durch den Hersteller oder eine Drittfirma; so werden beispielsweise bei einem Lohnbuchhaltungssystem die Daten nicht beim Kunden, sondern über eine Netzwerkverbindung beim Hersteller erfaßt und verarbeitet. Der Anbieter von ASP kümmert sich dann auch um die Administration des Systems und Nebenaufgaben wie Datensicherung.	27
Aspekt	Ein Aspekt ist ein Modul im Sinne des Softwareentwurfs, welches dazu dient, Dinge zu implementieren, die nicht in einem abgeschlossenen funktionalen Bereich, sondern modulübergreifendend wirken. Aspektorientierte Programmierung dient dazu, diese modulübergreifenden Anforderungen (englisch: “Cross-cutting concerns”), d. h. Aspekte, zu modellieren und implementieren (vgl. [Fö05; Scha]).	13

Benutzerfirewall	Eine Benutzerfirewall ist auch als Personal Firewall bekannt. Eine Firewall dient dazu, den Netzwerkverkehr über einen bestimmten Knoten einzuschränken und ggf. auch zu kontrollieren. Dabei wird der Verkehr üblicherweise auf bestimmte Dienste oder die Kommunikation mit bestimmten Systemen eingeschränkt. Varianten, die auf dem lokalen System des Endbenutzers installiert werden und dem Benutzer die Möglichkeit geben, den Verkehr einzuschränken, werden dann als Benutzerfirewall bezeichnet.	52
Brute Force	Unter Brute Force versteht man ein Verfahren zur Ermittlung geheimer Paßwörter und Schlüssel, bei dem alle möglichen Kombinationen systematisch durchgeführt werden, bis der richtige Wert gefunden wurde. Bei Paßworteingaben wird dies erschwert, indem nach Eingabe eines falschen Paßworts eine gewisse Zeit keine weitere Eingabe mehr möglich ist.	44
Cäsar-Chiffre	Die Cäsar-Chiffre ist eine einfache Form der Verschlüsselung von Texten. Der Legende nach soll Gaius Julius Caesar Nachrichten dadurch verschlüsselt haben, daß er jeden Buchstaben durch den in einem bestimmten, in der Nachricht immer gleichen, Abstand stehenden Buchstaben ersetzt hat.	45
Call-Center	Eigenständiges Unternehmer oder Gruppe eines solches; das dort arbeitende Personal führt über Telefon gewisse Aufgaben wie die Annahme von Bestellungen, die Betreuung bestehender Kunden oder die Gewinnung neuer Kunden durch.	28
Carry-Flag	Unter dem Carry-Flag versteht man ein einzelnes Bit eines Registers, das dann gesetzt wird, wenn bei einer arithmetischen Operation ein Übertrag entsteht.	45
CGI	CGI steht für “Common Gateway Interface”. Es handelt sich dabei um eine Standardschnittstelle für den Datenaustausch zwischen einem Webserver und Skripten bzw. Programmen.	85
Client-Server-Architektur	Architektur in der Softwareentwicklung, bei der auf einem zentralen System (dem Server) gewisse Aufgaben für andere Systeme (die Clients) durchgeführt werden, an welchen die eigentlichen Benutzer arbeiten.	51
CMOS	steht eigentlich für Complementary Metal Oxide Semiconductor, also für integrierte Schaltkreise, die gleichzeitig sowohl p=Kanal als auch n=Kanal=MOSFETs verwenden. Bei PC=Systemen wird der Begriff aber auch für das batteriegepufferte SRAM, in dem die BIOS=Parameter gespeichert werden, verwendet	47

Crack	Unter einem “Crack” versteht man einen Mechanismus oder ein Programm, daß dazu dient, Schutzmaßnahmen bei Software, insbesondere auch Kopierschutzmechanismen, zu umgehen. Die Personen, welche den Crack programmiert wird als “Cracker” (und nicht wie oft fälschlich als “Hacker”) bezeichnet. Ein Crack-Programm kann dabei u. a. bestimmte Funktionen der Originalsoftware ändern oder ersetzen.	23, 50
Cracker	Ein Cracker ist eine Person, die Cracks erstellt (vgl. dort)	44
CRC	Cyclic redundancy check, zyklische Redundanzprüfung: ein Verfahren zur Bestimmung von Prüfsummen über Daten. Es zeichnet sich dadurch aus, daß Einbitfehler garantiert erkannt werden.	74
dev-Dateisystem	besonderes Dateisystem in von Unix abgeleiteten Systemen, enthält Dateien, die für den Zugriff auf Geräte bzw. deren Treiber genutzt werden.	69
DLL	Dynamic Link Library, darunter versteht man bei Windows=Systemen eine dynamisch gebundene Bibliothek.	79
ERP-System	Enterprise Ressource Planning, System zur Verwaltung aller Geschäftsprozesse in einem Unternehmer, oft auch spezieller: Verwaltung von Betriebsmitteln, Personal und Kapital.	52
FPGA	Field Programmable Grid Arrays, d. h. frei programmierbare Logikschaltkreise. Ein FPGA besteht aus einer Vielzahl von Logikschaltkreisen, deren Logik programmierbar ist. Dabei gibt es sowohl mehrfach als auch nur einmalig programmierbare Bausteine. Sie werden oft zu Verarbeitung einfacher Algorithmen speziell im Bereich Signalverarbeitung eingesetzt. Durch die Vielzahl von einsetzbaren Schaltkreisen ist ein hoher Grad von Parallelverarbeitung möglich. Daher sind Aufgaben mit geringerer Komplexität vielfach schneller als mit herkömmlichen Prozessoren zu bearbeiten.	72
HTTP-Protokoll	Das HTTP-Protokoll ist ein Netzwerkprotokoll auf Anwendungsebene (i. d. R. über dem TCP-Protokoll angesiedelt). HTTP steht für “Hypertext Transfer Protocol” und es dient hauptsächlich zum Austausch von Daten innerhalb des World Wide Web, das inzwischen immer mehr als “Internet” bezeichnet wird. Viele Firmennetzwerke erlauben ihren Benutzern Datenverbindungen nach außen nur über das HTTP-Protokoll.	28, 81

inline-Aufrufe	Unter inline-Aufrufen von Funktionen, versteht man eine Technik, mittels derer dem Compiler mitgeteilt wird, daß der Code einer Funktion bei jedem Aufruf in das Maschinenprogramm eingefügt werden soll. Ansonsten werden Funktionen über einen Funktionsaufruf aufgerufen und deren Code steht nur einmal im Maschinenprogramm. Inline-Aufrufe werden für kurze zeitkritische Codeabschnitte verwendet, wenn die Zeit für den Aufruf der Funktion (u. a. für die Übergabe der Parameter über den Stack und die Speicherung der Rücksprungadresse) im Verhältnis zum Nutzcode relativ lang ist. Der erzeugte Code wird an dieser Stelle länger, aber i. allg. auch schneller.	44
ISDN	ISDN stand ursprünglich für “Integrierter Sprach- und Datennetz”, wird inzwischen jedoch als “Integrated Services Digital Network” verstanden, ohne daß sich die Bedeutung geändert hat. ISDN ist ein Protokoll für digitale Telefon- und Datendienste.	33
Kernel-Debugger	Debugger, die nicht als Applikation gestartet werden, sondern als eine Art Treiber mit dem System gestartet werden. Diese werden neben der Analyse kopiergeschützter Software auch zum Debuggen von Treibern verwendet.	76
Keygen	Keygen ist eine verkürzte Form des englischen Worts “key generator”; Als Keygen werden Programme bezeichnet, die zur Erstellung von Lizenzierungsinformationen und Aktivierungsschlüsseln dienen. Üblicherweise werden nur die von Dritten erzeugten Programme, welche die unlizenzierte Nutzung der Software ermöglichen sollen, so bezeichnet, aber nicht diejenige Software, welche der Hersteller zur Erzeugung der Schlüssel für seine Kunden nutzt.	23, 85
Konfigurationsmanagement	Konfigurationsmanagement sind nach [DIN03] ‘koordinierte Tätigkeiten zur Leitung und Lenkung der Konfiguration’, wobei unter einer Konfiguration ‘miteinander verbundene funktionelle und physikalische Merkmale eines Produkts [...]’ verstanden werden; Bei modularer und erweiterbarer Software sind dies insbesondere die Verwaltung der Quellen und die Zusammenstellung verschiedener Module und Erweiterungen, die Koordination des Zusammenwirkens der zu unterschiedlichen Zeiten durch unterschiedliche Beteiligte entwickelten Module und Aspekte.	14

MAC-Adresse	MAC steht für “Media Access Control”. Die MAC-Adresse ist eine eindeutige Adresse einer Netzwerkkomponenten auf Ebene der Sicherungsschicht des OSI–Modells. Sie wird üblicherweise vom Hersteller der Komponenten vergeben und muß in einem Netzwerk eindeutig sein, da ansonsten die Pakete nicht mehr korrekt adressiert werden könnten.	69
maliziöse Software	Software, die auf Kosten des Betreibers des Systems unerwünschte Funktionen ausführt. Dazu gehören Viren, Trojaner und Rootkits	27
NOOP	NOOP steht für “no operation” und bezeichnet einen Maschinenbefehl, der keinerlei Auswirkungen (außer der Veränderung des Befehlszeigers) hat. Bei den meisten Prozessor-Plattformen wird er als Null-Byte codiert.	44
OCX	OCX-Komponenten heißen eigentlich ActiveX-Komponenten, werden aber aufgrund ihrer Dateiendung ‚OCX‘ so genannt. ActiveX ist eine Erweiterung des proprietären “Component Object Model” (COM) Standards der Firma Microsoft. COM dient dazu, aus Bibliotheken Klassen über einen definierten Client-Server-Mechanismus zu exportieren.	41
OEM	Original Equipment Manufacturer; ein solcher Hersteller stellt Produkte oder Komponenten her, die von anderen Herstellern ggf. weiterverarbeitet und schließlich in Verkehr gebracht werden. Wenn Software von einem OEM erstellt wird, kommt diese i. allg. in Zusammenhang mit Hardware des anderen Herstellers zum Einsatz. Letzterer bringt das Gesamtprodukt in Verkehr und übernimmt Pflichten wie Gewährleistung gegenüber dem Endkunden.	26
proc-Dateisystem	besonderes Dateisystem in von Unix abgeleiteten Systemen, enthält neben Daten über die gerade vorhandenen Prozesse auch Informationen über die angeschlossenen Prozesse.	69
Product ID	Eine “Product ID” ist eine eindeutige Nummer, die einer bestimmten Lizenz einer Software zugeordnet wird. Sie ist also eindeutig und mit dem Endkunden verbunden. Manchmal stimmt sie mit der Aktivierungsnummmer überein.	50

Proxy-Server	Ein Proxy ist ein System, daß zwischen einem Client und einem Server steht und Anfragen vom Client zum Server weiterleitet. Er kann die Anfragen direkt und unverändert weitergeben (transparenter Proxy) und sieht damit für den Client wie der Server und für den Server wie der Client aus. Manche Systeme führen bei der Weitergabe noch andere Schritte wie die Protokollierung des Verkehrs, das Caching von oft angefragten Daten oder die Filterung von Inhalten durch. Proxys werden oft in Zusammenhang mit dem HTTP-Protokoll in Firmennetzwerken eingesetzt, um u. a. den Datenverkehr nach außen zu minimieren und schädliche Inhalte auszufiltern.	81
Rootkit	Ein Rootkit ist ein Programm, daß entweder dazu dient oder es unfreiwillig bewirkt, daß unberechtigte Dritte Zugang zu einem Computersystem erhalten. Diese Personen erhalten dann administrative Rechte wie beispielsweise die Rechte des Überbenutzers “root” auf Unix-Systemen (daher auch der Name “Rootkit”). Die administrativen Rechte bewirken, daß das System beliebig genutzt oder manipuliert werden kann.	24
ROT-13	ROT-13 war in den Anfangszeiten von Newsgroups ein Verfahren, mit dem verhindert werden sollte, daß Texte von jedermann gelesen werden konnten. Dabei wurde wie bei der Cäsar-Chiffre (vgl. dort) vorgegangen, jedoch immer um 13 Zeichen verschoben. Das das Alphabet aus 26 Buchstaben besteht, waren der Ver- und Entschlüsselungsalgorithmus identisch.	45
RSA	RSA, nach den Initialen seiner Erfinder Rivest, Shamir und Adleman benannt, ist asymmetrisches Verschlüsselungsverfahren. Asymmetrische Verfahren nutzen für die Ver- und Entschlüsselung von Daten unterschiedliche Schlüssel.	75
Schreibtischtest	Testmethode aus der Softwareentwicklung; beim Schreibtischtest wird das Programm Schritt für Schritt auf Papier ausgeführt und die Variablen- bzw. Registerwerte dazu notiert.	76
Spaghetti-Code	Unter Spaghetti-Code versteht man einen Programmierstil, bei das Programm oder Teile des Programms möglichst unstrukturiert hintereinander geschrieben werden. Er ist schwer zu warten und schlecht zu lesen.	44
SSL	SSL steht für Secure Sockets Layer und bezeichnet ein Verschlüsselungsprotokoll, daß oft in Verbindung mit dem HTTP-Protokoll eingesetzt wird.	81

Timebombing	Als “Timebombing” wird in der Umgangssprache der Informatiker die technische Gestaltung von Hard- und Software bezeichnet, die dafür sorgt, daß ein Produkt nach einer bestimmten Zeit oder einer bestimmten Anzahl Vorgänge nicht mehr nutzbar ist. Bei Geräten wird dies oft damit begründet, daß Verschleißteile dann ihre Lebensdauer überschritten hätten. In manchen Fällen wird der Kunde weder über das Verhalten aufgeklärt, noch kann er die Ursache der Fehlfunktion erkennen. Ein solches Vorgehen kann unter bestimmten Umständen als Betrug gewertet werden.	82
Trap Flag	Schalter im Flags-Register der i386-kompatiblen Prozessoren, in vergleichbarer Form auch auf anderen Systemen vorhanden; sorgt dafür, daß die Ausführung nach jedem Befehl mit einer Exception/Interrupt Service Routine unterbrochen wird	76
USB	USB steht für “Universal Serial Bus”, USB ist ein serielles Bus-System, daß für das Verbinden von Peripheriegeräten mit einem Computer entwickelt wurde. Es ersetzt die bisherigen PC-Standardschnittstellen PS2, serielle (RS232) und parallele Schnittstelle.	71
WAN	Wide Area Network, d. h. ein Netzwerk, das zur Kommunikation zwischen mehreren entfernten Teilnehmern dient	49
Wasserzeichen	Ein Wasserzeichen war ursprünglich eine im Papier eingebrachte Markierung, die nur im Gegenlicht erkennbar war. Manche Menschen lassen sich Papier mit einem für sie spezifischen Wasserzeichen herstellen, um damit die Authentizität ihre Schriftstücke zu untermauern. In der Datenverarbeitung sind Wasserzeichen Markierungen in Daten, die nicht auf den ersten Blick erkennbar, aber mit gewissen technischen Mitteln kontrolliert werden können. Oft werden sie mittels Steganographie in Bilddateien eingefügt, um die (auch unlizenzierte) Verbreitung von Bildern oder Audiodateien eines Urhebers nachvollziehen zu können.	82
Windows-Registry	Die Windows-Registry ist eine Datenbank in Windows-Betriebssystemen, welche zur Speicherung von Einstellungen des Systems und von Anwendungsprogrammen vorgesehen ist. Die Werte werden in einer Baumstruktur gespeichert und auf sie wird mittels ihres Pfades entlang des Baums zugegriffen.	46

D. Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literatur und andere Quellen

[Act] ActiveSoft.NET: *Software Activation Service*. http://www.activatesoft.net/. – Produktinformationsseite, Abruf: 15.01.2006

[Agi] Agilis Software LLC: *EasyLicenser Datasheet*. http://www.agilis-sw.com/ezlm/datasheet.htm. – Produktinformationsseite, Abruf: 21.01.2006

[Akt] Aktiv Company: *Guardant Anti-Piracy Protection*. http://guardant.com/. – Produktinformationsseite, Abruf: 5.11.2005

[Alaa] Aladdin Knowledge Systems Germany: *DRM - Digital Rights Management - HASP Software DRM Lizenzierung Kopierschutz Softwaresicherheit - HASP von Aladdin Knowledge Systems*. http://www.aladdin.de/produkte/software_drm/hasp_uebersicht.html. – Produktinformationsseite, Abruf: 27.10.2005

[Alab] Aladdin Knowledge Systems Germany: *Privilege SCP 6.0 Software Aktivierung, Produktaktivierung, eCommerce, Pressemitteilung 09/2003 von Aladdin Knowledge Systems*. http://www.aladdin.de/presse/pr2003_09.html. – Pressemitteilung, Abruf: 26.10.2005

[Alk] Alkonost Software: *Alkonost ContraCopy Software distribution protection key diskettes*. http://www.contracopy.com/index.htm. – Produktinformationsseite, Abruf: 5.11.2005

[Ant] AntiCracking: *SVKP*. http://www.anticracking.sk/products_svkp.html. – Produktinformationsseite, Abruf: 5.11.2005

[ASP] ASPack Software: *ASPACK SOFTWARE Best Choice Compression and Protection Tools for Software Developers*. http://www.aspack.com/asprotect.html. – Produktinformationsseite, Abruf: 13.02.2006

[Atm] Atma Software: *Atma Software*. http://atma-software.com/. – Produktinformationsseite, Abruf: 5.11.2005

[BIL06] Software und Spiele legal kopieren. In: *Computer Bild* (2006), Nr. 3, S. 50–58

[Boc] BOCKELKAMP, Matthias: *Bockelkamps Homepage*. http://mbockelkamp.dyndns.org/mbockelkamp/disconti.htm. – Informationsseite, Abruf: 04.02.2006

[Bun] Bundesgerichthof: *Urteil vom 06.07.2000, Aktenzeichen Nr. 1 ZR 244/97*. http://www.jurpc.de/rechtspr/20000220.htm. – Urteil, Abruf: 15.01.2006

[Bys] Byssus Software Solutions Ltd: *Software Activation*. http://www.byssus.com/solutions/features_and_benefits.html. – Produktinformationsseite, Abruf: 3.11.2005

[CE00] CZARNECKI, Krysztof ; EISENECKER, Ulrich: *Generative Programming: Methods, Tools and Applications*. Boston, MA, USA : Addison Wesley Professional, 2000. – ISBN 0–201–309777

[Cek] CEKLIC, Blagoje: *PC Guard for Win32/.NET - Software protection and licensing system*. http://www.sofpro.com/pcgw32.htm. – Produktinformationsseite, Abruf: 3.11.2005

[Čer02] ČERVEŇ, Pavol: *Crackproof your software*. San Francisco, CA, USA : No Starch Press Inc., 2002. – ISBN 1–886411–79–4

[Com] Compuware Corporation: *Compuware SoftICE Driver Suite*. http://www.compuware.com/products/driverstudio/softice.htm. – Produktinformationsseite, Abruf: 07.02.2006

[Con] Concept Software Inc.: *Software Activation Service*. http://www.softwarekey.com/swk_products/software_solutions/licensing/licensing.asp. – Produktinformationsseite, Abruf: 3.11.2005

[Cry] CrypKey Inc.: *CrypKey SDK Product Information - CrypKey Software Copy Protection & Licensing Control since 1992*. http://www.crypkey.com/sdk.asp. – Produktinformationsseite, Abruf: 3.11.2005

[Cus] Custom Software Solutions Ltd.: *CodeProtector - Software Protection System For Visual Studio 6 And .Net C/C++ Software*. http://www.customsoftwaresolutions.co.uk/CodeProtector.htm. – Produktinformationsseite, Abruf: 28.10.2005

[Dem] DEMEULEMEESTER, Samuel: *CPU-Z*. http://www.cpuid.org/cpuz.php. – Informationsseite, Abruf: 03.02.2006

[Dig] Digital River Inc.: *Silicon Realms*. http://siliconrealms.com/armadillo.shtml. – Produktinformationsseite, Abruf: 09.01.2005

[DIN03] Norm DIN ISO 10007:2003. *Qualitätsmanagement – Leitfaden für Konfigurationsmanagement*

[Dre01] DREHER, Daniel: *Sicherheit in sozialwissenschaftlicher Sicht*. Furtwangen im Schwarzwald, Fachhochschule Furtwangen, Seminararbeit, 2001. http://www.pentad.de/HausarbeitSeminar.pdf. – Online–Ressource

[Dre02] DREHER, Daniel: *Embedded PCs und Linux als Benutzerschnittstelle zu einem Server am Beispiel eines Zeiterfassungssystems*. Furtwangen im Schwarzwald, Deutschland, Fachhochschule Furtwangen, Diplomarbeit, 2002

[Dyk06] DYKI, Krzysztof: *Some questions about your products*. 2006. – Briefwechsel

[Ecl] Eclipse Foundation Inc.: *The AspectJ Project at Eclipse.org*. http://www.eclipse.org/aspectj/. – Internetseite, Abruf: 04.03.2006

[EGK] ESSWEIN, Werner ; GREIFFENBERG, Steffen ; KLUGE, Christian: *Konfigurationsmanagement von Modellen*. http://wise.wiwi.tu-dresden.de/systementwicklung/profil/publikation/artikel_buchbeitraege/downloads/konfigurationsmanagement.pdf. – Internetseite, Abruf: 21.02.2006

[Eil05] EILAM, Eldad: *Reversing: Secrets of Reverse Engineering*. Indianapolis, IN, USA : Wiley Publishing Inc., 2005. – ISBN 0–7645–7481–7

[Exe] Exeshield.com: *Leading Software Copy Protections for virtually any Win32 executable - ExeShield*. http://www.exeshield.com/features.htm. – Produktinformationsseite, Abruf: 3.11.2005

[Fel99] FELLNER, Richey: *Richey's Anti-Cracking FAQ*. Version: 1999. http://www.woodmann.com/crackz/Tutorials/Anticrk.htm. – FAQ, Abruf: 28.10.2005

[Fer] FERRAZ, Nelson: *Protect your VB programs against piracy with ActiveLock (Freeware)*. http://www.activelock.com/. – Produktinformationsseite, Abruf: 5.11.2005

[Fö05] FÖRSTER, Florian: *Point-To Analyse und darauf basierende Advice Interferenzanalyse für eine Kernsprache für AspectJ*. Passau, Deutschland, Universität Passau, Diplomarbeit, 2005

[Fre] Free Software Foundation: *FSF - GNU General Public License*. http://www.fsf.org/licensing/licenses/gpl.html. – Internetseite, Abruf: 1.11.2005 Version 2 vom Juni 1991

[Fuc05] FUCHS, Joachim: Du darfst, du darfst nicht... In: *dotnetpro* 20. Jahrgang (2005), Nr. 9, S. 46–52

[GAP] GAPS Inc.: *GAPSInc.com - Global Anti-Piracy Systems*. http://www.gapsinc.com/. – Produktinformationsseite, Abruf: 25.1.2006

[Ges] Gesellschaft für Datenanalyse und Fernerkundung bR: *Die Definition freier Software*. http://www.gdf-hannover.de/lit_html/grass60_v1.2/node91.html. – Internetseite, Abruf: 3.11.2005

[Gooa] Google: *Crack - Google-Suche*. http://www.google.de/search?q=Crack. – Internetseite, Abruf: 24.3.2006

[Goob] Google: *Keygen - Google-Suche*. http://www.google.de/search?q=Keygen. – Internetseite, Abruf: 24.3.2006

[Hei03a] Heise Zeitschriften Verlag GmbH & Co. KG: *Filmwirtschaft startet Abschreckungskampagne gegen Raubkopierer*. Version: 2003. http://www.heise.de/newsticker/meldung/42431. – Nachrichtenseite, Abruf: 24.3.2006

[Hei03b] Heise Zeitschriften Verlag GmbH & Co. KG: *Kein Vertrauen in Trusted Computing*. Version: 2003. http://www.heise.de/newsticker/meldung/39280. – Nachrichtenseite, Abruf: 12.03.2006

[Hei05] Heise Zeitschriften Verlag GmbH & Co. KG: *Sony BMGs Kopierschutz mit Rootkit-Funktionen*. Version: 2005. http://www.heise.de/security/news/meldung/65602. – Nachrichtenseite, Abruf: 13.02.2006

[Her99] HEROLD, Helmut: *Linux-Unix-Systemprogrammierung*. 2. Auflage. München, Deutschland : Addison-Wesley-Longman, 1999. – ISBN 3–8273–1512–3

[Ins] Institut für Rechtsfragen der Freien und Open Source Software: *Lizenz-Center*. http://www.ifross.de/ifross_html/lizenzcenter.html. – Internetseite, Abruf: 20.11.2005

[Int] Interactive Studios Inc.: *Quick License Manager - Create professional looking license keys*. http://www.interactive-studios.net/products/qlm.htm. – Produktinformationsseite, Abruf: 26.10.2005

[Ion] Ionworx Technologie: *Ionworx Technologie - the next generation of electronics delivery*. http://www.ionworx.com/solutions.html. – Produktinformationsseite, Abruf: 28.10.2005

[Isk] ISKANDAR, Zeddy: *How to retrieve the REAL Hard Drive Serial Number*. http://www.codeproject.com/csharp/hard_disk_serialno.asp. – Manual, Abruf: 11.02.2006

[Jup] Jupitermedia Corporation: *What is software licensing? A Word Definition From The Webopedia Computer Dictionary*. http://www.webopedia.com/TERM/S/software_licensing.html. – Internetseite, Abruf: 3.11.2005

[Kla06] Klaß & Ihlenfeld Verlag GmbH: *Boykott-Aufruf gegen StarForce-Kopierschutz*. Version: 2006. http://www.golem.de/0601/43056.html. – Nachrichtenseite, Abruf: 13.02.2006

[Li] LI, Paul: *Windows Management Instrumentation (WMI) Implementation*. http://www.codeproject.com/csharp/wmi.asp. – Manual, Abruf: 11.02.2006

[Lic] Licensing Technologies Limited: *Sheriff - Software Copy Protection*. http://www.sheriff-software.com/. – Produktinformationsseite, Abruf: 5.11.2005

[Lud] LUDLOFF, Chistian: *IA-32 architecture*. http://www.sandpile.org/ia32/cpuid.htm. – Informationsseite, Abruf: 03.02.2006

[Mara] BÄCKMANN, Markus (Hrsg.): *Windows Product Key auslesen*. http://www.c-plusplus.de/forum/viewtopic-var-t-is-104590-and-start-is-0-and-postdays-is-0-and-postorder-is-asc-and-highlight-is-.html. – Internetseite, Abruf: 29.11.2005

[Marb] MARR, Stefan: *Modulare Software Architektur und Abstrakte Datentypen*. http://www.stefan-marr.de/artikel/informatik/msa-adt.html. – Internetseite, Abruf: 30.10.2005

[MARc] MARX Software Security GmbH: *Überblick MARX Software Security*. http://www.marx.com/de/softwareschutz/ueberblick.php. – Produktinformationsseite, Abruf: 13.02.2006

[Mas05a] MASLO, Andreas: Programminstallation online überwachen. In: *dotnetpro* 20. Jahrgang (2005), Nr. 9, S. 38–45

[Mas05b] MASLO, Andreas: Programmschutz à la Microsoft. In: *dotnetpro* 20. Jahrgang (2005), Nr. 9, S. 22–37

[Mica] Microcomputer Applications Inc. (MAI): *KEY-LOK Dongle - Preventing Software Piracy for over 21 Years*. http://www.keylok.com/products.shtml. – Produktinformationsseite, Abruf: 3.11.2005

[Micb] Microcosm: *CopyMinder - The next Generation of Copy Protection*. http://www.copyminder.com/details.php. – Produktinformationsseite, Abruf: 3.11.2005

[Micc] Microsoft Corporation: *How To Get the MAC Address for an Ethernet Adapter*. http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q118623. – Manual, Abruf: 04.02.2006

[Micd] Microsoft Corporation: *Microsoft Product Activation FAQ - Software Piracy Protection*. http://www.microsoft.com/piracy/activation_faq.mspx. – FAQ, Abruf: 28.10.2005

[Mice] Microsoft Corporation: *Volume Management Functions*. http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q118623. – Manual, Abruf: 11.02.2006

[Micf] Microsoft Corporation: *Volume Management Functions*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/getdrivetype.asp. – Manual, Abruf: 11.02.2006

[Micg] Microsoft Corporation: *Windows Management Instrumentation*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_start_page.asp. – Manual, Abruf: 11.02.2006

[Mich] Microsoft Corporation: *Windows XP: Microsoft Product Activation*. http://www.microsoft.com/technet/prodtechnol/winxppro/evaluate/xpactiv.mspx. – Manual, Abruf: 11.02.2006

[Mici] Microsoft Corporation: *WMI C++ Application Examples*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/example__getting_wmi_data_from_the_local_computer.asp. – Manual, Abruf: 03.02.2006

[Mil] Milde Software Solutions: *Armadillo/SoftwarePassport: Produktübersicht und -einführung*. http://www.siliconrealms.de/produkte/. – Produktinformationsseite, Abruf: 26.10.2005

[Mir] Mirage Computer Systems GmbH: *Mirage Computer Systems GmbH: Licence Protector*. http://www.mirage-systems.de/403.html?&L=2. – Produktinformationsseite, Abruf: 26.10.2005

[Nal] Nalpeiron: *Nalpeiron copy protection and licensing systems*. http://www.nalpeiron.com/products/overview.asp. – Produktinformationsseite, Abruf: 3.11.2005

[Ope] Open Source Technology Group: *CpuId for Windows*. http://gnuwin32.sourceforge.net/packages/cpuid.htm. – Informationsseite, Abruf: 04.02.2006

[Opt] Optimum Gesellschaft für Automatisierungstechnik mbH: *Machen Sie aus genutzter Software bezahlte Software!* http://www.optimum-gmbh.de/Optimum/Softwareprodukte/Professional/OLicense-Suite/. – Produktinformationsseite, Abruf: 7.11.2005

[Ore] Oreans Technologies: *Oreans Technologies: Software Security Defined*. http://www.themida.com/. – Produktinformationsseite, Abruf: 13.02.2006

[PAC] PACE Anti Piracy: *Welcome to PACE Anti Piracy*. http://www.paceap.com. – Produktinformationsseite, Abruf: 5.11.2005

[Pal] PALMER, Eric: *How to Use All of CPUID for x64 Platforms under Microsoft Visual Studio™.NET 2005*. http://www.intel.com/cd/ids/developer/asmo-na/eng/257129.htm?prn=Y. – Informationsseite, Abruf: 03.02.2006

[PEi] *PEiD*. http://peid.tk. – Produktinformationsseite, Abruf: 12.03.2006

[Pix] PixelPlanet GmbH: *XPressUpdate Produkte Info*. http://www.xpressupdate.de/produkte/index.html. – Produktinformationsseite, Abruf: 25.10.2005

[Pro] Protection Technology Russia: *Starforce®*. http://www.star-force.com/protection/protection.phtml?c=300. – Produktinformationsseite, Abruf: 13.02.2006

[Ram97] RAMANCHAUSKAS, Vitas: *What is software licensing? A Word Definition From The Webopedia Computer Dictionary*. Version: 1997. http://lastbit.com/vitas/antihack.asp. – Internetseite, Abruf: 3.11.2005

[Rav] RAVIA, Fjalar: *How to protect better*. http://www.searchlores.org/protec/protec.htm. – Internetseite, Abruf: 5.11.2005

[Rei] Reinhold Software Services: *Reinhold Software Services*. http://www.reinhold-software-services.de/produkte.htm. – Produktinformationsseite, Abruf: 3.11.2005

[Ris] Risco Software Inc.: *AntiCrack Software Protection – software protection and software license management tools for developers*. http://www.ultraprotect.com/. – Produktinformationsseite, Abruf: 26.10.2005

[Saf] SafeNet Inc.: *Sentinel™UltraPro*. http://www.safenet-inc.com/products/sentinel/ultraPro.asp. – Produktinformationsseite, Abruf: 13.02.2006

[Scha] SCHULT, Wolfgang: *Aspektorientierte Programmierung*. http://kbs.cs.tu-berlin.de/%7Emwerner/discourse/documents/BlockLV03/schult-aop.pdf. – Internetseite, Abruf: 26.02.2006

[Schb] SCHULZ, Hans-Peter: *BIOS Kompendium*. http://www.bios-info.de/download/dlkomp.htm. – Informationsseite, Abruf: 04.02.2006

[Sch06] SCHNEIDER, Toni: Kopierern das Handwerk legen. In: *Computer & Automation* (2006), Nr. 1, S. 41–43

[SDP] SDProtector: *Software protection, copy protection with SDProtector Software Copy Protection System*. http://www.sdprotector.com/. – Produktinformationsseite, Abruf: 27.10.2005

[SG] SG Intec Ltd. & Co. KG: *SG-Lock Kopierschutz und Kryptosysteme*. http://www.sg-lock.de/. – Produktinformationsseite, Abruf: 26.1.2006

[Sis] Siskinsoft.com: *AntiCrack Protector anticrack antidebug protect exe compress exe protect dll compress dll*. http://www.siskinsoft.com/protector/. – Produktinformationsseite, Abruf: 28.10.2005

[Smi] SMITH, Adam: *Defending Shareware Against Cracks*. http://www.viratech.com/sharenc.htm. – Internetseite, Abruf: 5.11.2005

[Sof] SoftwareShield Technologies: *Introduction To The SoftwareShield System*. http://www.softwareshield.com/Introduction/introduction.htm. – Produktinformationsseite, Abruf: 28.10.2005

[Son] Sony DaDC: *SecuROM*. http://www.securom.com/. – Produktinformationsseite, Abruf: 13.02.2006

[Ste90] STEVENS, W. R.: *Unix Network Programming*. Englewood Cliffs, NJ, USA : Prentice-Hall International, 1990

[Ste92] STEVENS, W. R.: *Advanced Programming in the UNIX Environment*. Reading, MA, USA : Addison Wesley, 1992

[StG05] *Strafgesetzbuch*. 2005. – Gesetz i. d. F. v. 1.9.2005

[TDi] TDi GmbH: *Matrixlock - Software Protection System*. http://www.matrixlock.de/. – Produktinformationsseite, Abruf: 3.11.2005

[Tor] TORFALL, Martin: *Obsidium Sofware Protection And Licensing System*. http://www.obsidium.de/show.php?details. – Produktinformationsseite, Abruf: 5.11.2005

[Urh03] *Urheberrechtsgesetz*. 2003. – Gesetz i. d. F. v. 10.9.2003

[Wei] WEIKERT, Alexandra: *Steganographie – Eine andere Art von Verschlüsselung*. http://www.fitug.de/bildung/kongress/stegano.html. – Informationsseite, Abruf: 18.03.2006

[Wika] Wikipedia Foundation Inc.: *Free Software License - Wikipedia*. http://en.wikipedia.org/wiki/Free_software_license. – Internetseite, Abruf: 3.11.2005

[Wikb] Wikipedia Foundation Inc.: *Hauptseite - Wikipedia*. http://de.wikipedia.org/. – Internetseite, Abruf: 26.3.2006

[Wiz] WizzKey Technology b.v.: *Wizzkey API*. http://www.wizzkey.com/wizzkey/api.html. – Produktinformationsseite, Abruf: 5.11.2005

[Woi] WOIJCIK, Bartosz: *PElock homepage*. http://www.pelock.prv.pl/. – Produktinformationsseite, Abruf: 13.02.2006

[Xop] Xope Systems: *Xope Systems - Software Protection, Software Security, Software Reverse Engineering Services, Software Piracy, Crack Proof*. http://www.xopesystems.com/. – Produktinformationsseite, Abruf: 5.11.2005

Aus Gründen der Übersicht habe ich bei den Produktioninformationsseiten und Prospekten, die ich für die Untersuchung der wichtigsten auf dem Markt zu findenden Kopierschutzlösungen verwendet habe, immer nur die Einstiegs-URL angegeben. Die mit diesen verlinkten und logisch zusammenhängenden Seiten innerhalb derselben Homepage haben mir aber auch als Quelle für diese Arbeit gedient.