



(19)
Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 602 00 323 T2** 2005.02.24

(12)

Übersetzung der europäischen Patentschrift

(97) **EP 1 349 033 B1**

(51) Int Cl.⁷: **G06F 1/00**

(21) Deutsches Aktenzeichen: **602 00 323.7**

(96) Europäisches Aktenzeichen: **02 006 935.7**

(96) Europäischer Anmeldetag: **26.03.2002**

(97) Erstveröffentlichung durch das EPA: **01.10.2003**

(97) Veröffentlichungstag

der Patenterteilung beim EPA: **31.03.2004**

(47) Veröffentlichungstag im Patentblatt: **24.02.2005**

(73) Patentinhaber:

Soteres GmbH, 44791 Bochum, DE

(84) Benannte Vertragsstaaten:

**AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT,
LI, LU, MC, NL, PT, SE, TR**

(74) Vertreter:

Betten & Resch, 80333 München

(72) Erfinder:

**Zisowski, Frank, 45470 Mülheim, DE; Armgardt,
Matthias, 45527 Hattingen, DE**

(54) Bezeichnung: **Verfahren zum Schutz der Integrität von Programmen**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung**GEBIET DER ERFINDUNG**

[0001] Die vorliegende Erfindung betrifft die Sicherheit von Computersystemen und insbesondere den Schutz der Integrität und Authentizität eines auf einer Computervorrichtung laufenden Computerprogramms sowie ein Kommunikationsverfahren zwischen wenigstens zwei Modulen eines Computersystems.

BESCHREIBUNG DER ZUGRUNDE LIEGENDEN TECHNIK

[0002] Die Sicherheit von Computersystemen hängt von drei Faktoren ab: Verfügbarkeit, Vertraulichkeit und Integrität. Zudem kommt auf Grund ihrer Bedeutung in einer wachsenden Anzahl von Anwendungen die Authentizität als ein vierter Faktor hinzu.

[0003] Verfügbarkeit ist definiert als Erfordernis, dass ein Computer berechtigten Anwendern seine Dienste innerhalb eines Zeitraums anbietet, der durch die Sicherheitspolitik bestimmt ist. Eine Vertraulichkeitsforderung bestimmt in Übereinstimmung mit der Sicherheitspolitik, dass einige Daten vor nicht berechtigten Anwendern geheim gehalten werden müssen. Integrität ist das Erfordernis, dass Daten und Programme nur von gemäß der Sicherheitspolitik berechtigten Anwendern modifiziert, d. h. erzeugt, geändert und gelöscht werden. Und schließlich erfordert die Authentizität, dass der Empfänger von Daten oder eines Programms in der Lage ist, den Identitätsanspruch des Absenders zu verifizieren.

[0004] Die vorliegende Erfindung geht Probleme an, die durch böswillige Programme, z. B. Trojanische Pferde, Viren und Würmer, verursacht werden, die auf einem Computer eines Anwenders oder in seinem Account ohne sein Einverständnis und meist auch ohne sein Wissen laufen. Es gibt viele Wege, die ein böswilliges-Programm nutzen kann, um in ein Computersystem eines Anwenders einzudringen. Weit verbreitete Betriebssysteme wie z. B. Windows 2000 oder Linux weisen ein Zugriffskontrollsystem auf, das einem Programm auf der Grundlage der Identität des Anwenders, der es startete, Rechte einräumt. Diese Betriebssysteme unterstützen keine programmbasierende Steuerung der Zugriffsrechte. Folglich ist ein böswilliges Programm, sobald es platziert ist, während seiner Ausführung mit den Rechten und Privilegien des Anwenders ausgestattet und kann eine ganze Reihe von böswilligen Operationen ausführen, die von dem Anwender nicht beabsichtigt sind, z. B. Dateien löschen, Programme modifizieren oder störend in die Kommunikation von Programmen eingreifen.

[0005] Die Bedrohungen, die böswillige Programme

darstellen, sind ernst. Einerseits gibt es eine wachsende Zahl von Anwendungen, die sicherheitsrelevante Aufgaben im Account des Anwenders ausführen, z. B. Verschlüsselung, digitale Signaturen, Homebanking, E-Commerce, Finanz- oder andere vertrauliche Transaktionen. Die Sicherheit dieser Anwendungen hängt meist von ihrer korrekten Ausführung im Account des Anwenders oder auf seinem Personalcomputer ab. Andererseits zielen derzeitige Schutzmechanismen wie etwa Firewalls, Virens Scanner und E-Mail-Filter nur darauf ab, ein böswilliges Programm am Eindringen in den Computer zu hindern. Die Praxis zeigt jedoch, dass dieser Schutz oftmals unzureichend ist.

[0006] Der überwiegende Teil computergestützter Arbeitsabläufe, Verfahren und Operationen erfordert eine Verbindung zu einem lokalen Netz (LAN) oder zum Internet. In Anbetracht der kommerziellen Entwicklung stationärer und mobiler E-Dienste und ihrer Förderung durch viele Regierungen wird die Anzahl netzwerkabhängiger Anwendungen zweifellos weiter zunehmen. Das Internet, das eine zentrale Rolle im Nachrichtenfernverkehr spielt, ist ein weltweites, offenes Netz. Infolge seiner Struktur und seines Aufbaus ist diese Kommunikation über das Internet vielfältigen Gefahren ausgesetzt. Daten, die darüber befördert werden, können von anderen Personen als gerade von ihrem Empfänger abgefangen werden, wodurch die Vertraulichkeit der Daten sowie die Privatsphäre der kommunizierenden Parteien verletzt werden. Die Daten können auch von einer nicht berechtigten Partei im Internet modifiziert werden, so dass ihre Integrität verletzt wird, oder falsche Daten können eingefügt werden, so dass ihre Authentizität verletzt wird, wobei die Folgen unvorhersehbar sein können. Schließlich kann ein Angreifer im Internet die Verfügbarkeit beeinflussen.

[0007] Kryptographische Verfahren werden als ein starker Schutz von Vertraulichkeit, Integrität und Authentizität angesehen.

[0008] Um die Vertraulichkeit durchzusetzen verschlüsselt der Absender die Daten auf seinem Computer und sendet sie in verschlüsselter Form an den Empfänger. Die Verschlüsselung verbirgt die Semantik der Daten, und der verwendete Algorithmus stellt sicher, dass ein nicht berechtigter Anwender, der in der Lage ist, diese Daten zu lesen, trotzdem nicht im Stande ist, die beabsichtigte Bedeutung zu rekonstruieren und zu erkennen. Nur der berechtigte Empfänger ist in der Lage, die verschlüsselten Daten mit seinem geheimen Schlüssel zu entschlüsseln. Folglich verhindern kryptographische Verfahren eine Verletzung der Vertraulichkeit.

[0009] Um die Integrität und die Authentizität durchzusetzen, fügt der Absender den Daten eine digitale Signatur an, die sowohl die Daten als auch den Ab-

sender identifiziert. Die Verwendung einer digitalen Signatur ist in **Fig. 2** auf schematische Weise veranschaulicht. Im Schritt S1 erzeugt der Absender den geheimen privaten Schlüssel und einen öffentlichen Schlüssel, auf den vom Empfänger zugegriffen werden kann (Schritt S2). Die digitale Signatur wird dann einer Nachricht unter Verwendung des privaten Schlüssels angefügt. Die Nachricht mit der angefügten digitalen Signatur wird zu dem Empfänger gesendet, der in der Lage ist, die Integrität und Authentizität der Nachricht unter Verwendung des öffentlichen Schlüssels des Absenders zu verifizieren. Ein nicht berechtigter Anwender, der in der Lage ist, die Daten oder ihre Signatur zu modifizieren, ist trotzdem nicht im Stande, sie derart zu modifizieren, dass die Signatur zu den Daten und zu der ursprünglichen Identität des Absenders passt. Der Empfänger der Daten mit der angefügten digitalen Signatur sowie jeder andere Anwender, der über öffentliche Informationen über den Absender verfügt, kann prüfen, ob die Daten der Signatur während der Übertragung modifiziert worden sind, und so den Identitätsanspruch ihres Absenders verifizieren.

[0010] Folglich können kryptographische Verfahren wie die digitale Signatur eine Verletzung der Integrität und Authentizität einer Nachricht, die über ein Kommunikationsnetz gesendet worden ist, nicht verhindern, aber zumindest eine Erfassung ermöglichen. Die Wirksamkeit von kryptographischen Verfahren hängt von zwei Voraussetzungen ab:

1. Der potenzielle Angreifer hat nur zu den verschlüsselten oder signierten Daten und zu anderweitigen öffentlich dargebotenen Informationen Zugang.
2. Anhand dieser Daten ist es rechnerisch nicht möglich, die Daten zu entschlüsseln oder eine digitale Signatur zu fälschen.

[0011] Die zweite Voraussetzung ist auf der Grundlage formaler mathematischer Argumente und der derzeitigen Rechenleistung gerechtfertigt, sie kann mit einem strengen Schluss eines Beweises verifiziert werden.

[0012] Im Gegensatz dazu werden die Bedingungen der ersten Voraussetzung einfach als gegeben hingenommen. Dies ist jedoch der Punkt, an dem böswillige Programme einer genaueren Betrachtung bedürfen.

[0013] Je stärker die mathematischen Eigenschaften kryptographischer Verfahren sind, desto weniger wahrscheinlich ist es, dass sie von einem Angreifer angegriffen werden. Folglich werden andere, weniger sichere Teile des Systems zum bevorzugten Ziel eines Angriffs.

[0014] Um die Vertraulichkeit zu gefährden kann ein

potenzieller Angreifer versuchen, auf direkte Weise Zugang zu den Anwenderdaten zu gewinnen und sie zu stehlen. Eine Zugriffskontrolle ist jedoch ein zuverlässiges Verfahren, und Einwirkungen von Außenstehenden auf ein Computersystem werden oftmals sorgfältig überwacht. Eine für den Angreifer bequemere Methode besteht darin, den Anwender dazu zu bringen, ein böswilliges Programm laufen zu lassen, und sich von diesem Programm die geheimen Daten des geheimen Schlüssels des Anwenders übermitteln zu lassen, als ob dies im Auftrag des Anwenders selbst geschehen würde.

[0015] Eine noch größere Aufmerksamkeit sollte auf böswillige Programme gerichtet werden, wenn der Angreifer versucht, die Integrität und die Authentizität zu gefährden, indem er z. B. eine digitale Signatur fälscht. Dieser geheime Signaturschlüssel, der erforderlich ist, um derartige Signaturen zu erzeugen, ist oftmals auf nicht wiederzugewinnende Art und Weise auf einer Chipkarte gespeichert, die ihrerseits durch eine geheime Nummer, die PIN, die nur dem Anwender bekannt ist, geschützt ist. Folglich muss der Angreifer, um digitale Signaturen zu seinen Gunsten zu erzeugen, die Chipkarte des Anwenders entwenden und ihn zwingen, die PIN zu offenbaren. Im Ergebnis dieser Tat wird der Anwender wahrscheinlich die Chipkarte und folglich alle gefälschten Signaturen ungültig machen. Andererseits kann ein böswilliges Programm, das im Account des Anwenders oder auf seinem Computer läuft, ganz harmlos digitale Signaturen für irgendwelche Daten erzeugen und diese signierten Daten als im Auftrag des Anwenders absetzen.

[0016] Im Internetzeitalter sind böswillige Programme eine weit verbreitete und bequeme Angriffsmittel. Sie können leicht verteilt werden, und mitunter benötigen sie nicht einmal irgendeine explizite Handlung des angegriffenen Anwenders, um die Ausführung zu starten, die Ausführung kann verborgen sein, und sie können jede Operation ausführen, die auch der Anwender ausführen kann, sobald sie platziert sind.

[0017] Aus WO 00/ 21 238 ist ein System für die Verifizierung der Integrität und Berechtigungszuweisung von Software vor der Ausführung auf einer lokalen Plattform bekannt, bei dem die Integrität der Boot-Abbildung einer herunterzuladenden Softwareanwendung verifiziert wird, indem ein neu erzielter Hash-Wert der heruntergeladenen Boot-Abbildung mit einem früher berechneten, sicheren Hash-Wert verglichen wird, der mittels des öffentlichen Schlüssels eines digitalen Signaturmechanismus zugänglich ist.

[0018] Aus US 5,343,527 ist ein Verfahren zum Schützen der Integrität von Softwareanwendungen einer mehrfach aufrufbaren Bibliothek bekannt, bei

dem die Integrität der herunterladbaren Softwarekomponente verifiziert wird, indem der Hash-Wert einer verschlüsselten Klartextdarstellung der Softwarekomponente mit einem entschlüsselten Hash-Wert verglichen wird, der zu einem früheren Zeitpunkt auf der Grundlage der verschlüsselten Klartextdarstellung der Softwareanwendung verschlüsselt worden ist.

[0019] Aus US 6,149,522 ist ein Authentifizierungsprogramm für einen Casinospiel-Datensatz bekannt, bei dem die Integrität eines ladbaren Datensatzes durch Vergleichen eines neu berechneten Hash-Wertes mit einem entschlüsselten Hash-Wert verifiziert wird, der von dem Spielmaschinenhersteller vorgelegt worden ist.

[0020] Die in den oben erwähnten Dokumenten beschriebenen Verfahren liefern keinen Schutz gegen einen Angreifer, der mit Erfolg ein böswilliges Programm im Account eines Anwenders auf dem lokalen Computer, auf welchen die Software oder die Daten heruntergeladen werden, laufen lässt. Der Angreifer kann beispielsweise private Schlüssel, die für den Anwender vorgesehen sind, ohne dessen Wissen entwenden. Die Sicherheit aller drei Systeme, die oben angeführt worden sind und die sich auf eine Kombination aus einem öffentlichen Schlüssel mit einer Hash-Wert-Berechnung verlassen, ist folglich gefährdet.

ZUSAMMENFASSUNG DER ERFINDUNG

[0021] Deshalb geht die vorliegende Erfindung die zwei folgenden Probleme an:

1. die Durchsetzung der Integrität eines Programmcodes während seiner Ausführung,
2. die Durchsetzung der Integrität und Authentizität einer lokalen Kommunikation in einem Computersystem.

[0022] Das erste Problem wird durch ein Verfahren zum Schützen der Integrität eines auf einer Computervorrichtung laufenden Computerprogramms gelöst, das die folgenden Schritte umfasst: Starten der Ausführung des Programms, Erfassen, ob eine unberechtigte Modifikation des Adressraums des Programms aufgetreten ist, Beenden der Programmausführung, falls eine unberechtigte Modifikation erfasst wird, und Fortsetzen der Programmausführung, falls keine solche Modifikation erfasst worden ist.

[0023] Das Verfahren gewährleistet, dass der in den Adressraum des Programms geladene Code während seiner Ausführung, d. h. sowohl der eigentliche Programmcode als auch der damit verknüpfte Code abhängiger Bibliotheken und Module, jener ist, der vom Urheber des Programms vorgesehen war.

[0024] Es könnte eine Fehlernachricht ausgegeben werden, falls eine unberechtigte Modifikation des Adressraums des Programms erfasst wird.

[0025] Um eine unberechtigte Modifikation eines Adressraums des Programms zu erfassen könnte eine kryptographisch starke Hash-Funktion verwendet werden. Dadurch wird sichergestellt, dass die Modifikation nur eines Bits im Adressraum zu einem signifikanten Unterschied in der Hash-Funktion führt, der leicht erfasst werden kann.

[0026] Das Verfahren könnte eine Programminstallationsprozedur aufweisen, die die Schritte der Berechnung einer Hash-Funktion $h(M)$ der Programmmodule und der Berechnung einer Hash-Funktion $h(L)$ der Ausführungsbibliotheken und verknüpften Module umfasst.

[0027] Vorzugsweise sind die Hash-Funktionen in einem vor Modifikationen geschützten Speicherbereich gespeichert. Der Modifikationsschutz könnte durch die Betriebsumgebung über Zugriffskontrollen und/oder Anwender-Accounts verwirklicht werden. Eine unberechtigte Modifikation der Hash-Funktionen kann folglich verhindert werden. Auf Grund der geringen Größe der Hash-Funktionen kann ein Speicherbereich, der vor einer Modifikation zu schützen ist, sehr klein sein, z. B. einige Kilobytes umfassen.

[0028] Dem Programm selbst wird nur während des Installationsvorgangs ein temporärer Zugriff auf den vor Modifikationen geschützten Speicherbereich gewährt.

[0029] Der Erfassungsschritt könnte das Identifizieren von fehlenden, hinzugefügten oder modifizierten Programmmodulen umfassen. Dies vereinfacht die Erfassung eines möglicherweise böswilligen Programms, das für die unberechtigte Modifikation verantwortlich ist.

[0030] Das Verfahren gemäß der Erfindung ist im Anspruch 1 definiert. Es könnte auf jede Art von Computerprogramm angewendet werden, das auf einer Computervorrichtung läuft, von Großrechnern, Tischcomputern, mobilen Vorrichtungen wie Laptops, persönliche digitale Assistenten oder Mobiltelefonen bis hin zu eingebetteten Computern oder Mikrosteuerbausteinen. Die Erfindung ist besonders zweckmäßig für sicherheitsempfindliche Programme wie etwa Digitalsignatur-Programme, Finanztransaktionsprogramme oder dergleichen.

[0031] Außerdem schafft die vorliegende Erfindung sowohl eine Computervorrichtung, die so programmiert ist, dass sie das Verfahren, wie es im Anspruch 1 definiert ist, ausführt, wobei ein Computerprogramm einen Programmcode umfasst, um das Ver-

fahren, wie es im Anspruch 1 definiert ist, auszuführen, als auch ein Speichermedium, auf dem ein Computerprogramm mit einem Programmcode für die Ausführung des Verfahrens gemäß dem Anspruch 1 gespeichert ist.

[0032] Gemäß einem zweiten Aspekt der vorliegenden Erfindung, der im Anspruch 20 definiert ist, wird ein Verfahren für die Kommunikation zwischen wenigstens zwei Modulen einer Computervorrichtung geschaffen, das die folgenden Schritte umfasst: Starten einer Kommunikationssequenz zwischen den wenigstens zwei Modulen, Erzeugen bei jedem der Module für jede Kommunikationssequenz eines privaten Schlüssels und eines öffentlichen Schlüssels für eine digitale Signatur, Verfügbarmachen des öffentlichen Schlüssels eines Moduls für die jeweils anderen Module und Austauschen von Nachrichten mit angefügten digitalen Signaturen zwischen den Modulen unter Verwendung der entsprechenden privaten und öffentlichen Schlüssel.

[0033] Mit dem zweiten Aspekt der vorliegenden Erfindung kann gewährleistet werden, dass zwei oder mehr Programme, Treiber oder andere Module, die in einer einzigen Vorrichtung miteinander kommunizieren, bei einem Zugang von Daten prüfen können, ob die Daten während der Übermittlung modifiziert worden sind und ob die Daten von dem erwarteten Absender stammen. Der zweite Aspekt der Erfindung löst folglich das zweite der weiter oben angeführten Probleme.

[0034] Da die Sequenzen zum Nachrichtenaustausch zwischen verschiedenen Modulen eines Computersystems in den meisten Fällen recht kurz sind, ist es für einen Angreifer sogar schwer, eine weniger starke digitale Signatur zu fälschen. Vorzugsweise werden die Schlüssel der digitalen Signatur beim Beenden jeder Kommunikationssequenz ungültig gemacht.

[0035] Der öffentliche Schlüssel jedes Moduls, für das eine sichere Kommunikation ermöglicht werden soll, könnte auf einer im Voraus definierten festen Adresse in einem vor Modifikationen geschützten Speicherbereich gespeichert sein. In dem vor Modifikationen geschützten Speicherbereich eines Moduls sind außerdem die festen Adressen der öffentlichen Schlüssel der übrigen Kommunikationspartner der sicheren Kommunikation gespeichert. Jedes Modul hat folglich Zugriff zu den entsprechenden privaten Schlüsseln der anderen Module. Die festen Adressen könnten in dem Speicherbereich festcodiert sein, wobei sie dritten Parteien außerhalb des Systems unbekannt sind.

[0036] Jedes Modul könnte einem Integritätsschutzverfahren gemäß dem ersten Aspekt der vorliegenden Erfindung unterworfen sein.

[0037] Gemäß dem zweiten Aspekt der Erfindung wird außerdem eine Computervorrichtung geschaffen, die mehrere funktionale Module umfasst, die miteinander kommunizieren, umfassend: eine Schlüsselerzeugungseinheit, die einen privaten Schlüssel und einen entsprechenden öffentlichen Schlüssel erzeugt, eine Signaturanfügeeinheit, die eine Nachricht mit angefügter digitaler Signatur unter Verwendung des privaten Schlüssels bereitstellt; einen vor Modifikationen geschützten Speicherbereich, der eine feste Adresse für die Speicherung des öffentlichen Schlüssels hat, einen vor Modifikationen geschützten Speicherbereich, in dem die festen Adressen der öffentlichen Schlüssel der anderen Module gespeichert sind, für die eine sichere Kommunikation ermöglicht werden soll, und eine Signaturverifikationseinheit, die die Integrität und Authentizität der von den anderen Modulen empfangenen Nachrichten unter Verwendung des jeweiligen öffentlichen Schlüssels der anderen Module verifiziert.

[0038] Gemäß dem zweiten Aspekt der Erfindung wird außerdem sowohl ein Computerprogramm geschaffen, das einen Programmcode enthält, der das Verfahren des Anspruchs 20 ausführt, als auch ein Speichermedium, in dem ein Computerprogramm für die Ausführung des Verfahrens, das im Anspruch 20 definiert ist, gespeichert ist.

KURZBESCHREIBUNG DER ZEICHNUNG

[0039] Die vorliegende Erfindung sowie weitere ihrer Merkmale und Vorteile werden ohne weiteres aus der folgenden Beschreibung besonderer Ausführungsformen der vorliegenden Erfindung deutlich, die auf die beigefügte Zeichnung Bezug nimmt, worin:

[0040] Fig. 1 eine Prinzipskizze der Ausführung eines Computerprogramms ist, die den Adressraum dieses darstellt;

[0041] Fig. 2 ein schematischer Ablaufplan der Prozedur des Sendens einer Nachricht unter Verwendung einer digitalen Signatur ist;

[0042] Fig. 3 eine Prinzipskizze einer Computervorrichtung ist, in welche die vorliegende Erfindung implementiert werden könnte;

[0043] Fig. 4 eine schematische Veranschaulichung der Verfahrensschritte einer Ausführungsform des ersten Aspekts der vorliegenden Erfindung ist;

[0044] Fig. 5 ein schematischer Ablaufplan der Operationen des Schritts S20 von Fig. 4 ist;

[0045] Fig. 6 ein schematischer Ablaufplan einer Programminstallationsprozedur gemäß einer besonderen Ausführungsform des ersten Aspekts der vorliegenden Erfindung ist;

[0046] Fig. 7 eine schematische Veranschaulichung der Operation der Erzeugung von Hash-Funktionen in einer Ausführungsform des ersten Aspekts der vorliegenden Erfindung ist;

[0047] Fig. 8 eine schematische Veranschaulichung einer Computervorrichtung ist, die mehrere miteinander kommunizierende Module aufweist;

[0048] Fig. 9 eine schematische Veranschaulichung einer Computervorrichtung gemäß einer besonderen Ausführungsform des zweiten Aspekts der vorliegenden Erfindung ist;

[0049] Fig. 10 ein schematischer Ablaufplan ist, der die Verfahrensschritte einer besonderen Ausführungsform des Verfahrens gemäß dem zweiten Aspekt der vorliegenden Erfindung veranschaulicht; und

[0050] Fig. 11 ein schematischer Ablaufplan ist, der die Verfahrensschritte einer Installationsprozedur einer besonderen Ausführungsform gemäß dem zweiten Aspekt der vorliegenden Erfindung veranschaulicht.

A) Erläuterung einiger Ausdrücke

[0051] Computervorrichtung: jede Vorrichtung beliebiger Größe oder Leistungsfähigkeit, die im Voraus definierte Anweisungen in Übereinstimmung mit einem Programmcode ausführt, wobei Großrechner, Tischcomputer, Laptops, persönliche digitale Assistenten, Mobiltelefone, eingebettete Computer und Mikrosteuerbausteine eingeschlossen sind.

[0052] Programmmodul: Teil eines Computerprogramms, das mit anderen Modulen kommuniziert, Unterprogramme oder Subroutinen, Treiber, Schnittstellen eingeschlossen.

[0053] Adressraum: Code eines Programms plus Code aller Bibliotheken oder Module, die mit ihm verknüpft sind oder während der Programmausführung von ihm benutzt werden;
asm-Funktion: Funktion, die die Namen aller Module in einem Adressraum eines Programms zurückgibt; d. h. jene, die von dem Programm selbst geliefert werden und jene, die von der Betriebsumgebung während der Laufzeit geliefert werden.

[0054] Betriebsumgebung: Programm- oder Netzwerkfunktionen, die grundlegende Verwaltungsfunktionen für ein Programm bereitstellen; Beispiele sind PC-Betriebssysteme wie Windows XP oder Linux oder PDA-Betriebssysteme wie Palm-OS.

[0055] Hash-Funktion: mathematische Funktion, die eine festgelegte Datenmenge auf einen mathematischen Wert abbildet, so dass durch die Hash-Funkti-

on eine eindeutige Identifikation der Datenmenge möglich ist.

[0056] Digitale Signatur: Methode der Anwendung kryptographischer Techniken, um die Quelle und die Integrität eines bestimmten elektronischen Dokuments zu bescheinigen.

[0057] Kryptographischer Schlüssel: Mathematischer Wert, der in einem Algorithmus zum Verschlüsseln eines elektronischen Dokuments verwendet wird.

B) Analyse möglicher Bedrohungsszenarien

[0058] Wann immer böswillige Programme als eine Bedrohung angesehen werden, werden Firewalls, Virens Scanner und Zugriffskontrollen als Mittel zum Schutz vorgeschlagen. Jedoch ist der tatsächliche Schutz, den sie bieten, meist nicht ausreichend.

[0059] Eine Firewall kontrolliert die Ports eines Computers, d. h. die Steckplätze, die er verwendet, um Daten in das Netz, mit dem er verbunden ist, zu senden oder aus diesem zu empfangen. Daten, die sich durch ein Netz bewegen, sind in Paketen zusammengefasst. Eine Firewall kann entscheiden:

- ob ein Port für eine Kommunikation offen ist,
- ob es einer Anwendung erlaubt ist, einen Port zu benutzen,
- ob es einem ankommenden Datenpaket erlaubt ist, in den Computer zu gelangen;
- ob es einem abgehenden Datenpaket erlaubt ist, den Computer zu verlassen.

[0060] Böswillige Programme dringen aus einem Netz in Datenpaketen in einen Computer ein – auf gleiche Art und Weise wie alle Daten. Das Vermögen einer Firewall, dieses Eintreten zu verhindern, ist stark eingeschränkt. Ein geschlossener Port kann nicht zum Empfangen oder Senden von irgendwelchen Daten verwendet werden, was trivialerweise zur Folge hat, dass er auch nicht für die Übertragung eines böswilligen Programms verwendet werden kann. Angenommen, ein Port ist offen und einer Anwendung, z. B. einem E-Mail-Programm ist es gewährt, ihn nach ankommenden Datenpaketen abzufragen. Aus der Sichtweise einer Firewall umfasst ein Datenpaket die Netzwerkadresse des Absenders, die Netzwerkadresse des Empfängers und Nutzinformationen. Die Firewall hat keine Ahnung von der Semantik der Nutzinformationen und kann folglich nicht erkennen, ob es sich um einen Teil eines böswilligen Programms handelt. Die Entscheidung, ein ankommendes Datenpaket, das an einen offenen Port adressiert ist, zurückzuweisen oder passieren zu lassen, muss folglich auf der Grundlage der Netzwerkadresse des Absenders getroffen werden. Einige Anwendungen, z. B. Datenbanksysteme erwarten oftmals Daten von

nur einer kleinen Gruppe von Quellen, deren Adressen in einer Firewall spezifiziert werden können und die von der Firewall geprüft werden können. Viele Anwendungen jedoch, wovon einige unentbehrlich sind, z. B. E-Mail-Dienstprogramme und Web-Browser, akzeptieren Daten von allen Quellen, gegebenenfalls unter Ausschluss einiger weniger. Da die Firewall fast jedes Datenpaket passieren lassen wird, ist sie hier offensichtlich nutzlos. Diese Tatsache ist einer der Hauptgründe für die Beliebtheit der Einbettung böswilliger Programme in E-Mail-Nachrichten und HTML-Seiten. Außerdem ist die Netzwerkadresse des Absenders in einem Datenpaket unzuverlässig. Sie kann leicht gefälscht werden und liefert keinen Herkunftsbeweis. Schließlich wird zur Verteilung von böswilligen Programmen immer häufiger eine technische Variante des "Social Engineering" verwendet. Sobald ein böswilliges Programm in einen Computer eingedrungen ist, sucht es nach Adressen von anderen, vermutlich befreundeten Computern, z. B. im Adressbuch des Anwenders, und sendet sich selbst oder andere böswillige Programme an diese. Die Firewall des Empfängers kann nicht erkennen, dass der Computer des Absenders sabotiert wird, und da der böswillige Inhalt gegebenenfalls in eine freundliche Nachricht eingebettet ist, hat der Empfänger selbst nur eine geringe Chance, den böswilligen Inhalt zu bemerken.

[0061] Ein Virens Scanner ist ein Programm, das versucht, böswillige Programme durch Vergleichen der Speicherungen des Computers, z. B. der Dateien oder des Hauptspeicherbereiches, mit seiner eigenen Liste von Mustern derartiger Programme zu erfassen. Diese Erfassungsmethode weist schon auf ihre Nachteile hin. Erstens muss das böswillige Programm, damit es in eine Virens Scannerliste von Mustern aufgenommen wird, schon bekannt und analysiert sein, d. h. jemand muss als Erster durch eine neues böswilliges Programm geschädigt worden sein, an dem der Scanner versagte, und es dann dem Hersteller des Scanners in der Hoffnung gemeldet haben, dass es in der Zukunft entdeckt wird. In Anbetracht dieser Tatsache ist es zweitens unwahrscheinlich, dass ein Virens Scanner ein maßgeschneidertes böswilliges Programm oder eine weiterentwickelte Variante eines bekannten entdeckt. Und schließlich, wenn die Daten in verschlüsselter Form möglicherweise von einer freundlichen, jedoch sabotierten Quelle ankommen, neigen sowohl der Virens Scanner als auch der Anwender dazu, beim Erfassen des in ihnen enthaltenen böswilligen Inhaltes zu versagen.

[0062] Zugriffskontrollsysteme, die angeblich letzten Verteidigungsmechanismen gegen böswillige Programme, die in modernen Betriebssystemen zu finden sind, implementieren ein anwenderbasierendes Schutzschema. Sie schränken den Zugriff von Programmen auf die Betriebsmittel des Computers, z. B. Dateien und Programme, auf der Grundlage der

Identität des Anwenders, dem der Account zugeordnet ist, in dem ein Programm gestartet worden ist, ein. Mit dem Gewähren oder Ablehnen einer Programmzugriffsanforderung schätzen diese Zugriffskontrollsysteme genau genommen die Vertrauenswürdigkeit des Anwenders, in dessen Auftrag das Programm ausgeführt werden soll, und nicht jene des Programms selbst ein. Das Zugriffskontrollsystem kann eine Zugriffsanforderung durch ein vertrauenswürdiges Programm nicht von jener durch ein böswilliges Programm unterscheiden. Wann immer ein böswilliges Programm eine Zugriffsanforderung beantragt, nimmt folglich das Zugriffskontrollsystem an, dass diese Anforderung im Auftrag und mit dem Einverständnis des Besitzers des Accounts erfolgt. Obwohl es Methoden gibt, um das Schutzvermögen von Zugriffskontrollsystemen bei gleichzeitiger Bewahrung ihrer Flexibilität mit programmorientierten Kontrollen zu verstärken, die den potenziellen Schaden, der durch ein böswilliges Programm zugefügt wird, erheblich verringern, sind die derzeitigen Betriebssysteme weit davon entfernt, diese implementieren zu können. Folglich stellen Zugriffskontrollsysteme, wenn sie überhaupt vorhanden sind, in ihrer derzeitigen Form keine wirklichen Beschränkungen der Wirkungsweise eines böswilligen Programms dar.

[0063] Um die Analyse von Bedrohungsszenarien abzuschließen: Böswillige Programme stellen eine Bedrohung der Integrität dar, die von den derzeitigen technischen Schutzmechanismen nur unzureichend angegangen wird. Folglich werden Verwaltungsregeln, d. h. eine Menge von abrätenden Hinweisen, Verboten und Empfehlungen, die versuchen, den Anwender anzuweisen, wie er das Hereinlassen eines böswilligen Programms verhindern kann, oftmals als der wesentliche Schlüssel zum Schutz angesehen. Folglich sind, falls ein böswilliges Programm Schaden anrichtet, bequeme Sündenböcke, denen die Schuld zugeschoben werden kann, schon zur Hand, beispielsweise "der Anwender, der unbestreitbar immer das schwächste Glied in einer Kette von Schutzmaßnahmen ist, hat nicht genügend auf sein Handeln geachtet, so ist es sein eigener Fehler...". Ein ehrlicher Beobachter wird jedoch sofort einwenden, dass diese Politik von dem Anwender verlangt, Entscheidungen zu treffen, für die ihm die Grundlagen völlig fehlen.

[0064] Es ist durchaus trügerisch, alle Sicherheitsbedenken auf der Grundlage der Annahme auszumerzen, dass in der Anwenderumgebung keine böswilligen Programme ausgeführt werden. Es muss sich der Tatsache gestellt werden, dass ein Angreifer meist einen heimlichen Weg finden wird, um ein solches Programm auf den angegriffenen Computer zu bringen. Deshalb muss eine wirksame Strategie zum Schutz der Integrität böswillige Programme berücksichtigen und sich auf technische Kontrollen verlassen, welche die Integrität und Authentizität der Pro-

grammausführung und Kommunikation auch in Gegenwart böswilliger Programme durchsetzen.

C) Voraussetzungen für den Schutz der Integrität und Authentizität

[0065] Bei der programmatischen Formulierung einer Sicherheitspolitik erfordert der Schutz der Integrität und Authentizität, dass Programme nur berechnete Operationen ausführen. Die Strategie, um die Integrität und Authentizität erfolgreich durchzusetzen, umfasst die drei folgenden Schritte:

- i) Identifiziere die Komponenten und Faktoren, von denen eine Programmausführung abhängig ist;
- ii) Analysiere die Schwachstellen und Wege, die die böswilliges Programm ausnutzen kann, um die Integrität und Authentizität zu verletzen;
- iii) Ersinne geeigneter, robuster Schutzmechanismen.

[0066] Fig. 1 zeigt in schematischer Weise die Komponenten, die an der Ausführung eines Programms beteiligt sind. Die Programmausführung umfasst die folgenden Schritte:

- ein Anwender oder Programm startet das Programm mit einem Startbefehl **63**;
- die Betriebsumgebung stellt alle Objektmodule **62** und Ausführungsbibliotheken **61** zusammen, die von dem Programm während der Laufzeit benötigt werden und fügt sie in ihren Adressraum **60** oder virtuellen Speicher ein;
- das Betriebssystem startet die Programmausführung;
- ein Programm stellt seine Eingangsdaten **64** zusammen, wovon ein Teil durch den Anwender geliefert werden kann;
- das Programm kommuniziert mit weiteren Prozessen **68**, Treibern **66**, Diensten **69** und weiteren Informationen übermittelnden Komponenten;
- das Programm erzeugt seine Ausgangsdaten **65**;
- das Programm endet oder wird beendet.

[0067] Die Schritte der Programmausführung können mehrere Male in beliebiger Reihenfolge wiederholt werden.

[0068] Im Folgenden wird jeder der oben aufgeführten Schritte gründlich untersucht, und es werden die notwendigen Voraussetzungen für die Bewahrung ihrer beabsichtigten Semantik bestimmt. Dann werden einige Möglichkeiten der Verletzung ihrer Integrität oder Authentizität diskutiert.

a) Integrität und Authentizität des Programmstarts

[0069] Ein Programm kann durch eine Interaktion

eines Anwenders oder durch einen Befehl eines laufenden Programms gestartet werden. Um eine Verletzung der Integrität oder Authentizität schon in diesem frühen Stadium zu vermeiden, müssen zwei Voraussetzungen erfüllt sein:

[0070] Voraussetzung 1: Der Initiator muss sicherstellen, dass das Betriebssystem das vorgesehene Programm startet.

[0071] Voraussetzung 2: Das für eine Ausführung durch das Betriebssystem ausgewählte Programm muss sicherstellen, dass der Befehl zu seiner Ausführung durch den Anwender autorisiert ist.

[0072] Der eindeutige Name eines auf einem Computer angesiedelten Programms ist durch einen vollständigen Verzeichnispfad und einen Segmentnamen in diesem Verzeichnis definiert. Verletzungen der beiden oben erwähnten Voraussetzungen können bei böswilligen Programmen sehr häufig beobachtet werden.

[0073] Um den Suchpfad für Daten oder die relative Pfadadressierung von Dateien zu benutzen, die von den meisten Betriebssystemen angeboten werden, wird einem böswilligen Programm der Segmentname des vorgesehenen, schon vorhandenen Programms gegeben und es wird an einem Ort gespeichert, der für diesen Namen vor dem Ort abgesucht wird, an dem das beabsichtigte Programm angesiedelt ist. Wenn ein böswilliges Programm einen modifizierenden Zugriff auf das vorgesehene Programm hat, kann es seinen Inhalt direkt ersetzen oder manipulieren.

[0074] Um die Voraussetzung 1 zu erfüllen, muss folglich der Initiator sicherstellen, dass das für die Ausführung durch das Betriebssystem ausgewählte Programm das Programm mit dem beabsichtigten Inhalt ist.

[0075] Die Voraussetzung 2 betrifft das Problem, dass durch ein böswilliges Programm ein Programm gestartet werden kann, um nicht zugelassene Operationen auszuführen. Um diese Voraussetzung zu erfüllen, muss ein Programm selbst oder das Betriebssystem die autorisierten Bedingungen für ein Starten dieses Programms kennen, und es muss möglich sein, diese Bedingungen zu verifizieren. Beispielsweise darf ein Programm, wenn es nur auf interaktive Art durch den Anwender gestartet werden soll, dann nicht ausgeführt werden, wenn es aus einer Befehlsdatei oder einem anderen (möglicherweise böswilligen) Programm heraus gestartet wird.

b) Integrität und Authentizität des Programm-Adressraums

[0076] Der Adressraum eines Programms wird vom Betriebssystem mit seinen statischen Objektmodulen

und mit dynamischen Ausführungsbibliotheken besiedelt.

[0077] Die statischen Module sind Bestandteil der Programmdatei. Wenn die oben angegebene Voraussetzung 1 durchgesetzt ist, kann als selbstverständlich angenommen werden, dass die vorgesehenen Objektmodule im Adressraum des Programms platziert sind.

[0078] Um die Integrität oder Authentizität des Programms in diesem Stadium zu verletzen, muss folglich ein böswilliges Programm eine unbeabsichtigte Menge von Ausführungsbibliotheken in den Adressraum des angegriffenen Programms einfügen. Die Bestimmung der einzufügenden Ausführungsbibliotheken kann rekursiv erfolgen. Die Namen der Ausführungsbibliotheken, von denen das Programm direkt abhängt, sind in der Programmdatei spezifiziert. Wenn eine Bibliothek von weiteren Bibliotheken abhängt, dann sind ihre Namen in der Bibliotheksdatei spezifiziert. Der Auswahlvorgang endet, wenn alle Abhängigkeiten zufriedengestellt sind und keine weiteren Bibliotheken erforderlich sind, d. h. bei der Berechnung der transitiven Hülle der Bibliotheken.

[0079] Da in dem Dateisystem Ausführungsbibliotheken wie gewöhnliche Dateien gespeichert werden, kann ein böswilliges Programm die im vorangehenden Abschnitt beschriebenen Angriffe nutzen, um das Betriebssystem dazu zu bringen, eine nicht beabsichtigte Bibliothek auszuwählen.

[0080] Es kann nun versucht werden, analog zu der oben angegebenen Voraussetzung 1 eine Voraussetzung für Ausführungsbibliotheken zu formulieren. Trotz seiner Effektivität ist dieses Merkmal wahrscheinlich ineffizient. Der Autor eines Programms kennt sicherlich nur die Namen der direkt erforderlichen Ausführungsbibliotheken. Jedoch können ihr Inhalt und mögliche weitere Bibliotheken in der sich ergebenden transitiven Hülle, welche die kleinste Menge von tatsächlich für die Programmausführung auf einem bestimmten Computer erforderlichen Bibliotheken ist, von der Version des Betriebssystems, seinen Dienstpaketen, Korrekturprogrammen und anderen Eigenschaften der Betriebsumgebung abhängen.

[0081] Folglich wird die Bestimmung aller erforderlichen Ausführungsbibliotheken auf dem Computer des Entwicklers wahrscheinlich die Fähigkeit des Programms einschränken, auf einer breiten Palette von Computern ausgeführt zu werden. Es ist deshalb erforderlich, den beabsichtigten Zustand einer Ausführungsbibliothek auf dem Zielcomputer herzustellen. Angesichts des Ziels der Gewährleistung der Programm-Adressraumintegrität und -authentizität an erster Stelle kann jedoch diese Prozedur auf die Ausführungsbibliotheken der transitiven Hülle auf dem Zielcomputer beschränkt werden. Die Untersuchung

führt zu zwei Voraussetzungen:

[0082] Voraussetzung 3: Das Programm muss die Menge der vorgesehenen Ausführungsbibliotheken, die für seine Ausführung erforderlich sind, ermitteln.

[0083] Voraussetzung 4: Das Programm muss sicherstellen, dass das Betriebssystem nur beabsichtigte Ausführungsbibliotheken in seinem Adressraum enthält.

c) Integrität und Authentizität der Kommunikation einer Komponente oder eines Moduls mit anderen Komponenten

[0084] Es wird nun angenommen, dass die richtige Ausführung eines Programms auch von der Kommunikation mit unabhängigen Komponenten oder Modulen außerhalb seines eigenen Adressraums, d. h. mit anderen Programmen, Treibern oder Diensten, abhängt. Die Voraussetzungen für die Integrität und Authentizität von lokal kommunizierenden Komponenten sind genau die gleichen wie bei über Entfernungen kommunizierenden Parteien.

[0085] Voraussetzung 5: Beim Empfang von Daten muss der Empfänger in der Lage sein:

- i) nachzuweisen, ob die Daten während der Übertragung modifiziert worden sind,
- ii) den Urheberanspruch der Daten zu verifizieren, und
- iii) den Absenderanspruch der Daten zu verifizieren.

[0086] Eine Kommunikation zwischen lokalen Komponenten wird gewöhnlich durch das Betriebssystem des Computers gemanagt. Der Absender verwendet einen Systemaufruf, um Daten an einen Empfänger zu senden (oftmals als eine Nachricht bezeichnet), d. h. der Absender weist das Betriebssystem an, die Daten an den Empfänger zu liefern. Das Betriebssystem handelt wie ein Postdienst, d. h. es sammelt die Daten beim Absender ein und liefert sie an einen Empfänger aus. Die Voraussetzung 5i) wird von dem Betriebssystem meist erfüllt. Aber weder Linux noch Windows versorgen den Empfänger mit Informationen über die Identität des Absenders, obwohl sie ihnen zur Verfügung stehen, so dass die Voraussetzung 5iii) von den kommunizierenden Parteien beachtet werden muss. Und schließlich hat das Betriebssystem nichts vorgesehen, um die Identität des Urhebers eines Datendokuments aufzuzeichnen. Folglich sind die miteinander kommunizierenden Parteien auch für diese Voraussetzung (5ii) verantwortlich.

[0087] Die Unterscheidung zwischen dem Absender und dem Urheber von Daten ist wichtig, falls das böswillige Programm die Daten, die zwischen den zu-

gelassenen Parteien ausgetauscht werden, aufzeichnen kann. Angenommen, eine Komponente A führt eine kritische Operation aus, jedoch nur auf Anforderung durch eine bestimmte andere Komponente B. Wenn ein böswilliges Programm die von B an A gesendeten Daten aufzeichnet, kann es sie zu einem späteren Zeitpunkt wieder an A schicken und die Ausführung der kritischen Operation anfordern. Auch wenn diese Daten einen Nachweis der Urheberschaft von A tragen, wird B dennoch nicht in der Lage sein, ihren böswilligen Absender zu entdecken.

[0088] Ein üblicher Vorschlag, um die Voraussetzung 5 zu erzwingen, ist die Verwendung von digitalen Signaturen. Jeder Kommunikationskomponente wird ein individueller, geheimer privater Schlüssel und ein öffentlicher Schlüssel gegeben; der Absender signiert die abgehenden Daten mit seinem privaten Schlüssel, und der Empfänger der Daten kann die Ansprüche des Absenders mit seinem öffentlichen Schlüssel überprüfen. Diese naive Methode weist in der Praxis zwei Probleme auf: das Problem des Senders, den privaten Schlüssel geheimzuhalten, und das Problem des Empfängers, den authentischen öffentlichen Schlüssel des anspruchstellenden Absenders zu erhalten. Der Absender muss seinen privaten Schlüssel an einem bestimmten Ort speichern, z. B. kann er in seine ausführbare Datei codiert sein, oder er kann in einem externen Speicher abgelegt sein, und er muss diesen Schlüssel während seiner Ausführung in seinen Adressraum laden. Auf dem einen oder anderen Weg, falls der Angreifer eine Kopie der ausführbaren Datei oder ein Fragment des Adressraums mit dem privaten Schlüssel für eine rechnerunabhängige Analyse erhalten kann, gibt es zuverlässige Verfahren, um einen privaten Schlüssel in derartigen Objekten aufzuspüren und ihn aus diesen wiederzugewinnen. Wenn der öffentliche Schlüssel des Absenders wie alle Daten an den Empfänger übermittelt wird, dann kann ein böswilliges Programm versuchen, eine Kommunikation mit einer Komponente aufzubauen, indem es ihr seinen eigenen öffentlichen Schlüssel schickt. Der Empfänger wird dann signierte Daten von einer böswilligen Quelle empfangen.

[0089] Die weitere Untersuchung unterscheidet zwischen zwei Klassifikationen einer Kommunikationskomponente:

- i) ihrem ausführbarem Inhalt:
 - Anwenderbetriebsart: Ausführung unterliegt Zugriffskontrolle
 - privilegierte Betriebsart: Ausführung mit uneingeschränktem Zugriff
- ii) ihrer Kooperationsbetriebsart:
 - besonderer Zweck: kommuniziert nur mit bestimmten Komponenten
 - allgemein: kommuniziert mit allen Komponenten.

[0090] Die erste Unterscheidung hat einen gewaltigen Einfluss auf die Folgen einer Unterwanderung der Komponente durch ein böswilliges Programm. Und die zweite entscheidet über die Grenzen der Machbarkeit, eine Dienste liefernde Komponente mit einer Liste zugelassener Client-Komponenten auszustatten.

[0091] Eine Komponente, die in der Anwenderbetriebsart ausführt, unterliegt einerseits den Zugriffskontrollen des Betriebssystems, kann aber andererseits durch diese geschützt sein. Da Zugriffskontrollentscheidungen in der Anwenderbetriebsart mit einem Account verknüpft sind, kann ein böswilliges Programm alle Programme, Daten und weiteren Komponenten gefährden, die zu demselben Account und zu weiteren Accounts gehören, für die Zugriff gewährt wird. Die privilegierte Betriebsart bietet einer Komponente einen vollkommenen Schutz vor Anwenderbetriebsart-Komponenten und gleichzeitig einen uneingeschränkten Zugriff auf alle Betriebsmittel. Negativ gesehen hat dies zur Folge, dass es viel schwerer, wenn nicht unmöglich ist, ein böswilliges Programm, das in der privilegierten Betriebsart ausführt, einzuschränken oder gar zu entdecken. Die meisten Schutzstrategien setzen deshalb voraus, dass ein Account in der privilegierten Betriebsart, z. B. der Administrator oder das Wurzelverzeichnis, keine böswilligen Komponenten enthält. Dennoch muss es immer einige Komponenten geben, die eine privilegierte Ausführungsumgebung verlangen, wobei die Entscheidung über die Ausführungsumgebung vieler Komponenten dem Ermessen der Entwickler eines Betriebssystems überlassen ist. Beispielsweise führt Windows 2000 alle Gerätetreiber und Dienste in einer privilegierten Betriebsart aus.

[0092] Die Komponenten des Betriebssystems sind meist mit dem Ziel konzipiert, auf Anforderung einer anderen Komponente einen Dienst zu liefern. Diese allgemeinen Komponenten kümmern sich weder um die Identität des Clients noch, zumindest bisher nicht, um die Integrität einer Kommunikation. Es ist klar, dass, sobald die Operation einer Komponente kritisch ist, dieses Verhalten eine Gefahr für die Sicherheit darstellt. Falls eine Anwendung die Kommunikation mehrerer unabhängiger Komponenten für besondere Zwecke erfordert, z. B. ein Programm und einen Treiber, dann können solche Komponenten so beschaffen sein, dass sie allein darauf Acht haben, dass sie nur mit im Voraus definierten Komponenten kommunizieren und die Forderungen der oben angegebenen Voraussetzung 5 sicherstellen.

d) Integrität und Authentizität der Ein- und Ausgangsdaten

[0093] Dieses wichtige Anliegen ist außerhalb des Geltungsbereiches dieser Erfindung.

e) Integrität und Authentizität der Programmbeendigung

[0094] Dieses wichtige Anliegen ist außerhalb des Geltungsbereiches der Erfindung.

[0095] Beide Aspekte der vorliegenden Erfindung nutzen Verfahren und Schutzmaßnahmen, die in großen Netzen angewendet werden, für den Schutz der Integrität und Authentizität der Ausführungsumgebung eines einzigen Computers. Diese Verfahren umfassen:

- kryptographisch starke Hash-Funktionen,
- digitale Signaturen,
- Erzeugen von kryptographischen Schlüsseln während der Laufzeit,
- vor Modifikationen geschützte Speicherung.

[0096] Die vorliegende Erfindung strebt an, Lösungen für die weiter oben aufgestellten Voraussetzungen 1 bis 5 aufzuzeigen.

D) Erster Aspekt der vorliegenden Erfindung: Schutz der Integrität und Authentizität des Adressraums eines Computerprogramms

[0097] Fig. 4 ist ein Ablaufplan, der in schematischer Weise die Operationsschritte eines Verfahrens zum Schützen der Integrität eines Computerprogramms veranschaulicht, das auf einer Computervorrichtung gemäß dem ersten Aspekt der vorliegenden Erfindung läuft. Die Computervorrichtung könnte, wie an früherer Stelle erläutert worden ist, ein Computersystem beliebiger Größe oder beliebiger Leistungsfähigkeit sein. Ein Beispiel für eine solche Computervorrichtung **50** ist in schematischer Weise in Fig. 3 dargestellt. Sie umfasst eine Zentraleinheit **51**, die aus einem oder aus mehreren Mikroprozessoren gebildet sein könnte, eine Eingabe/Ausgabe-Einheit **52**, die als Schnittstelle sowohl zu Eingabe/Ausgabe-Vorrichtungen wie etwa einem Datenübertragungsglied, einer Tastatur, einer Maus oder einem Drucker, als auch zu einer Speichervorrichtung **53** dient. Die Letztere umfasst einen kleinen Bereich **54**, der gegen Modifikationen, beispielsweise durch das Betriebssystem des Computers, geschützt ist. Die in Fig. 4 gezeigte Prozedur beginnt mit dem Start der Programmausführung im Verfahrensschritt S10. Im nachfolgenden Verfahrensschritt S20 wird dann während der Laufzeit geprüft, ob eine unberechtigte Modifikation des Adressraums des Programms stattgefunden hat oder nicht. Wenn dies nicht der Fall ist, sind Authentizität und Integrität des Adressraums verifiziert und die Programmausführung wird mit dem Schritt S30 fortgesetzt. Wenn im Schritt S20 eine unberechtigte Modifikation des Adressraumes erfasst worden ist, wird im Schritt S40 eine Fehlernachricht gesendet, und die Programmausführung wird im nachfolgenden Schritt S41 beendet.

[0098] Gemäß einer besonderen Ausführungsform identifiziert der Erfassungsschritt S20 fehlende, hinzugefügte oder modifizierte Programmmodule oder Ausführungsbibliotheken im Adressraum des Programms.

[0099] Der Erfassungsschritt S20 wird mit Bezug auf einen Ablaufplan von Fig. 5 ausführlicher erläutert. Nach dem Starten einer Programmausführung werden im Verfahrensschritt S21 die Module und die Ausführungsbibliotheken des Programms P erhalten. Dazu könnte eine asm-Funktion verwendet werden, wie weiter unten ausführlicher erläutert wird. Im folgenden Schritt S22 werden die Hash-Funktionen der Module und Ausführungsbibliotheken berechnet. Diese im Schritt S22 erhaltenen Hash-Funktionen werden dann im Schritt S23 mit zuvor gespeicherten Referenz-Hash-Funktionen des Moduls und der Ausführungsbibliotheken verglichen, die zu einem früheren Zeitpunkt bei der Programminstallation berechnet und in einen gegen Modifikationen geschützten Speicherbereich (Bereich **54** in Fig. 3 oder Bereich **121** in Fig. 7) gespeichert worden sind. Eine Hash-Funktion $h(D)$ eines Dokuments oder einer Datenmenge identifiziert das Dokument oder die Datenmenge D eindeutig. Obwohl es theoretisch möglich ist, dass zwei verschiedene Dokumente die gleichen Hash-Funktionen haben, ist die Wahrscheinlichkeit, dass die Hash-Funktion $h(D')$ eines modifizierten Dokuments D' jener des Originaldokuments völlig gleich ist, extrem klein.

[0100] Andererseits ist die Hash-Funktion ein mathematischer Wert mit einer Bitlänge von ungefähr 100 oder 160 Bits. Um die Integrität und Authentizität des Programm-Adressraums zu schützen ist daher der Modifikationsschutz eines sehr kleinen Speicherbereichs ausreichend.

[0101] Der Programminstallationsvorgang für die Berechnung der Hash-Funktion des Adressraums des Programms ist in schematischer Weise auf dem Ablaufplan von Fig. 6 gezeigt.

[0102] Nach dem Starten der Programminstallation im Verfahrensschritt S50 wird die Hash-Funktion $h(M)$ der Menge der Module des Programms P im nachfolgenden Verfahrensschritt S51 berechnet. Dann werden die Namen der von dem Programm P benutzten Ausführungsbibliotheken unter Verwendung einer so genannten asm-Funktion erhalten, die von vielen Betriebssystemen bereitgestellt wird. Anschließend, im Schritt S53, wird die Hash-Funktion $h(L)$ der Ausführungsbibliotheken berechnet, und im Schritt S54 erfolgt eine Speicherung der Hash-Funktionen $h(M)$ und $h(L)$ oder einer kombinierten Hash-Funktion in dem vor Modifikationen geschützten Speicherbereich des Computersystems. Dann wird der Schreibzugriff auf den geschützten Speicherbereich im Schritt S55 aufgehoben, und die

Installationsprozedur wird beendet.

[0103] Fig. 7 zeigt in schematischer Weise die Prozedur zur Berechnung der Hash-Funktionen des Programm-Adressraums, entweder während der in Fig. 6 gezeigten Installationsprozedur oder während der in Fig. 4 gezeigten Programmausführung. Die Ausführungsumgebung **100** umfasst eine Dienstfunktion **101**, die eine asm-Funktion **102**, eine Hash-Funktion **103** bereitstellt und einen Speicher verwaltet, der einen vor Modifikationen geschützten Speicherbereich **121** enthält.

[0104] Einerseits gibt die asm-Funktion **102** sowohl die Namen aller Module $M_1, \dots, M_k$ im Adressraum **115** des Programms P als auch die Namen aller Ausführungsbibliotheken $L_1, \dots, L_m$ zurück. Dann wird eine Hash-Funktion $h(M)$ der Menge der Module und eine Hash-Funktion $h(L)$ der Ausführungsbibliotheken unter Verwendung der Hash-Funktionsfunktionalität **103** berechnet. Alternativ könnten eine kombinierte Hash-Funktion für Module und Ausführungsbibliotheken verwendet werden.

[0105] Im Folgenden wird ein Beispiel für das Verfahren des ersten Aspekts der vorliegenden Erfindung ausführlich erläutert.

[0106] Die Grundvoraussetzungen sind:

- ein kleiner Bereich eines Speichers, der gegen Modifikationen geschützt werden kann, d. h. der auf Nur-Lesen gesetzt werden kann;
- eine asm-Funktion, welche die Namen der während der Laufzeit in den Adressraum eingebrachten Module zurückgibt,
- eine kryptographisch starke Hash-Funktion, z. B. 160 Bits

[0107] Prozedur für ein Programm P:

[0108] Phase 1: Vorbereitung von P

- i) Es sei $M = \{M_1, \dots, M_k\}$ die Menge der Module, die von P geliefert werden
- ii) berechne mit der Hash-Funktion h die Menge der Werte $h(M) = \{h(M_1), \dots, h(M_k)\}$

[0109] Phase 2: Installation von P

- i) kopiere P und h an ihren Bestimmungsort
- ii) gewähre P Schreibzugriff auf den vor Modifikationen geschützten Speicherbereich
- iii) bringe $h(M)$ in den vor Modifikationen geschützten Speicherbereich ein
- iv) starte Ausführung von P
- v) verwende die asm-Funktion, um $L = \{L_1, \dots, L_m\}$, die Teilmenge der Namen der Ausführungsbibliotheken, die in den Adressraum von P auf dem Zielcomputer eingebracht worden sind, zu erhalten

ten

- vi) berechne mit h die Menge der Werte $h(L) = \{h(L_1), \dots, h(L_m)\}$
- vii) bringe $h(L)$ in den vor Modifikationen geschützten Speicherbereich ein
- viii) annulliere den Schreibzugriff von P auf den vor Modifikationen geschützten Speicherbereich
- ix) beende Ausführung von P

[0110] Phase 3: Ausführung von P

- i) starte Ausführung von P
- ii) verwende die asm-Funktion, um die Menge der Namen aller Module M' und aller Ausführungsbibliotheken L' im Adressraum von P zu erhalten
- iii) berechne mit h die Menge der Werte $h(L')$ und $h(M')$
- iv) lies $h(M)$ und $h(L)$ aus dem vor Modifikationen geschützten Speicher aus
- v) teste, ob $h(L') \cup h(M') = h(M) \cup h(L)$
- a) beende Programm P, wenn der Test negativ ist
- b) setze mit der normalen Ausführung des Programms fort, wenn der Test bestanden worden ist

[0111] Wie zuvor erläutert worden ist, identifiziert gemäß einer Variation des Verfahrens der Test, der in der Phase 3 im Schritt (v) ausgeführt wird, fehlende Module, hinzugefügte Module und modifizierte Module. Wenn in dem vor Modifikationen geschützten (Nur-Lese-)Speicherbereich auch der Name und der Speicherort der Module gespeichert sind, dann kann der oben erwähnte Test erweitert werden, um Manipulationen dieser Daten zu erfassen, z. B. wenn ein Modul umbenannt oder an einen anderen Speicherort verschoben worden ist.

[0112] Der erste Aspekt der Erfindung geht Gefahren an, die durch böswillige Programme hervorgerufen werden. Diese Programme werden im Account oder in der Umgebung des Anwenders ausgeführt, und folglich erhalten sie Rechte und Privilegien, die von der Betriebsumgebung allen Programmen gewährt werden, die in diesem Account ausgeführt werden. Ein böswilliges Programm könnte jedoch auch unbeabsichtigte und daher auch von dem Anwender nicht autorisierte Operationen ausführen. Da die Betriebsumgebung nicht in der Lage ist, ein vertrauenswürdiges Programm von einem böswilligen zu unterscheiden, können Programme mit entsprechenden Sicherheitsanforderungen nicht von der Betriebsumgebung abhängen, um gegen böswillige Programme geschützt zu sein. Derartige Programme müssen deshalb, trotz der Unterstützung durch die Betriebsumgebung, selbst einen Schutz sicherstellen.

[0113] Im Folgenden werden einige typische Angriffe eines böswilligen Programms auf das zu schützende Programm und das Schutzverhalten des Verfahrens gemäß dem ersten Aspekt der vorliegenden Erfindung diskutiert:

1. Angriff: Ein Modul, das in dem Programm-Adressraum erforderlich ist, wurde aus diesem entfernt.

Schutz: Der Test in der Phase 3/Schritt (v) ermittelt ein fehlendes Element in der während der Laufzeit berechneten Menge der Module.

2. Angriff: Ein Modul, das in dem Adressraum nicht erforderlich ist, wurde ihm hinzugefügt.

Schutz: Der Test in der Phase 3/Schritt (v) ermittelt ein zusätzliches Element in der während der Laufzeit berechneten Menge der Module.

3. Angriff: Ein im Adressraum erforderliches Modul wurde modifiziert.

Schutz: Der Test in der Phase 3/Schritt (v) ermittelt sowohl ein fehlendes als auch ein zusätzliches Element in der während der Laufzeit berechneten Menge der Module.

[0114] Die Erfassung verschiedener Angriffe hängt von dem Vermögen ab, die oben definierte Schutzprozedur mit dem Ziel einer Programmzuverlässigkeit auszuführen. Dies hat folgende Auswirkungen:

1. Die Liste der Module im Adressraum des Programms wird durch die Betriebsumgebung verwaltet. Die Liste darf von dem Angreifer nicht manipuliert werden.

2. Die Verwendung einer Hash-Funktion verringert die erforderliche Größe des gegen Modifikationen geschützten Speicherbereichs auf einen geringen Betrag. Und die Voraussetzung einer kryptographisch starken Hash-Funktion stellt sicher, dass einem Modul eindeutig sein Hash-Wert zugeordnet werden kann.

3. Die Installation erfolgt in den Account sowohl der Module, die vom Urheber des Programms geliefert werden, als auch der Module, die vom Hersteller der Betriebsumgebung des zu schützenden Systems geliefert werden. Einerseits gibt dies dem Programm die Flexibilität, auf jeder kompatiblen Version der Betriebsumgebung zu laufen. Andererseits erfordert eine Aktualisierung irgendeines Moduls im Programm-Adressraum eine Wiederholung der Vorbereitungsphase oder der Installationsphase für das betreffende Modul.

4. Der vor Modifikationen geschützte Speicher, der die Referenz-Hash-Werte speichert, muss die folgenden Eigenschaften aufweisen:

a) Das Programm selbst muss einen zeitweiligen modifizierenden Zugriff auf den vor Modifikationen geschützten Speicher haben, um während der Installationsphase die Referenz-Hash-Werte zu speichern.

b) Kein anderes Programm sollte die Referenz-Hash-Werte modifizieren. Folglich darf ein möglicherweise böswilliges Programm keinen modifizierenden Zugriff zu dem vor Modifikationen geschützten Speicher erhalten. Ein Nur-Lese-Zugriff braucht nicht untersagt zu werden, da die gespeicherten Hash-Werte nicht geheim gehalten

werden müssen.

In einer Betriebsumgebung dieses Typs kann der vor Modifikationen geschützte Speicher mit Zugriffskontrollen und Anwender-Accounts verwirklicht werden.

5. Kein Programm kann seine eigene Integrität prüfen. Dies ist einfach zu entdecken, da ein Angreifer das Modul, das den Entscheidungstest (Phase 3/Schritt (v)) ausführt, durch ein solches ersetzen kann, bei dem dieser Test weggelassen ist und das folglich das gesamte Schutzschema unbrauchbar macht. Deshalb muss dieser Test von einer dritten Partei ausgeführt werden, deren Integrität durch den mutmaßlichen Angreifer nicht gefährdet werden kann. In einem Betriebssystem kann dies durch einen geschützten Dienst dieses Betriebssystems oder der Betriebsumgebung verwirklicht werden.

[0115] Das Verfahren gemäß dem ersten Aspekt der vorliegenden Erfindung ermöglicht folglich die effektive Erfassung von Manipulationen des Adressraums eines Programms.

E) Zweiter Aspekt der Erfindung

[0116] Der zweite Aspekt der vorliegenden Erfindung betrifft die Kommunikation zwischen zwei oder mehr Modulen oder Kommunikationsparteien A, B,... in einem Computersystem. Es wird angenommen, dass die Module A, B... über ein Nachrichtenübermittlungssystem kommunizieren, das von vielen (möglicherweise böswilligen) Modulen benutzt wird.

[0117] Fig. 8 zeigt in schematischer Weise eine Computervorrichtung **200** mit den Modulen A, B, C und D. Selbstverständlich ist jede andere Anzahl Module möglich. Die Module kommunizieren miteinander über ein Nachrichtenübermittlungssystem, das beispielsweise durch die Betriebsumgebung verwaltet wird. Wenn ein böswilliges Programm in der Lage ist, Daten zu manipulieren, die zwischen den miteinander kommunizierenden Parteien gesendet werden, dann können offensichtlich Authentizität und Integrität nicht durchgesetzt sein. Digitale Signaturen bieten jedoch dem Empfänger von Daten die Möglichkeit, eine Verletzung dieser Eigenschaften zu erfassen. Angenommen, es werden anerkannte Schemata digitaler Signaturen mit entsprechenden Schlüsseln verwendet, dann hängt ihre Zuverlässigkeit auf der Absenderseite von ihrem Vermögen ab, jede andere Partei daran zu hindern, digitale Signaturen mit dem geheimen privaten Schlüssel zu erzeugen, während sie auf der Empfängerseite vom Besitz des authentischen öffentlichen Schlüssels des Senders abhängt.

[0118] Das Kommunikationsverfahren gemäß dem zweiten Aspekt der vorliegenden Erfindung ist in schematischer Weise auf dem Ablaufplan von Fig. 10 veranschaulicht. Im Schritt S80 startet eine

Kommunikationssequenz zwischen zwei Modulen. Dann wird in jedem der zwei Module A, B, die an dieser Kommunikationssequenz beteiligt sind, während der Laufzeit ein Schlüsselpaar (privater, geheimer Schlüssel und öffentlicher Schlüssel) erzeugt (Schritt S81). Der öffentliche Schlüssel wird dann dem jeweils anderen Modul, das an der Kommunikationssequenz beteiligt ist, zur Verfügung gestellt, d. h. dass der öffentliche Schlüssel des Moduls A dem Modul B zu Verfügung gestellt wird und umgekehrt. Es ist dann möglich, auf der Grundlage des Paares aus privatem und öffentlichem Schlüssel Nachrichten mit einer angefügten digitalen Signatur zwischen A und B in jeder Richtung auszutauschen (Schritt S83). Bei dieser Prozedur ist es nicht möglich, eine Verletzung der Integrität oder der Authentizität der Nachricht (d. h. eine Änderung des Inhalts oder des vermutlichen Urhebers oder Absenders der Nachricht) zu verhindern, es ist jedoch möglich, jede solche Verletzung wie im Fall der Verwendung digitaler Signaturen in großen Netzwerken zu erfassen. Da die Dauer einer Kommunikationssequenz zwischen zwei Modulen eines Computersystems recht kurz ist und für jede neue Kommunikationssequenz ein neuer Satz kryptographischer Schlüssel verwendet wird, ist es nicht möglich, eine rechnerunabhängige Analyse auszuführen, um den privaten Schlüssel herauszufinden.

[0119] Eine Installationsprozedur für ein Modul gemäß einer besonderen Ausführungsform des zweiten Aspekts der vorliegenden Erfindung ist in schematischer Weise auf dem Ablaufplan von **Fig. 11** veranschaulicht. Nach dem Start der Modulinstallation im Schritt S90 wird im Schritt S91 ein vor Modifikationen geschützter Speicherbereich definiert. Dann wird eine feste Adresse für den öffentlichen Schlüssel dieses Moduls in den vor Modifikationen geschützten Speicherbereich festcodiert (Schritt S92), und anschließend, im Schritt **93**, werden auch die festen Adressen der öffentlichen Schlüssel in dem jeweiligen vor Modifikationen geschützten Speicherbereich der Kommunikationspartner (Module, zu und von denen eine sichere Kommunikation möglich sein sollte) in den vor Modifikationen geschützten Speicherbereich gespeichert. Jedes Modul kann folglich auf der Grundlage der festcodierten Adressinformationen einfach auf die öffentlichen Schlüssel der Kommunikationspartner zugreifen. Auf Grund des Modifikationsschutzes können diese Adressinformationen nicht durch ein böswilliges Programm modifiziert werden.

[0120] Ein Beispiel für eine Computervorrichtung gemäß dem zweiten Aspekt der Erfindung ist in schematischer Weise in **Fig. 9** gezeigt. Zwecks Vereinfachung sind nur zwei Module A, B gezeigt. Jedes dieser Module enthält eine Schlüsselerzeugungseinheit **212**, um während der Ausführungszeit ein entsprechendes Paar aus einem privaten und einem öffentlichen Schlüssel zur Verwendung mit der digitalen Si-

gnatur zu erzeugen. Jedes der Module A, B umfasst außerdem sowohl eine Signaturanfügeeinheit **211**, um einer Nachricht unter Verwendung des privaten Schlüssels eine digitale Signatur hinzuzufügen, als auch eine Signaturverifikationseinheit **210**, um unter Verwendung des öffentlichen Schlüssels die digitale Signatur zu verifizieren. Ein Adressbereich **213** jedes der Module umfasst einen vor Modifikationen geschützten Bereich **214**, in dem der öffentliche Schlüssel des entsprechenden Moduls und die festen Adressen aus den vor einer Modifikation geschützten Speicherbereichen der anderen Module festcodiert sind.

[0121] Um eine rechnerunabhängige Analyse durch einen Angreifer sinnlos zu machen und die "Man-In-The-Middle Attack" sowie verschiedene andere Angriffe zu verhindern, setzt eine besondere Ausführungsform des zweiten Aspekts der Erfindung folgende Eigenschaften ein:

1. Der private Schlüssel und der öffentliche Schlüssel, die für die Erzeugung und Verifizierung von digitalen Signaturen benutzt werden, werden für jede Kommunikationssequenz vom Absender während der Ausführungszeit wiederholt erzeugt.
2. Jeder Kommunikationspartei ist eine feste Adresse in dem vor Modifikationen geschützten Speicherbereich für die Ablage ihres aktuellen öffentlichen Schlüssels zugewiesen.
3. Der Name und die Adresse, wie unter 2. spezifiziert, jeder Partei, mit der ein Modul zu kommunizieren beabsichtigt, sind in dem Modul festcodiert. Für diese Merkmale müssen die folgenden Grundvoraussetzungen erfüllt sein:
 - ein kleiner Speicherbereich, der gegen Modifikationen geschützt werden kann, d. h. der auf Nur-Lesen gesetzt werden kann;
 - eine asm-Funktion, welche die Namen der während der Laufzeit in dem Adressraum abgelegten Module zurückgibt,
 - eine kryptographisch starke Hash-Funktion h
 - das Vermögen, digitale Signaturen zu erzeugen und zu verifizieren.

[0122] Ein Beispiel für das Kommunikationsverfahren gemäß dem zweiten Aspekt der vorliegenden Erfindung wird nun ausführlicher erläutert.

[0123] Vorbereitung des Moduls A:

- i) weise A eine feste Adresse in dem vor Modifikationen geschützten Speicherbereich für die Ablage seines öffentlichen Schlüssels zu und codiere diese in A fest
- ii) codiere in A die festen Adressen der öffentlichen Schlüssel jedes Moduls, mit dem A zu kommunizieren beabsichtigt, fest

[0124] Kommunikation zwischen zwei Modulen A

und B (kann auf n Parteien erweitert werden):

[0125] Phase 1: Allgemeine Vorbereitung

i) Jedes Modul wird installiert und startet, wie weiter oben in Bezug auf die Phasen 1 bis 3 des Beispiels des ersten Aspekts der vorliegenden Erfindung beschrieben worden ist.

[0126] Phase 2: Vorbereitung der Kommunikation

- i) A und B signalisieren einander die Bereitschaft, eine geschützte Kommunikation aufzubauen
- ii) A erzeugt ein Paar Schlüssel und schreibt den öffentlichen Schlüssel auf seine Adresse in dem vor Modifikationen geschützten Speicherbereich (jedes Kommunikationsmodul führt diesen Schritt aus)
- iii) A nimmt Bs öffentlichen Schlüssel aus dem vor Modifikationen geschützten Speicherbereich auf (jedes Kommunikationsmodul führt diesen Schritt aus).
- iv) Geschützte Kommunikation: A und B tauschen Nachrichten aus, denen digitale Signaturen angefügt worden sind
- v) Optional: Sowohl A als auch die anderen Module machen ihren öffentlichen Schlüssel in dem vor Modifikationen geschützten Speicherbereich ungültig, z. B. mit einem vereinbarten sinnlosen Wert wie etwa null.

[0127] Das Vermögen eines böswilligen Programms, in eine Kommunikation störend einzugreifen, ist durch das Kommunikationsmodell einer Betriebsumgebung bestimmt. Ein wesentlicher Schlüssel zur Wirksamkeit des Schutzes des Verfahrens gemäß der Erfindung ist die Bereitstellung eines kleinen vor Modifikationen geschützten Speicherbereichs, der die öffentlichen Schlüssel der Kommunikationsparteien speichert. Er muss die folgenden Eigenschaften aufweisen:

- a) Die Kommunikationspartei selbst muss nach der Absichtserklärung, eine geschützte Kommunikation ausführen zu wollen, einen zeitweiligen modifizierenden Zugriff auf den vor Modifikationen geschützten Speicherbereich für die Speicherung ihres während der Ausführungszeit erzeugten, aktuellen öffentlichen Schlüssels erhalten.
- b) Kein anderes Programm sollte in der Lage sein, einen öffentlichen Schlüssel zu modifizieren. Folglich darf ein möglicherweise böswilliges Programm keinerlei modifizierenden Zugriff zu dem vor Modifikationen geschützten Speicher erhalten (ein Nur-Lese-Zugriff braucht für den gespeicherten öffentlichen Schlüssel nicht verhindert zu werden (braucht nicht geheim gehalten zu werden)).

[0128] Wie im Fall des ersten Aspekts der Erfindung kann dieser Speicherungstyp in einer Betriebsumge-

bung mittels Zugriffskontrollen und/oder Anwender-Accounts verwirklicht werden.

[0129] Im Folgenden werden einige mögliche Angriffe und ihre Folgen für das beschriebene System diskutiert:

1. "Man-In-The-Middle Attack" Dieser Angriff setzt voraus, dass der Angreifer in der Lage ist, die öffentlichen Schlüssel einer oder mehrerer Parteien durch seine eigenen öffentlichen Schlüssel zu ersetzen, für welche er auch die passenden privaten Schlüssel besitzt.

Schutz: Kein möglicherweise böswilliges Programm hat einen modifizierenden Zugriff auf den Speicherbereich für die öffentlichen Schlüssel.

2. Angriff: Die Authentizität eines öffentlichen Schlüssels hängt nun von der Tatsache ab, dass ein Programm die korrekte Adresse des öffentlichen Schlüssels seines Kommunikationspartners besitzt. Die Adresse ist in dem Programm festcodiert. Ein böswilliges Programm kann folglich den Programmcode modifizieren und die ursprüngliche Adresse durch eine gefälschte ersetzen.

Schutz: Eine solche Modifikation wird zu einem Fehler beim Integritätstest des Adressraums des Programms führen, wie im ersten Aspekt der vorliegenden Erfindung definiert worden ist.

3. Angriff: In die Kommunikationssequenz wird eine Nachricht eingefügt.

Schutz: Die digitale Signatur deckt die Störung der Authentizität der Nachricht auf.

4. Angriff: Eine Kopie des Programmcodes der Kommunikationsparteien wird zur rechnerunabhängigen Analyse über das Netz geschickt.

Schutz: Der Code umfasst keine geheimen Daten.

5. Angriff: Der während der Laufzeit erzeugte private Schlüssel wird nur im virtuellen Speicher der Partei gespeichert. Der Angreifer fragt auf der Suche nach dem privaten Schlüssel einer kommunizierenden Partei den virtuellen Speicher ab.

Schutz: Für jede Kommunikationssequenz werden neue Schlüssel erzeugt. Die Länge der Schlüssel ist so gewählt, dass eine rechnerunabhängige Analyse nicht genügt, um die Schlüssel in Gefahr zu bringen. Gewöhnlich ist der virtuelle Speicher für andere Programme gesperrt und ein Abfragen ist unmöglich. Andererseits kann auf Grund der kurzen Lebensdauer der Schlüssel ihre Länge sehr gering gewählt werden, wodurch die Wirksamkeit von musterorientierten Abfragen zu nichte gemacht wird.

[0130] Eine geschützte Kommunikation kann sogar noch bei Anwesenheit von böswilligen Programmen erfolgen. Obwohl ein böswilliges Programm in eine Kommunikation störend eingreifen kann, sofern die zu Grunde liegenden Kommunikationsmodelle dies erlauben, können Verletzungen der Integrität und Authentizität der Kommunikation mit dem Verfahren der

vorliegenden Erfindung erfasst werden.

[0131] Eine geschützte Programmausführung und eine geschützte Kommunikation gemäß der vorliegenden Erfindung können in vielen Betriebsumgebungen ausgeführt werden, die auf diese Weise einen grundlegenden Schutz bieten. Die Ergebnisse sind sowohl auf allgemein verfügbare Betriebssysteme als auch auf systemgebundene Betriebsumgebungen, z. B. auf mobilen Vorrichtungen oder dergleichen, anwendbar.

Bezugszeichenliste

50	Computervorrichtung
51	Zentraleinheit
52	Eingabe/Ausgabe-Einheit
53	Speicher
54	vor Modifikationen geschützter Speicherbereich
60	Programm-Adressraum
61	Ausführungsbibliotheken
62	Objektmodule
63	Startbefehl
64	Eingangsdaten
65	Ausgangsdaten
66	Treibervorrichtung
67	Hardware-Vorrichtung
68	Software-Prozess
69	Dienstfunktion
100	Betriebsumgebung
101	Dienstfunktion
102	asm-Funktion
103	Hash-Funktion
110	Programm P
115	Adressraum von P
120	Speichervorrichtung
121	vor Modifikationen geschützter Speicherbereich
200	Computervorrichtung
210	Signaturverifikationseinheit
211	Signaturanfügungseinheit
212	Schlüsselerzeugungseinheit
213	Speichervorrichtung
214	vor Modifikationen geschützter Bereich

Patentansprüche

1. Verfahren zum Schützen der Integrität eines auf einer Computervorrichtung laufenden Computerprogramms, das die folgenden Schritte umfasst:

- Starten der Ausführung des Programms,
 - Erfassen, ob eine unberechtigte Modifikation des Programms aufgetreten ist; und
 - Beenden der Programmausführung, falls eine unberechtigte Modifikation erfasst wird, und Fortsetzen der Programmausführung, falls keine solche Modifikation erfasst worden ist;
- wobei das Verfahren **dadurch gekennzeichnet** ist, dass der Erfassungsschritt ferner die folgenden

Schritte umfasst:

- Ermittlung der Programmmodule des Programms und der Ausführungsbibliotheken, die von dem Programm verwendet werden und zusammen den Adressraum des Programms bilden,
- Berechnen einer Hash-Funktion der Programmmodule und der Ausführungsbibliotheken,
- Erfassen, ob eine unberechtigte Modifikation der Programmmodule und der Ausführungsbibliotheken aufgetreten ist, indem die jeweils berechnete(n) Hash-Funktionen mit früher erhaltenen Hash-Funktionen der Programmmodule und der Ausführungsbibliotheken, die in einem vor Modifikationen geschützten Speicherbereich der Computervorrichtung gespeichert sind, verglichen wird (werden).

2. Verfahren nach Anspruch 1, bei dem eine Fehlermeldung ausgegeben wird, falls eine unberechtigte Modifikation des Adressraums des Programms erfasst wird.

3. Verfahren nach Anspruch 1 oder 2, bei dem getrennte Hash-Funktionen der Programmmodule und der Ausführungsbibliotheken, die von dem Programm verwendet werden, berechnet werden.

4. Verfahren nach Anspruch 1 oder 2, bei dem eine kombinierte Hash-Funktion der Programmmodule und der Ausführungsbibliotheken, die von dem Programm verwendet werden, berechnet wird.

5. Verfahren nach Anspruch 1, bei dem eine kryptographisch starke Hash-Funktion mit einer Länge von wenigstens 100 Bits, vorzugsweise wenigstens 160 Bits, verwendet wird.

6. Verfahren nach Anspruch 3, das eine Programminstallationsprozedur umfasst, die die folgenden Schritte umfasst:

- Berechnen einer Hash-Funktion $h(M)$ der Programmmodule und
- Berechnen einer Hash-Funktion $h(L)$ der Ausführungsbibliotheken.

7. Verfahren nach Anspruch 6, bei dem die Installationsprozedur das Berechnen der Hash-Funktion $h(M)$ von Modulen, die von dem Autor des Programms bereitgestellt werden, und von externen Modulen, die durch die Betriebsumgebung bereitgestellt werden, umfasst.

8. Verfahren nach den Ansprüchen 6 oder 7, bei dem die Installationsprozedur wiederholt wird, falls eine Programmaktualisierung installiert wird.

9. Verfahren nach den Ansprüchen 6, 7 oder 8, bei dem $h(M)$ und $h(L)$ in einem vor Modifikationen geschützten Speicherbereich (**54**; **121**) der Computervorrichtung gespeichert werden.

10. Verfahren nach Anspruch 9, bei dem der Erfassungsschritt das Berechnen der Hash-Funktionen $h^*(M)$ der Programmmodule und $h^*(L)$ der Ausführungsbibliotheken beim Starten der Programmausführung und das Vergleichen von $h^*(M)$ und von $h^*(L)$ mit Hash-Funktionen $h(M)$ bzw. $h(L)$, die in dem vor Modifikationen geschützten Speicherbereich gespeichert sind, umfasst.

11. Verfahren nach Anspruch 3 bis 10, das die Verwendung einer asm-Funktion umfasst, um die Namen der Programmmodule und die Namen der Ausführungsbibliotheken des Adressraums des Programms zu erhalten.

12. Verfahren nach einem der Ansprüche 9 bis 11, bei dem der Modifikationsschutz des vor Modifikationen geschützten Speicherbereichs durch die Betriebsumgebung über Zugriffskontrollen und/oder Anwender-Accounts verwirklicht wird.

13. Verfahren nach einem der Ansprüche 9 bis 12, bei dem dem Programm selbst während der Installationsprozedur ein temporärer Zugriff auf den vor Modifikationen geschützten Speicherbereich gewährt wird.

14. Verfahren nach einem der Ansprüche 3 bis 13, bei dem der Erfassungsschritt das Identifizieren von fehlenden, hinzugefügten oder modifizierten Programmmodulen umfasst.

15. Verfahren nach einem der Ansprüche 1 bis 14, bei dem das geschützte Programm ein sicherheitssensitives Programm wie etwa ein Digitalsignatur-Programm, ein Finanztransaktions-Programm oder dergleichen ist.

16. Computervorrichtung, die eine Prozessoreinheit (51), wenigstens eine Speichervorrichtung (53) und Eingabe/Ausgabe-Einheiten (52) umfasst, die so programmiert sind, dass sie ein Verfahren nach einem der Ansprüche 1 bis 15 ausführen.

17. Computerprogramm, das einen Programmcode für die Ausführung eines Verfahrens nach einem der Ansprüche 1 bis 15 enthält.

18. Speichermedium, in dem ein Computerprogramm gespeichert ist, das einen Programmcode für die Ausführung eines Verfahrens nach einem der Ansprüche 1 bis 15 umfasst.

19. Verfahren nach einem der Ansprüche 6 oder 7, bei dem die Installationsprozedur wiederholt wird, wenn eine Programmaktualisierung installiert wird.

20. Verfahren für die Kommunikation zwischen wenigstens zwei Kommunikationsteilnehmern (A, B) einer Computervorrichtung, das die folgenden Schritte umfasst:

–

Ausführen der Integritätsschutzprozedur nach Anspruch 1 für die Programme der Kommunikationsteilnehmer (A, B),

– Starten einer Kommunikationssequenz zwischen den wenigstens zwei Kommunikationsteilnehmern,

– Erzeugen eines privaten Schlüssels und eines öffentlichen Schlüssels für eine digitale Signatur bei jedem der Kommunikationsteilnehmer für jede Kommunikationssequenz,

– Verfügbarmachen des öffentlichen Schlüssels eines Kommunikationsteilnehmers für den anderen Kommunikationsteilnehmer und

– Austauschen verschlüsselter Nachrichten zwischen den Kommunikationsteilnehmern mit angefügten digitalen Signaturen unter Verwendung der entsprechenden privaten und öffentlichen Schlüssel.

21. Verfahren nach Anspruch 20, das die Schritte des Schreibens des öffentlichen Schlüssels jedes Kommunikationsteilnehmers in eine im Voraus definierte feste Adresse in einem vor Modifikationen geschützten Speicherbereich umfasst.

22. Verfahren nach Anspruch 20 oder 21, das den Schritt des Ungültigmachens des öffentlichen Schlüssels jedes Kommunikationsteilnehmers bei Beendigung einer Kommunikationssequenz umfasst.

23. Verfahren nach Anspruch 21 oder Anspruch 22, das eine Installationsprozedur für jeden Kommunikationsteilnehmer umfasst, die die folgenden Schritte umfasst:

– Definieren eines vor Modifikationen geschützten Speicherbereichs,

– Festcodieren einer festen Adresse für den öffentlichen Schlüssel des Kommunikationsteilnehmers in dem vor Modifikationen geschützten Speicherbereich und

– Festcodieren der festen Adressen der öffentlichen Schlüssel aller anderen Kommunikationsteilnehmer, für die eine sichere Kommunikation freigegeben werden soll, in dem vor Modifikationen geschützten Speicherbereich.

24. Verfahren nach einem der Ansprüche 20 bis 23, bei dem die privaten und öffentlichen Schlüssel in Ausführungszeit erzeugt werden.

25. Verfahren nach einem der Ansprüche 20 bis 24, bei dem die Kommunikationsteilnehmer lokale Module eines Computersystems sind.

26. Computervorrichtung, die mehrere funktionale Module (A, B, C, D) umfasst, die miteinander unter Verwendung des Kommunikationsverfahrens nach einem der Ansprüche 20–25 kommunizieren, wobei die Computervorrichtung umfasst:

– eine Schlüsselerzeugungseinheit (212), die einen privaten Schlüssel und einen entsprechenden öffent-

lichen Schlüssel erzeugt,

- eine Signaturanfügungseinheit (**211**), die eine Nachricht mit angefügter digitaler Signatur unter Verwendung des privaten Schlüssels bereitstellt,
- einen vor Modifikationen geschützten Speicherbereich (**214**), der eine feste Adresse für die Speicherung des öffentlichen Schlüssels und weitere feste Adressen der öffentlichen Schlüssel der anderen Module, für die eine sichere Kommunikation freigegeben werden soll, besitzt, und
- eine Signaturverifikationseinheit (**210**), die von den anderen Modulen empfangene Nachrichten unter Verwendung der jeweiligen öffentlichen Schlüssel der sendenden Module verifiziert.

27. Computervorrichtung nach Anspruch 26, wobei die Computervorrichtung ein Personal Computer ist.

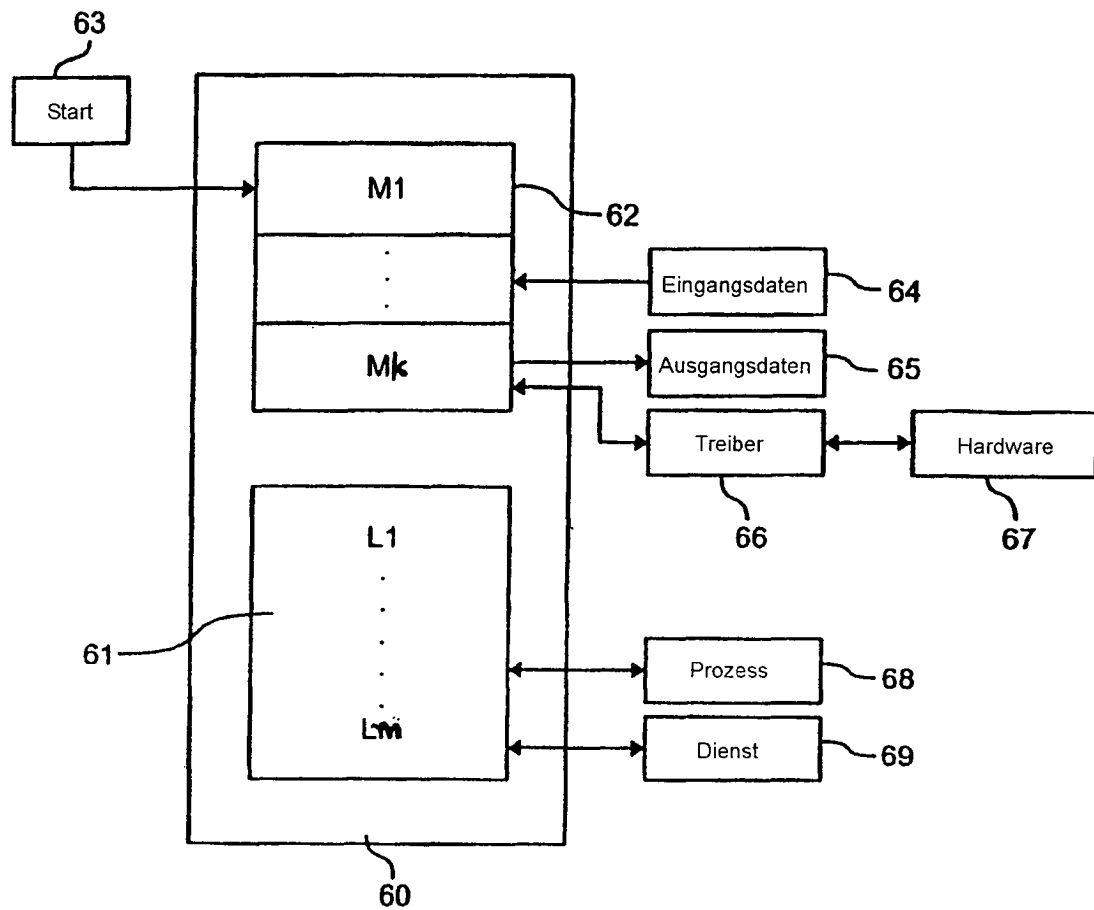
28. Computervorrichtung nach Anspruch 26, wobei die Computervorrichtung eine mobile Vorrichtung wie etwa ein Laptop, ein persönlicher digitaler Assistent oder ein Mobiltelefon ist.

29. Computerprogramm, das einen Programmcode enthält, der das Verfahren nach einem der Ansprüche 20 bis 25 ausführt, wenn er auf einem Computer ausgeführt wird.

30. Speichermedium, in dem ein Computerprogramm nach Anspruch 29 gespeichert ist.

Es folgen 11 Blatt Zeichnungen

Anhängende Zeichnungen

**Fig. 1**

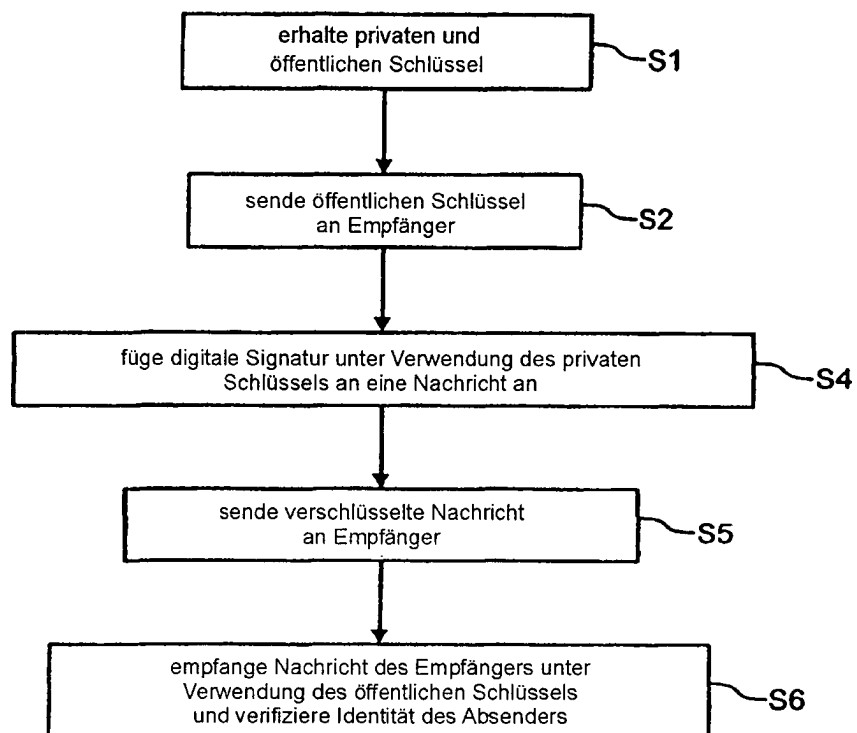


Fig. 2

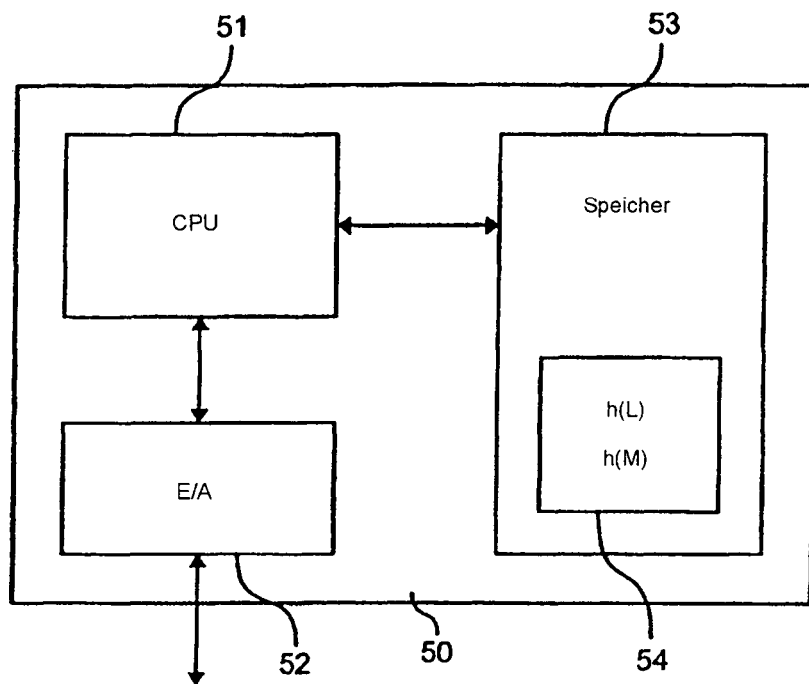


Fig. 3

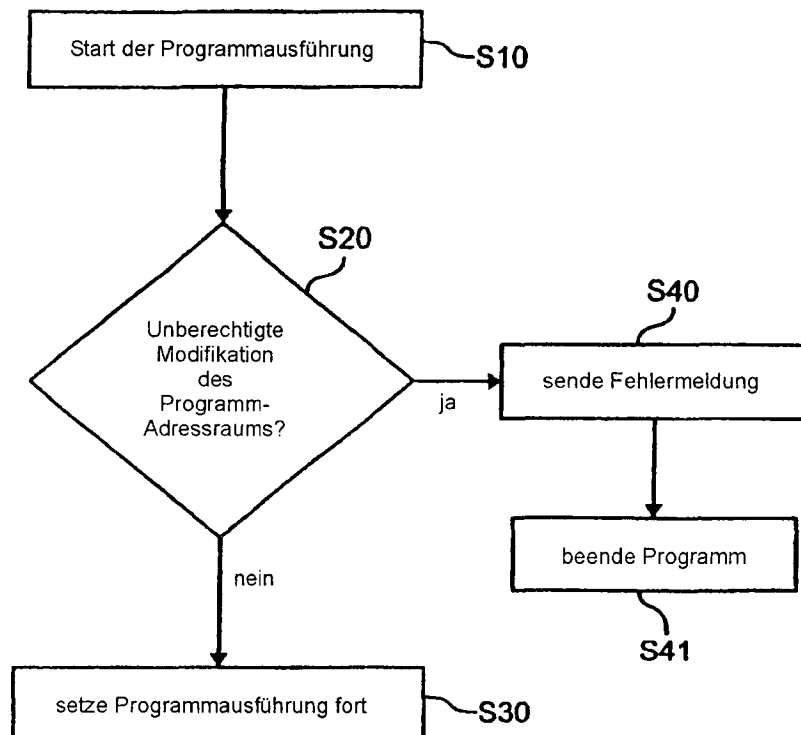


Fig. 4

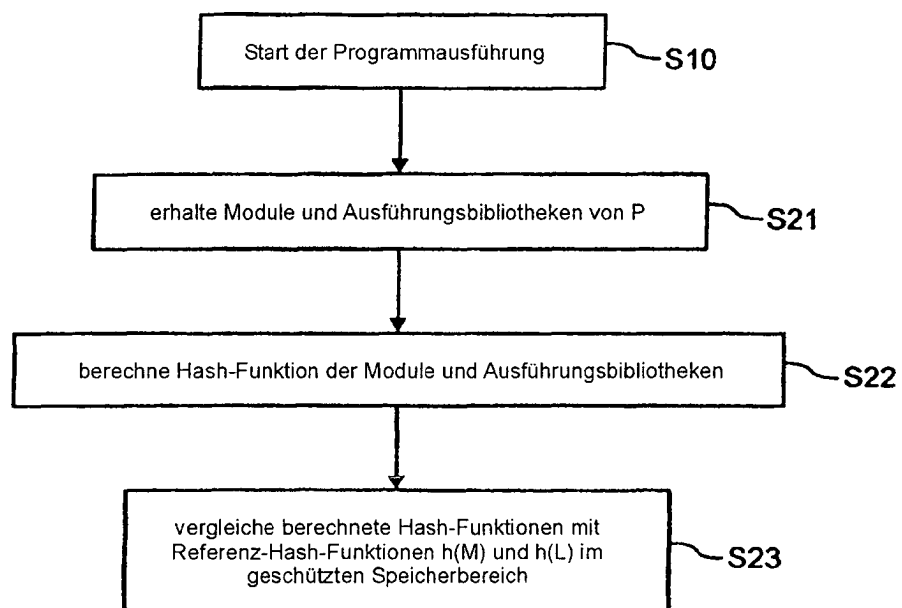


Fig. 5

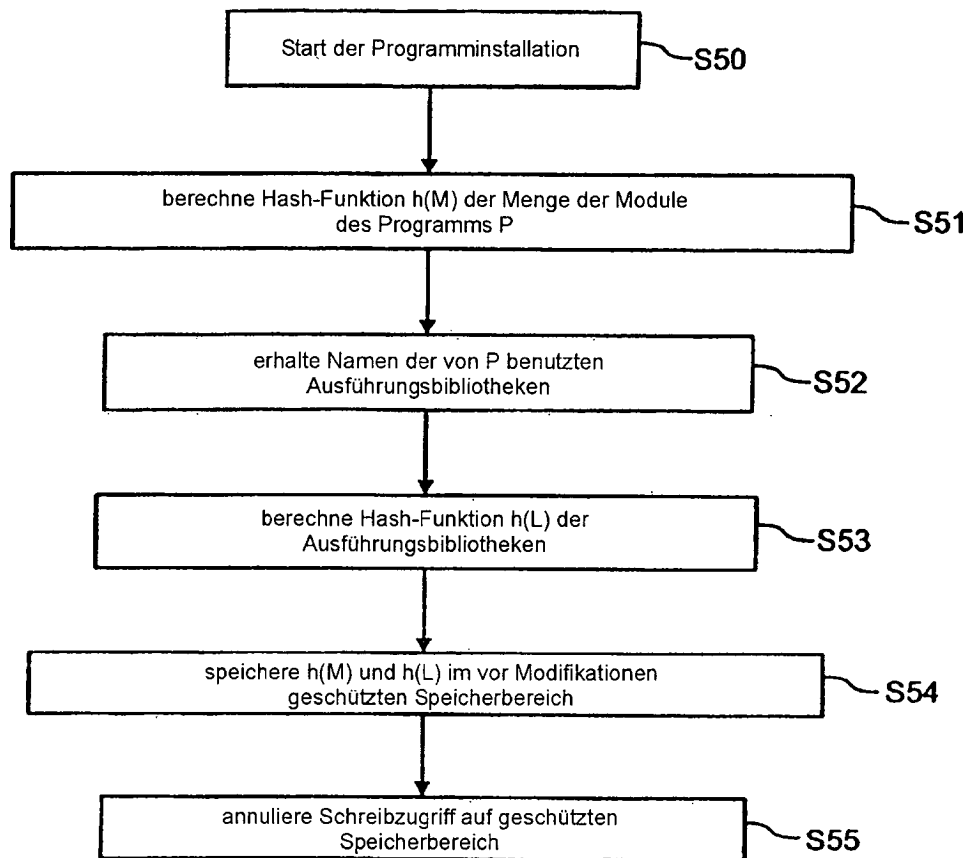


Fig. 6

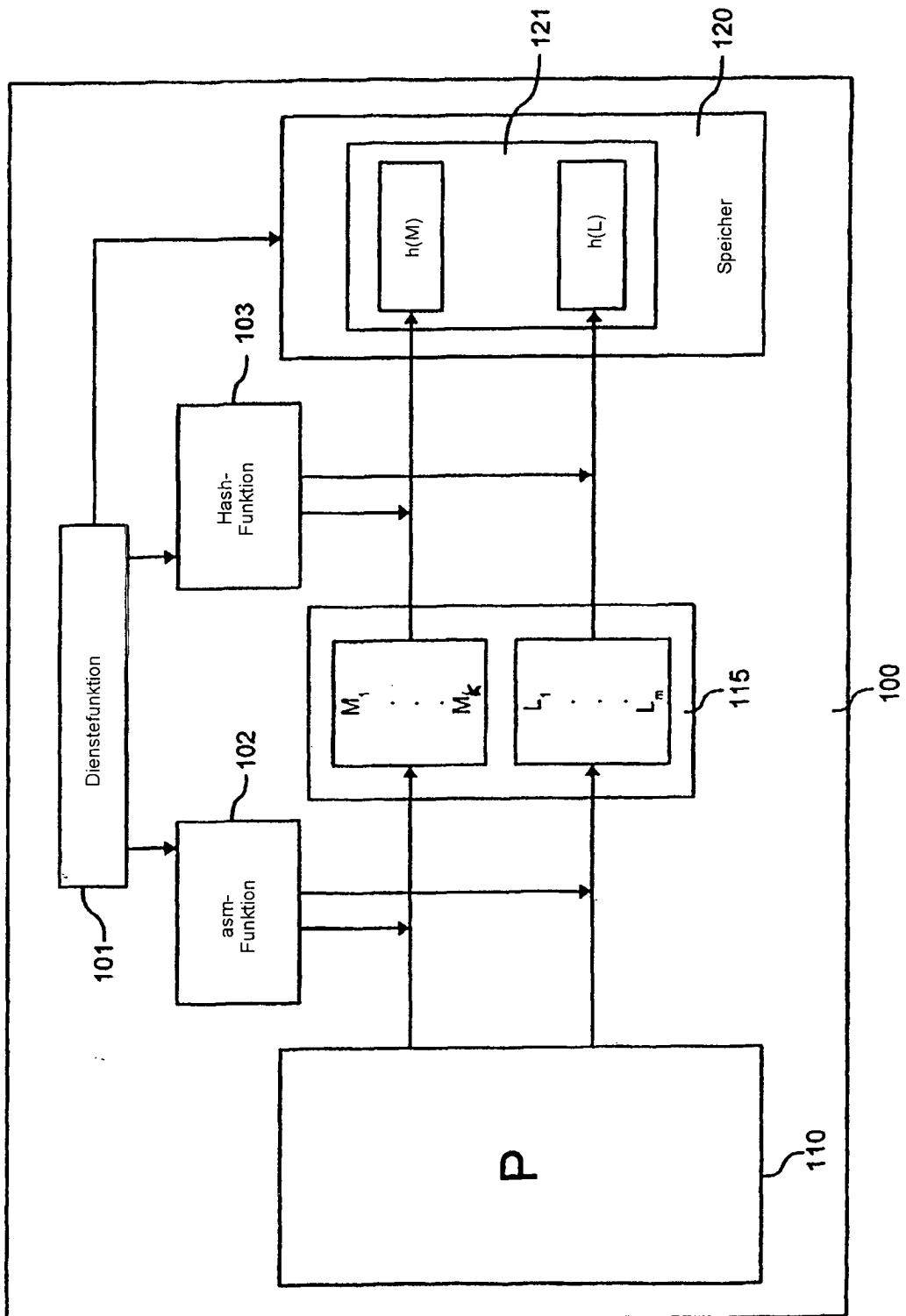


Fig. 7

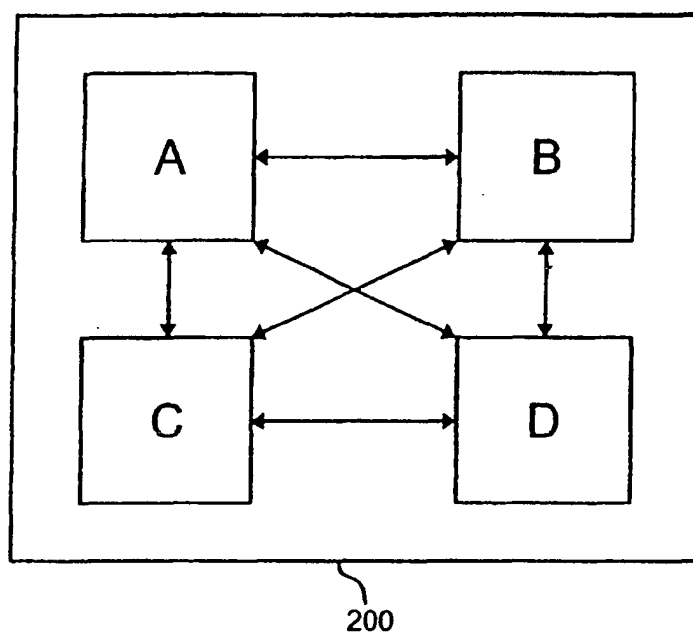


Fig. 8

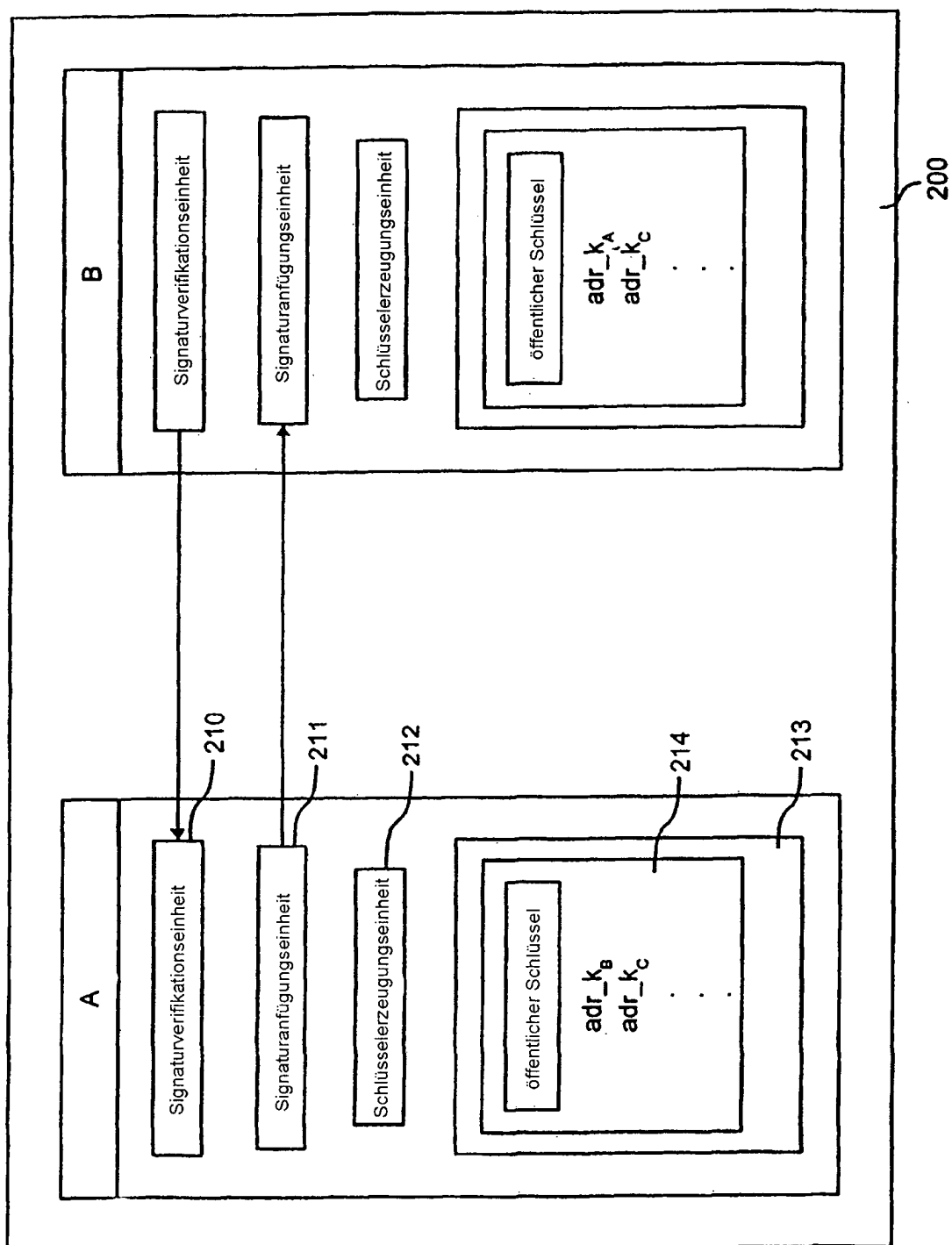


Fig. 9

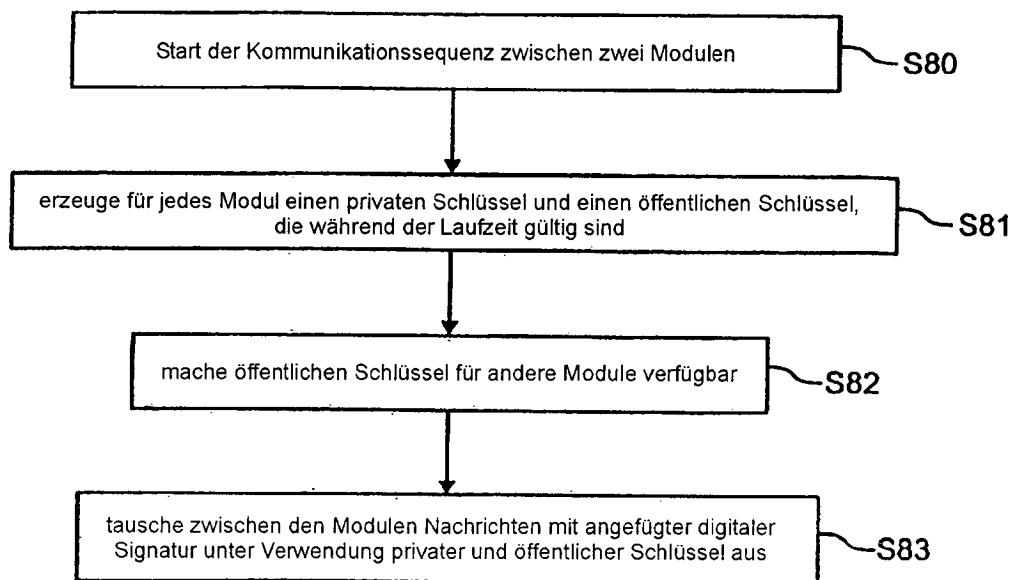


Fig. 10

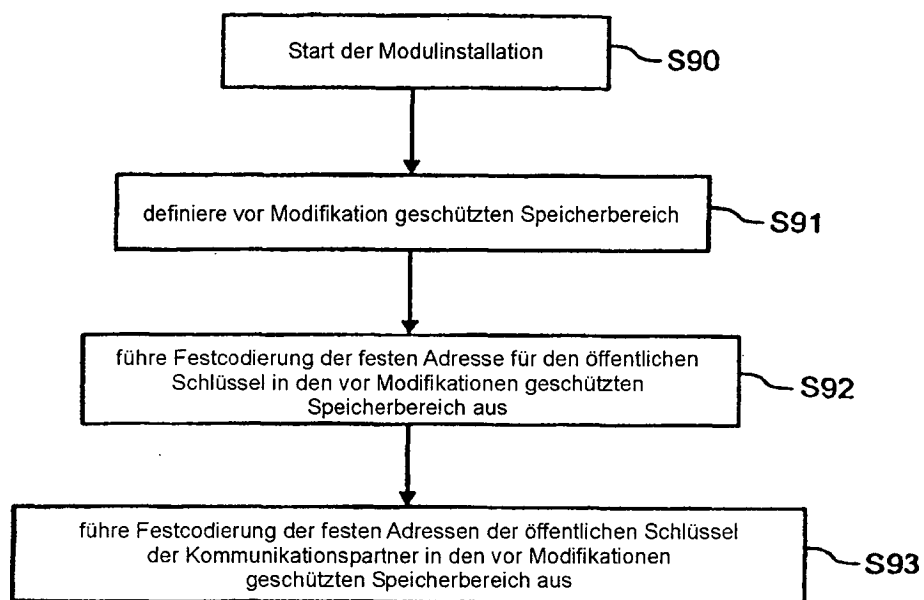


Fig. 11