(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 101 96 006 B4** 2008.07.24

(12) # Patentschrift

(21) Deutsches Aktenzeichen: **101 96 006.9**
(86) PCT-Aktenzeichen: **PCT/US01/08365**
(87) PCT-Veröffentlichungs-Nr.: **WO 2001/075563**
(86) PCT-Anmeldetag: **14.03.2001**
(87) PCT-Veröffentlichungstag: **11.10.2001**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **03.04.2003**
(45) Veröffentlichungstag der Patenterteilung: **24.07.2008**

(51) Int Cl.[8]: ***G06F 1/00*** (2006.01)

(30) Unionspriorität:
09/539,348 31.03.2000 US

(73) Patentinhaber:
Intel Corporation, Santa Clara, Calif., US

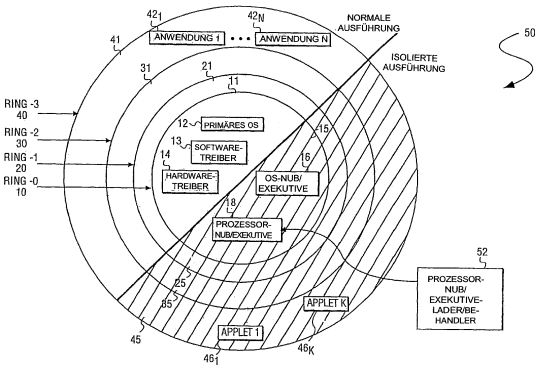
(74) Vertreter:
ZENZ Patent- und Rechtsanwälte, 45128 Essen

(72) Erfinder:
Ellison, Carl M., Portland, Oreg., US; Golliver, Roger A., Beaverton, Oreg., US; Herbert, Howard C., Phoenix, Ariz., US; Lin, Derrick C., San Mateo, Calif., US; Mckeen, Francis X., Portland, Oreg., US; Neiger, Gilbert, Portland, Oreg., US; Reneris, Ken, Wilbraham, Mass., US; Sutton, James A., Portland, Oreg., US; Thakkar, Shreekant S., Portland, Oreg., US; Mittal, Millind, Palo Alto, Calif., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
EP 06 00 112 A1
WO 98/44 402 A1

(54) Bezeichnung: **Erzeugen einer Schlüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung**

(57) Hauptanspruch: Einrichtung zum Erzeugen einer Schlüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung einer geschützten Plattform (100), wobei die geschützte Plattform einen Prozessor aufweist, der entweder in einem normalen Ausführungsmodus oder in einem isolierten Ausführungsmodus konfiguriert ist, wobei die die Einrichtung aufweist:
einen Schlüsselspeicher (155) zum Speichern eines für die geschützte Plattform einzigartigen Anfangsschlüssels (310); und
einen Chiffreschlüsselerzeuger (200) in der geschützten Plattform (100), der eine Hierarchie von Schlüsseln (327, 337, 347, 357) auf der Grundlage des Anfangsschlüssels (310) erzeugt, wobei der Chiffreschlüsselerzeuger (200) einen Schlüsselgenerator (210) für eine Hash-Operation eines ID (228; 320, 330, 340, 350) eines geladenen Softwarecodes mit dem Anfangsschlüssel (310) oder einem Schlüssel (225; 325, 335, 345) von ladendem Softwarecode zum Erzeugen eines Schlüssels (230; 325, 335, 345, 355) des geladenen Softwarecodes und einen Schlüsselauswähler (220) zum Auswählen eines kleineren symmetrischen Chiffreschlüssels (240; 327, 337, 347, 357) auf der Grundlage des Schlüssels (230; 325, 335,...



Beschreibung

[0001] Die vorliegende Erfindung bezieht sich auf eine Einrichtung und ein Verfahren zum Erzeugen einer Schlüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung einer geschützten Plattform sowie auf eine Plattform mit einer solchen Einrichtung.

[0002] Fortschritte bei Mikroprozessoren und in der Kommunikationstechnologie eröffneten viele Möglichkeiten für Anwendungen, die über die herkömmlichen Wege, Geschäfte auszuführen, hinausgehen. Elektronischer Handel (E-commerce) und Business-to-Business(B2B)-Transaktionen werden jetzt populär und mit ständig wachsenden Raten in globalen Märkten ausgeführt. Während moderne Mikroprozessorsysteme Benutzern bequeme und effiziente Methoden zur Ausführung des Geschäfts, zur Kommunikation und für Transaktionen zur Verfügung stellen, sind sie unglücklicherweise auch für skrupellose Angriffe anfällig. Beispiele dieser Angriffe umfassen Viren, ein Eindringen, Sicherheitsverletzungen und Verfälschungen, um nur wenige zu nennen. Die Computersicherheit wird folglich zunehmend wichtiger, um die Integrität der Computersysteme zu schützen und das Vertrauen der Benutzer zu erhöhen.

[0003] Von skrupellosen Angriffen verursachte Gefahren können eine Reihe von Formen annehmen. Ein eingreifender, von der Ferne aus gestarteter Angriff durch Hacker kann den normalen Betrieb eines mit Tausenden oder gar Millionen von Nutzern verbundenen Systems stören. Ein Virusprogramm kann Befehlscode und/oder Daten einer Einzelbenutzerplattform zerstören.

[0004] Vorhandene Techniken zum Schutz gegen Angriffe haben eine Reihe von Nachteilen. Antivirenprogramme können nur nach bekannten Viren suchen und diese erfassen. Sicherheitskoprozessoren oder Smartcards, die kryptographische oder andere Sicherheitstechniken verwenden, haben Grenzen bei der Geschwindigkeit, Leistung, Speicherkapazität und Flexibilität. Darüber hinaus schafft eine Neuentwicklung von Betriebssystemen Softwarekompabilitätsprobleme und erfordert gewaltige Investitionen bei den Entwicklungsbemühungen.

[0005] Aus der EP 0 600 112 A1 ist eine Datenverarbeitungsanlage (eine Plattform) mit virtueller Speicheradressierung und schlüsselgesteuertem Zugriff bekannt. Datenabschnitte eines Speichers (z. B. Seiten) sind durch zugeordnete Speicherschlüssel gegen unberechtigten Zugriff geschützt, indem ein Zugriff nur bei Übereinstimmung eines Zugriffsschlüssels mit dem jeweiligen Speicherschlüssel erlaubt wird.

[0006] Aus der WO 98/44402 A1 ist ein Verfahren zum Verschlüsseln von Daten, welche über das Internet übertragen werden, bekannt, bei dem diese Verschlüsselung ein unberechtigtes Kopieren der Daten auf dem empfangenden Computersystem vermeiden soll.

[0007] Aufgabe der Erfindung ist es, Schlüssel zum Verschlüsseln von Daten derart bereitzustellen, dass ein unbefugtes Manipulieren der Schlüssel vermieden wird.

[0008] Diese Aufgabe wird durch eine Einrichtung mit den Merkmalen des Anspruchs 1 bzw. ein Verfahren mit den Merkmalen des Anspruchs 9 bzw. eine Plattform mit den Merkmalen des Anspruchs 18 gelöst.

[0009] Vorteilhafte und/oder bevorzugte Weiterbildungen der Erfindung sind in den Unteransprüchen gekennzeichnet.

[0010] Die Aufgabe und die Merkmale der vorliegenden Erfindung werden aus der vorliegenden detaillierten Beschreibung der vorliegenden Erfindung naher ersichtlich, in welcher:

[0011] Fig. 1A ein Schema ist, das ein Betriebssystem gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht.

[0012] Fig. 1B ist ein Schema, das die Zugreifbarkeit verschiedener Elemente in dem Betriebssystem und dem Prozessor gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht.

[0013] Fig. 1C ist ein Schema, das ein Computersystem veranschaulicht, in welchem ein Ausführungsbeispiel der Erfindung praktiziert werden kann.

[0014] Fig. 2 ist ein Schema, das einen Chiffreschlüsselerzeuger gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht.

[0015] Fig. 3 ist ein Schema, das einen Prozeß zum Erzeugen einer Schüsselhierarchie gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht.

[0016] Fig. 4 ist ein Schema, das ein System zum Sichern eines Anfangsschlüssels, um das Back-up und die Wiederherstellung von Daten zu ermöglichen, gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht.

[0017] Fig. 5 ist ein Ablaufdiagramm, das den Prozeß zum Erzeugen einer Schüsselhierarchie gemäß einem Ausführungsbeispiel der Erfindung näher veranschaulicht.

[0018] In der folgenden Beschreibung wird eine bestimmte Terminologie verwendet, um bestimmte

Merkmale der vorliegenden Erfindung zu diskutieren. Beispielsweise umfaßt eine "Plattform" eine Hardwareausrüstung und Software, die verschiedene Funktionen an gespeicherten Informationen ausführt. Beispiele einer Plattform umfassen beispielsweise einen Computer (zum Beispiel Desktop, Laptop, Hand-held, Server, Workstation, etc.) eine Desktop-Büroausrüstung (z. B. Drucker, Scanner, Faxgerät, etc.), einen drahtlosen Telefonhandapparat, eine Fernseh-Set-Top-Box und dergleichen. Ein "Softwaremodul" enthält Befehlscode, der, wenn er ausgeführt wird, eine bestimmte Funktion durchführt. Ein "nub" ("kleiner Klumpen") ist eine Serie von Codeanweisungen, möglicherweise eine Untermenge des Codes aus einem Softwaremodul. Eine "link" (Verbindung) wird allgemein als eines oder mehrere informationsübertragene Medien definiert (zum Beispiel Draht, Lichtleiter, Kabel, Bus oder drahtlose Signalisiertechnologie).

[0019] Darüber hinaus ist der Begriff "Informationen" als eines oder mehrere Bits von Daten, Adressen und/oder Steuersignalen definiert. Eine "Hash-Funktion" ist eine mathematische oder andere Funktion, die eine nicht-umkehrbare Konversion von Informationen in eine Darstellung fest vorgegebener Länge ausführt. Normalerweise ist diese als "Hash-Wert" oder "Digest" ("Auswahl") bezeichnete Darstellung von wesentlich geringerer Größe als die ursprünglichen Informationen.

[0020] Die vorliegende Erfindung ist ein Verfahren und eine Einrichtung zum Erzeugen einer Schüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung einer geschützten Plattform. Um Geheimnisse an bestimmtem Befehlscode, der in der isolierten Ausführung betrieben wird, zu binden, wird eine Schüsselhierarchie mit einer Reihe von symmetrischen Schlüsseln für eine symmetrische Standardchiffre benutzt. Die geschützte Plattform enthält einen Prozessor, der entweder in einem normalen Ausführungsmodus oder einem isolierten Ausführungsmodus konfiguriert ist. Ein Schlüsselspeicher speichert einen Anfangsschlüssel, der für die Plattform einzigartig ist. Ein Chiffreschlüsselerzeuger, der in der geschützten Plattform angeordnet ist, erzeugt die Hierarchie von Schlüsseln auf der Grundlage des Anfangsschlüssels. Der Chiffreschlüsselerzeuger erzeugt eine Reihe von symmetrischen Chiffreschlüsseln, um die Geheimnisse des geladenen Softwarecodes zu schützen.

ARCHITEKTURÜBERBLICK

[0021] Ein Prinzip zum Bereitstellen einer Sicherheit in einem Computersystem oder einer Plattform ist das Konzept einer Architektur der isolierten Ausführung. Die Isolierte-Ausführung-Architektur umfaßt logische und physikalische Definitionen von Hardware- und Softwarekomponenten, die direkt oder indirekt mit einem Betriebssystems des Computersystems oder der Plattform interagieren. Ein Betriebssystem und der Prozessor können verschiedene Ebenen der Hierarchie aufweisen, die als Ringe bezeichnet werden und verschiedenen Betriebsmodi entsprechen. Ein Ring ist eine logische Abteilung der Hardware- und Softwarekomponenten, der so ausgebildet ist, daß er spezielle Aufgaben in dem Betriebssystem ausführt. Die Aufteilung basiert typischerweise auf dem Grad oder dem Niveau der Privilegierung, nämlich der Fähigkeit, Änderungen an der Plattform vorzunehmen. Beispielsweise ist ein Ring-0 der innerste Ring, der sich auf dem höchsten Niveau der Hierarchie befindet. Der Ring-0 umfaßt die kritischsten, privilegierten Komponenten. Darüber hinaus können Module in Ring-0 auch auf weniger privilegierte Daten zugreifen, aber nicht umgekehrt. Ring-3 ist der äußerste Ring, der das niedrigste Niveau der Hierarchie darstellt. Ring-3 umfaßt typischerweise die Benutzer- oder Anwendungsebene und weist die geringste Privilegierung auf. Ring-1 und Ring-2 stellen dazwischenliegende Ringe mit abnehmenden Pivilegierungsniveaus dar.

[0022] Fig. 1A ist ein Schema, das eine logische Betriebsarchitektur **50** gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht. Die logische Betriebsarchitektur **50** ist eine Abstraktion der Komponenten eines Betriebssystems und des Prozessors. Die logische Betriebsarchitektur **50** enthält einen Ring-0 **10**, Ring-1 **20**, Ring-2 **30**, Ring-3 **40** und einen Prozessor-Nub-Lader **52**. Der Prozessor-Nub-Lader **52** ist eine Instanz eines Prozessor-Exekutive(PE)-Behandlers. Der PE-Behandler wird verwendet, um eine Prozessor-Exekutive (PE) zu behandeln und/oder zu verwalten, wie es später erörtert wird. Die logische Betriebsarchitektur **50** weist zwei Betriebsmodi auf: einen normalen Ausführungsmodus und einen isolierten Ausführungsmodus. Jeder Ring der logischen Betriebsarchitektur **50** kann in beiden Modi betrieben werden. Der Prozessor-Nub-Lader **52** arbeitet nur in dem isolierten Ausführungsmodus.

[0023] Ring-0 **10** enthält zwei Abschnitte: einen Normale-Ausführung-Ring-0 **11** und einen Isolierte-Ausführung-Ring-0 **15**. Der Normale-Ausführung-Ring-0 **11** enthält Softwaremodule, die für das Betriebssystem kritisch sind, üblicherweise als Kernel bezeichnet. Diese Softwaremodule umfassen das primäre Betriebssystem (z. B. Kernel) **12**, Softwaretreiber **13** und Hardwaretreiber **14**. Der Isolierte-Ausführung-Ring-0 **15** enthält einen Betriebssystem(OS)-Nub **16** und einen Prozessor-Nub **18**. Der OS-Nub **16** und der Prozessor-Nub **18** sind Instanzen einer OS-Exekutive (OSE) bzw. einer Prozessor-Exekutive (PE). Die OSE und die PE sind Teil der Exekutiveentitäten, die in einer geschützten Umgebung arbeiten, die einem isolierten Bereich und dem isolierten Ausführungsmodus zugeordnet ist. Der

Prozessor-Nub-Lader **52** ist ein geschützter Bootstrap-Lader-Code, der in einem Chipsatz in dem System gehalten wird und für das Laden des Prozessor-Nubs **18** aus dem Prozessor oder dem Chipsatz in einen isolierten Bereich verantwortlich ist, wie später erörtert werden wird.

[0024] In ähnlicher Weise umfassen Ring-1 **20**, Ring-2 **30** und Ring-3 **40** einen Ring-1 **21**, Ring-2 **32**, Ring-3 **41** für normale Ausführung und einen Ring-1 **25**, Ring-2 **35** und Ring-3 **45** für isolierte Ausführung. Insbesondere enthält der Normale-Ausführung-Ring-3 N Anwendungen $\mathbf{42}_1$ bis $\mathbf{42}_N$, und der Isolierte-Ausführung-Ring-3 enthält K Applets $\mathbf{46}_1$ bis $\mathbf{46}_K$.

[0025] Ein Konzept der Architektur iolierter Ausführung ist die Schaffung eines isolierten Gebiets in dem Systemspeicher, das als isolierter Bereich bezeichnet wird und das sowohl durch den Prozessor als auch den Chipsatz in dem Computersystem gestützt ist. Der isolierte Bereich kann sich auch im Cache-Speicher befinden und durch eine Übersetzungsnachschlagepuffer(TLB)-Zugriffsüberprüfung geschützt sein. Darüber hinaus kann das isolierte Gebiet in mehrere isolierte Speicherbereiche unterteilt sein, wie erörtert wird. Ein Zugriff auf dieses isolierte Gebiet ist nur über einen Frontseitenbus (FSB) des Prozessors unter Verwendung spezieller Buszyklen (z. B. Speicherlese- und Schreibzyklen), die als isolierte Lese- und Schreibzyklen bezeichnet werden, gestattet. Die speziellen Buszyklen werden auch für ein Snooping verwendet. Die isolierten Lese- und Schreibzyklen werden durch den in einem isolierten Ausführungsmodus ausführenden Prozessor ausgegeben. Der isolierte Ausführungsmodus wird unter Verwendung eines privilegierten Befehls in den Prozessor, kombiniert mit dem Prozessor-Nub-Lader **52**, initialisiert. Der Prozessor-Nub-Lader **52** überprüft und lädt ein Ring-0-Nub-Softwaremodul (z. B. Prozessor-Nub **18**) in den isolierten Bereich. Der Prozessor-Nub **18** stellt hardwarebezogene Dienste für die isolierte Ausführung zur Verfügung.

[0026] Eine Aufgabe des Prozessor-Nubs **18** besteht darin, den Ring-0-OS-Nub **16** in den isolierten Bereich zu laden und diesen zu überprüfen und die Wurzel der Schüsselhierarchie, die einzigartig für eine Kombination der Plattform, des Prozessor-Nubs **18** und des Betriebssystem-Nubs **16** ist, zu erzeugen. Der Prozessor-Nub **18** schafft eine anfängliche Einricht-Verwaltung niedriger Ebene des isolierten Bereichs einschließlich einer Überprüfung, eines Ladens und eines Protokollierens (Logging) des Betriebssystem-Nubs **16** und die Verwaltung eines symmetrischen Schlüssels, der verwendet wird, um die Geheimnisse des Betriebssystem-Nubs zu schützen. Der Prozessor-Nub **18** kann darüber hinaus Anwendungsprogrammierschnittstellen(API)-Abstraktionen an Sicherheitsdienste niedriger Ebene, die durch andere Hardware zur Verfügung gestellt werden, zur Verfügung stellen.

[0027] Der Betriebssystem-Nub **16** stellt Verbindungen (Links) zu Diensten in dem primären OS **12** (z. B. den ungeschützten Segmenten des Betriebssystems) zur Verfügung, schafft eine Seitenverwaltung in dem isolierten Bereich und ist dafür verantwortlich, die Ring-3-Anwendungsmodule **45** einschließlich der Applets $\mathbf{46}_1$ bis $\mathbf{46}_K$ in die in dem isolierten Bereich zugeteilten geschützten Seiten zu laden. Der Betriebssystem-Nub **16** kann darüber hinaus Ring-0-Unterstützungsmodule laden.

[0028] Der Betriebssystem-Nub **16** kann auswählen, daß er ein Paging der Daten zwischen dem isolierten Bereich und dem gewöhnlichen (d. h. nicht isolierten) Speicher unterstützt. Wenn dies der Fall ist, dann ist der Betriebssystem-Nub **16** außerdem für eine Verschlüsselung und ein Hashing der Seiten des isolierten Bereichs verantwortlich, bevor die Seite in den gewöhnlichen Speicher geräumt wird, und für eine Überprüfung des Seiteninhalts bei der Wiederherstellung der Seite. Die Isolierter-Modus-Applets $\mathbf{46}_1$ bis $\mathbf{46}_K$ und ihre Daten sind verfälschungssicher und überwachungsresistent gegenüber sämtlichen Softwareangriffen von anderen Applets sowie aus Anwendungen (z. B. $\mathbf{46}_1$ bis $\mathbf{46}_N$) aus dem nicht-isolierten Raum, von dynamischen Verbindungsbibliotheken (DLLs), Treibern und selbst gegenüber dem primären Betriebssystem **12**. Nur der Prozessor-Nub **18** oder der Betriebssystem-Nub **16** können in die Ausführung des Applets eingreifen oder diese überwachen.

[0029] Fig. 1B ist ein Schema, das die Zugreifbarkeit der verschiedenen Elemente in dem Betriebssystem **10** und dem Prozessor gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht. Zu Veranschaulichungszwecken sind nur Elemente des Rings-0 **10** und des Rings-3 **40** gezeigt. Die verschiedenen Elemente in der logischen Betriebsarchitektur **50** greifen auf einen zugreifbaren physikalischen Speicher **60** in Übereinstimmung mit ihrer Ringhierarchie und dem Ausführungsmodus zu.

[0030] Der zugreifbare physikalische Speicher **60** enthält einen isolierten Bereich **70** und einen nicht-isolierten Bereich **80**. Der isolierte Bereich **70** enthält Applet-Seiten **72** und Nub-Seiten **74**. Der nicht-isolierte Bereich **80** enthält Anwendungsseiten **82** und Betriebssystemseiten **84**. Der isolierte Bereich **70** ist nur für Elemente des Betriebssystems und des Prozessors, die in dem isolierten Ausführungsmodus arbeiten, zugreifbar. Der nicht-isolierte Bereich **80** ist für sämtliche Elemente des Ring-0-Betriebssystems und des Prozessors zugreifbar.

[0031] Der Normale-Ausführung-Ring-0 **11** enthält das primäre OS **12**, die Softwaretreiber **13** und Hard-

waretreiber **14** und kann sowohl auf die OS-Seiten **84** als auch die Anwendungsseiten **82** zugreifen. Der Normale-Ausführung-Ring-3, der die Anwendungen $\mathbf{42}_1$ bis $\mathbf{42}_N$ enthält, kann nur auf die Anwendungsseiten **82** zugreifen. Jedoch können weder der Ring-0 **11** noch der Ring-3 **41** der normalen Ausführung auf den isolierten Bereich **70** zugreifen.

[0032] Der Isolierte-Ausführung-Ring-0 **15** einschließlich des OS-Nubs **16** und des Prozessor-Nubs **18** kann sowohl auf den isolierten Bereich **70** einschließlich der Applet-Seiten **72** und der Nub-Seiten **74** als auch auf den nicht-isolierten Bereich **80** einschließlich der Anwendungsseiten **82** und der OS-Seiten **84** zugreifen. Der Isolierte-Ausführung-Ring-3 **45**, einschließlich der Applets $\mathbf{46}_1$ bis $\mathbf{46}_K$ kann nur auf die Anwendungsseiten **82** und die Applet-Seiten **72** zugreifen. Die Applets $\mathbf{46}_1$ bis $\mathbf{46}_K$ halten sich in dem isolierten Bereich **70** auf.

[0033] Fig. 1C ist ein Schema, das eine Plattform **100** veranschaulicht, in welcher ein Ausführungsbeispiel der Erfindung ausgeführt werden kann. Die Plattform **100** enthält einen Prozessor **110**, einen Host-Bus **120**, einen Speicher-Controller-Hub (MCH) **130**, einen Systemspeicher **140**, einen Eingabe/Ausgabe-Controller-Hub (ICH) **150**, einen nicht-flüchtigen Speicher oder System-Flash-Speicher **160**, einen Zufallszahlengenerator **166**, eine Massenspeichereinrichtung **170**, Eingabe/Ausgabe-Geräte **175**, einen Token-Bus **180**, ein Mutterplatinen(MB)-Token **182**, einen Leser **184** und ein Token **186**. Der MCH **130** kann in einen Chipsatz integriert sein, der eine vielfache Funktionalität, wie beispielsweise den isolierten Ausführungsmodus, eine Host-Zu-Peripherie-Schnittstelle, eine Speichersteuereinrichtung integriert. In ähnlicher Weise kann der ICH **150** ebenfalls in einen Chipsatz gemeinsam mit dem oder getrennt von dem MCH **130** integriert sein, um I/O-Funktionen auszuführen. Aus Gründen der Klarheit sind nicht sämtliche Peripheriebusse gezeigt. Es ist beabsichtigt, daß die Plattform **100** darüber hinaus Peripheriebusse, wie beispielsweise einen Peripheriekomponentenverbindungsbus (PCI-Bus), einen beschleunigten Graphikport (AGP), einen Industriestandardarchitektur(ISA)-Bus und einen universellen seriellen Bus (USB), etc., enthält.

[0034] Der Prozessor **110** repräsentiert eine zentrale Verarbeitungseinheit einer beliebigen Architektur, wie beispielsweise einer Komplexer-Befehlssatz-Computer(CISC)-Architektur, Reduzierter-Befehlssatz-Computer(RISC)-Architektur, Sehr-Langes-Befehlswort(VLIW)-Architektur oder einer Hybrid-Architektur. Bei einer Ausführungsform ist der Prozessor **110** mit einem Prozessor einer Intel-Architektur (IA), wie beispielsweise einem Prozessor der Pentium™-Serie mit IA-32™ und IA-64™ kompatibel. Der Prozessor **110** schließt einen normalen Ausführungsmodus und eine Isolierte-Ausführung-Schaltung **115** ein. Der normale Ausführungsmodus **112** ist der Modus, in welchem der Prozessor **110** in einer Nicht-Geschützten-Umgebung oder einer normalen Umgebung ohne die von dem isolierten Ausführungsmodus zur Verfügung gestellten Sicherheitsmerkmale betrieben wird. Die Isolierte-Ausführung-Schaltung **115** stellt einen Mechanismus zur Verfügung, der es dem Prozessor **110** ermöglicht, in einem isolierten Ausführungsmodus zu arbeiten. Die Isolierte-Ausführung-Schaltung **115** schafft die Hardware- und Software-Unterstützung für den isolierten Ausführungsmodus. Diese Unterstützung umfaßt eine Konfiguration für eine isolierte Ausführung, das Definieren eines isolierten Bereichs, das Definieren (z. B. Dekodieren und Ausführung) isolierter Befehle, das Erzeugen Isolierter-Zugriff-Buszyklen und das Erzeugen Isolierter-Modus-Interrupts.

[0035] Bei einem Ausführungsbeispiel kann die Plattform **100** ein Einzelprozessorsystem, wie beispielsweise ein Desktop-Computer sein, welcher nur eine Hauptzentralverarbeitungseinheit, beispielsweise den Prozessor **110**, aufweist. Bei anderen Ausführungsbeispielen kann die Plattform **100** mehrere Prozessoren, zum Beispiel die Prozessoren **110**, **110a**, **110b**, etc. enthalten, wie es in Fig. 1C gezeigt ist. So kann die Plattform **100** ein Multi-Prozessor-Computersystem mit einer beliebigen Anzahl von Prozessoren sein. Beispielsweise kann die Multi-Prozessor-Plattform **100** als Teil einer Server- oder Workstation-Umgebung arbeiten. Die grundlegende Beschreibung und die Betriebsweise des Prozessors **110** werden detailliert nachfolgend erörtert. Für Fachleute ist es klar, daß diese grundlegende Beschreibung und Betriebsweise des Prozessors **110** auch für die anderen Prozessoren **110a** und **110b**, die in Fig. 1C gezeigt sind, sowie für eine beliebige Anzahl weiterer Prozessoren, die in der Multi-Prozessor-Plattform **100** in Übereinstimmung mit einem Ausführungsbeispiel der vorliegenden Erfindung benutzt werden können, gilt.

[0036] Der Prozessor **110** kann darüber hinaus mehrere logische Prozessoren aufweisen. Ein logischer Prozessor, der manchmal als Thread bezeichnet wird, ist eine Funktionseinheit in einem physikalischen Prozessor mit einem Architekturzustand und gemäß irgendeiner Aufteilungsvorgehensweise zugeteilten physikalischen Ressourcen. Im Kontext der vorliegenden Erfindung sollen die Begriffe "Thread" und "logischer Prozessor" dieselbe Bedeutung haben. Ein Multi-Threaded-Prozessor ist ein Prozessor mit mehreren Threads oder mehreren logischen Prozessoren. Eine Multi-Prozessor-System, (zum Beispiel das die Prozessoren **110a** und **110b** aufweisende System), kann mehrere mehrfädige (multi-threaded) Prozessoren aufweisen.

[0037] Der Host-Bus **120** stellt Schnittstellensignale zur Verfügung, um es dem Prozessor **110** oder den

Prozessoren **110**, **110a** und **110b** zu ermöglichen, mit anderen Prozessoren oder Einrichtungen, beispielsweise dem MCH **130**, zu kommunizieren. Zusätzlich zum normalen Modus stellt der Host-Bus **120** einen Bus-Modus des isolierten Zugriffs mit zugehörigen Schnittstellensignalen für Speicherlese- und -schreibzyklen zur Verfügung, wenn der Prozessor **110** im isolierten Ausführungsmodus konfiguriert ist. Der Busmodus des isolierten Zugriffs wird bei Speicherzugriffen angelegt, die initiiert werden, während sich der Prozessor **110** im isolierten Ausführungsmodus befindet. Der Isolierter-Zugriff-Busmodus wird außerdem bei Befehls-Vorabruf- und Cache-Rückschreib-Zyklen angelegt, wenn die Adresse innerhalb des Adreßbereichs des isolierten Bereichs liegt und der Prozessor **110** in dem isolierten Ausführungsmodus initialisiert ist. Der Prozessor **110** antwortet auf Snoop-Zyklen zu einer cache-gespeicherten Adresse in dem Adreßbereich des isolierten Bereichs, wenn der Isolierte-Zugriff-Busmodus angelegt und der Prozessor **110** in den isolierten Ausführungsmodus initialisiert ist.

[0038] Der MCH **130** stellt eine Steuerung und Konfiguration des Speichers und der Eingabe/Ausgabe-Einrichtungen, wie beispielweise des Systemspeichers **140** und des ICH **150**, zur Verfügung. Der MCH **130** stellt Schnittstellenschaltungen zur Verfügung, um das Anlegen isolierten Zugriffs bei Speicherreferenzbuszyklen, einschließlich isolierter Speicherlese- und -Schreibzyklen, zu erkennen und diese zu bedienen. Zusätzlich weist der MCH **130** Speicherbereichsregister (z. B. Basis- und Längenregister) auf, um den isolierten Bereich in dem Systemspeicher **140** darzustellen. Sobald er konfiguriert ist, bricht der MCH **130** jegliche Zugriffe auf einen isolierten Bereich ab, bei denen kein Isolierter-Zugriff-Busmodus angelegt ist.

[0039] Der Systemspeicher **140** speichert Systemcode und Daten. Der Systemspeicher **140** ist typischerweise mit dynamischem Speicher mit wahlfreiem Zugriff (DRAN) oder statischem Speicher mit wahlfreiem Zugriff (SRAM) implementiert. Der Systemspeicher **140** enthält den zugreifbaren physikalischen Speicher **60**, der in Fig. 1B gezeigt ist. Der zugreifbare physikalische Speicher enthält ein geladenes Betriebssystem **142**, den isolierten Bereich **70** (Fig. 1B) und einen isolierten Steuer- und Statusraum **148**. Das geladene Betriebssystem **142** ist der Teil des Betriebssystems, der in den Systemspeicher **142** geladen wird. Das geladene OS **142** wird typischerweise aus einer Massenspeichereinrichtung über irgendeinen Anfangsladecode in einem Anfangsladespeicher, wie beispielsweise einem Boot-Nur-Lese-Speicher (ROM) geladen. Der isolierte Bereich **70** (Fig. 1B) ist derjenige Speicherbereich, der von dem Prozessor **110** definiert wird, wenn er in dem isolierten Ausführungsmodus arbeitet. Der Zugriff auf den isolierten Bereich ist eingeschränkt und wird von dem Prozessor **110** und/oder dem MCH **130** oder einem anderen Chipsatz durchgesetzt, der die Isolierter-Bereich-Funktionalitäten integriert. Der isolierte Steuer- und Statusraum **148** ist ein dem Eingabe/Ausgabe(I/O) ähnlicher unabhängiger Adreßraum, der in dem Prozessor **110** und/oder dem MCH **130** definiert wird. Der isolierte Steuer- und Statusraum **148** enthält hauptsächlich die Steuer- und Statusregister der isolierten Ausführung. Der isolierte Steuer- und Statusraum **148** überlappt keinen vorhandenen Adreßraum, und es wird auf ihn unter Verwendung der isolierten Buszyklen zugegriffen. Der Systemspeicher **140** kann darüber hinaus weitere Programme und Daten enthalten, welche nicht gezeigt sind.

[0040] Der ICH **150** repräsentiert einen bekannten Einzelpunkt in dem System, der die Funktionalität der isolierten Ausführung aufweist. Aus Gründen der Klarheit ist nur ein ICH **150** gezeigt. Die Plattform **100** kann viele ICHs enthalten, die ähnlich dem ICH **150** sind. Wenn es mehrere ICHs gibt, wird ein bestimmter ICH ausgewählt, um die Konfiguration und den Status des isolierten Bereichs zu kontrollieren. Bei einem Ausführungsbeispiel wird diese Auswahl durch ein externes Strapping-Pin ausgeführt. Wie es Fachleuten bekannt ist, können auch andere Verfahren der Auswahl verwendet werden, einschließlich der Verwendung programmierbarer Konfigurationsregister. Der ICH **150** weist eine Reihe von Funktionalitäten auf, die so ausgebildet sind, daß sie zusätzlich zu den herkömmlichen I/O-Funktionen den isolierten Ausführungsmodus unterstützen. Insbesondere enthält der ICH **150** eine Isolierter-Buszyklus-Schnittstelle **152**, den Prozessor-Nub-Lader **52** (der in Fig. 1A gezeigt ist), einen Digest-Speicher **154**, einen Speicher **155** für einen kryptographischen Schlüssel, eine kryptographische Hash-Einheit **157**, einen Isolierte-Ausführung-Logikverarbeitungs-Manager **156** und eine Token-Bus-Schnittstelle **159**.

[0041] Die Isolierter-Buszyklus-Schnittstelle **152** enthält Schaltungen zum Bilden einer Schnittstelle zu den Isolierter-Buszyklus-Signalen, um isolierte Buszyklen, wie beispielsweise die isolierten Lese- und Schreibbuszyklen zu erkennen und zu bedienen.

[0042] Der Prozessor-Nub-Lader **52**, wie er in Fig. 1A gezeigt ist, umfaßt einen Prozessor-Nub-Lader-Code und einen Digest-Wert (z. B. Hash-Wert). Der Prozessor-Nub-Lader **52** wird durch Ausführung eines geeigneten Isoliert-Befehls (zum Beispiel Iso-Init) aufgerufen und in den isolierten Bereich **70** übertragen. Aus dem isolierten Bereich kopiert der Prozessor-Nub-Lader **52** den Prozessor-Nub **18** aus dem System-Flash-Speicher (z. B. den Prozessor-Nub-Code **18** in dem nicht-flüchtigen Speicher **160**) in den isolierten Bereich **70**, überprüft und protokolliert seine Integrität und verwaltet einen symmetrischen Schlüssel, der verwendet wird, um die Ge-

heimnisse des Prozessor-Nubs zu schützen. Bei einem Ausführungsbeispiel ist der Prozessor-Nub-Lader **52** im Nur-Lese-Speicher (ROM) implementiert. Aus Sicherheitsgründen ist der Prozessor-Nub-Lader **52** unveränderbar, verfälschungssicher und nicht-austauschbar.

[0043] Der Digest-Speicher **154** speichert den Digest-Wert (z. B. Hash-Wert) der geladenen Softwaremodule. Insbesondere erwarten wir, daß der Digest-Speicher den Hash (oder den Zustand des zum Berechnen des Hash benötigten Werts) eines Prüfprotokolls (audit log) speichert. Das Prüfprotokoll enthält Prüfprotokolleinträge. Jeder Prüfprotokolleintrag enthält zumindest den kryptographischen Hash des geladenen Codes, wie beispielweise des Prozessor-Nubs **18**, des Betriebssystem(OS)-Nubs **16** und irgendwelcher anderen kritischen Module (z. B. Ring-0-Module), die in den isolierten Bereich geladen werden. Insbesondere ist das Prüfprotokoll eine Auflistung von Daten, die darstellen, welche Informationen erfolgreich in den Systemspeicher **140** nach dem Einschalten der Plattform **100** geladen worden sind. Beispielsweise können die repräsentativen Daten Hash-Werte jedes in dem Systemspeicher **140** während der aktuellen Einschaltperiode der Plattform **100** geladenen Softwaremoduls sein. So kann das Prüfprotokoll als Fingerabdruck dienen, der die in die Plattform geladenen Informationen identifiziert (z. B. den Ring-0-Code, der die Konfiguration und Operation der isolierten Ausführung steuert), und wird verwendet, um den Zustand der aktuellen isolierten Ausführung zu attestieren oder nachzuweisen.

[0044] Der Speicher **155** des kryptographischen Schlüssels hält einen symmetrischen Verschlüsselungs/Entschlüsselung-Schlüssel, beispielweise einen Anfangsschlüssel (der detaillierter später erörtert werden soll). Der Anfangsschlüssel ist für die Plattform **100** einzigartig. Bei einem Ausführungsbeispiel wird der Anfangsschlüssel durch einen externen Hardware-Zufallszahlgenerator erzeugt und zum Zeitpunkt der Herstellung in die Schmelzsicherungen des Eingabe/Ausgabe-Controller-Hubs (ICH) **150** programmiert. Bei einem anderen Ausführungsbeispiel wird der Anfangsschlüssel durch die Plattform selbst erzeugt, wenn die Plattform **100** erstmalig eingeschaltet wird, wobei der interne Hardwarezufallszahlengenerator **166** der Plattform verwendet wird. In beiden Fällen wird der Anfangsschlüssel in dem geschützten nicht-flüchtigen Speicher der Speichereinheit **155** des kryptographischen Schlüssels gespeichert. Jedoch ist es klar, daß der Anfangsschlüssel in geschützten nicht-flüchtigen Speichern in anderen Bereichen der Plattform gespeichert werden könnte.

[0045] Die kryptographische Hash-Einheit **157** enthält eine Logik, die eine Einweg-Hash-Funktion an eingegebenen Informationen durchführt. Der Begriff "Einweg-" zeigt an, daß es nicht ohne weiteres eine inverse Funktion gibt, um irgendeinen wahrnehmbaren Abschnitt der ursprünglichen Informationen aus dem Hash-Wert fester Länge wieder herzustellen. Beispiele einer Hash-Funktion umfassen die von der RSA Data Security aus Redwood City, Kalifornien, zur Verfügung gestellten MD5 oder einen sicheren Hash-Algorithmus (SHA-1), wie er in einer Veröffentlichung des Secure Hash Standard FIPS **180-1** mit dem Titel "Federal Information Processing Standards Publication" (17. April 1995) spezifiziert ist. Die kryptographische Hash-Einheit **157** kann verwendet werden, um Hashing-Funktionen auszuführen, um die Prozessor-Nub-Schlüssel, die OS-Nub-Schlüssel und die Applet-Schlüssel zu erzeugen. Diese Schlüssel werden detaillierter später erörtert.

[0046] Der Isolierte-Ausführung-Logikverarbeitungsmanager **156** verwaltet die Operation der logischen Prozessoren, die im isolierten Ausführungsmodus arbeiten. Bei einem Ausführungsbeispiel enthält der Isolierte-Ausführung-Logikverarbeitungsmananger **156** ein Logischer-Prozessor-Zählregister, das die Anzahl der logischen Prozessoren verfolgt, die an dem isolierten Ausführungsmodus teilnehmen. Die Token-Bus-Schnittstelle **159** bildet eine Schnittstelle zu dem Token-Bus **180**.

[0047] Der nicht-flüchtige Speicher **160** speichert nicht-flüchtige Informationen. Typischerweise ist der nicht-flüchtige Speicher **160** in Flash-Speicher implementiert. Der nichtflüchtige Speicher **160** enthält den Prozessor-Nub **18**.

[0048] Der Prozessor-Nub **18** stellt das anfängliche Einricht-Management auf niedrige Ebene für die isolierten Bereiche (in dem Systemspeicher **140**), einschließlich der Überprüfung, des Ladens und der Protokollierung (logging) des Betriebssystem-Nubs **16**, und die Verwaltung des symmetrischen Schlüssels, der zum Schützen der Geheimnisse des Betriebssystem-Nubs verwendet wird, zur Verfügung. Der Prozessor-Nub **18** kann darüber hinaus Anwendungsprogrammierschnittstellen(API)-Abstraktionen an Low-Level-Sicherheitsdienste zur Verfügung stellen, die von weiterer Hardware bereitgestellt werden. Der Prozessor-Nub **18** kann außerdem durch den ursprünglichen Ausrüstungshersteller (OEM) oder Betriebssystemanbieter (OSV) über eine Boot-Diskette verteilt werden.

[0049] Die Massenspeichereinrichtung **170** speichert Archivinformationen, wie beispielsweise Code (z. B. Prozessor-Nub **18**), Programme, Dateien, Daten, Anwendungen (z. B. Anwendungen $\mathbf{42_1}$ bis $\mathbf{42_N}$), Applets (z. B. Applets $\mathbf{46_1}$ bis $\mathbf{46_k}$) und Betriebssysteme. Die Massenspeichereinrichtung **170** kann eine Compact-Disk(CD)-ROM **172**, Disketten **174** und ein Festplattenlaufwerk **176** sowie beliebige weitere magnetische oder optische Speichereinrichtungen umfassen. Die Massenspeichereinrichtung **170** stellt ei-

nen Mechanismus zum Lesen eines maschinenlesbaren Mediums zur Verfügung.

[0050] Die I/O-Geräte **175** können beliebige I/O-Einrichtung zum Durchführen von I/O-Funktionen einschließen. Beispiele der I/O-Geräte **175** sind Controller für Eingabeeinrichtungen (z. B. Tastatur, Maus, Trackball, Zeigergerät), Medienkarten (z. B. Audio-, Video- und Grafikkarten), Netzwerkkarten und irgendwelche anderen Peripheriesteuereinrichtungen.

[0051] Der Token-Bus **180** stellt eine Schnittstelle zwischen dem ICH **150** und verschiedenen Tokens in dem System zur Verfügung. Ein Token ist eine Einrichtung, die spezielle Eingabe/Ausgabe-Funktionen mit Sicherheitsfunktionalität zur Verfügung stellt. Ein Token weist Charakteristika auf, die ähnlich einer Smart-Card sind, einschließlich wenigstens eines Schlüsselpaars aus öffentlichem/privatem Schlüssel für reservierte Zwecke und der Fähigkeit zum Unterzeichnen von Daten mit dem privaten Schlüssel. Beispiele des mit dem Token-Bus **180** verbundenen Tokens umfassen ein Mutterplatinen-Token **182**, einen Token-Leser **184** und weitere transportable Tokens **186** (z. B. Smart-Cards). Die Token-Bus-Schnittstelle **159** in dem ICH **150** verbindet sich über den Token-Bus **180** zu dem ICH **150** und sichert, daß dann, wenn sie aufgefordert ist, den Zustand der isolierten Ausführung nachzuweisen, der zugehörige Token (z. B. der Mutterplatinen-Token **182**, der Token **186**) nur gültig isolierte Digest-Informationen unterzeichnet. Aus Gründen der Sicherheit sollte der Token mit dem Digest-Speicher verbunden sein.

[0052] Wenn sie in Software implementiert sind, sind die Elemente der vorliegenden Erfindung Codesegmente zum Ausführen der erforderlichen Aufgaben. Die Programm- oder Codesegmente können in einem maschinenlesbaren Medium, wie beispielsweise einem prozessor-lesbaren Medium, gespeichert sein oder durch ein in einer Trägerwelle enthaltenes Computerdatensignal oder ein durch einen Träger moduliertes Signal über ein Übertragungsmedium übertragen werden. Das "prozessor-lesbare Medium" schließt irgendein Medium ein, das Informationen speichern oder übertragen kann. Beispiele prozessor-lesbarer Medien sind elektronische Schaltungen, Halbleiterspeicherbauelemente, ein ROM, ein Flash-Speicher, ein löschbarer und programmierbarer ROM (EPROM), eine Diskette, eine CD-ROM, eine optische Platte, eine Festplatte, ein Lichtleitermedium, eine Hochfrequenz(HF)-Verbindung, etc. Das Computerdatensignal kann irgendein Signal einschließen, das über ein Übertragungsmedium weitergeleitet werden kann, wie beispielsweise über elektronische Netzwerkkanäle, Lichtleiter, Luft, elektromagnetische, HF-Verbindungen, etc. Die Codesegmente können heruntergeladen werden über Computernetzwerke, wie beispielsweise das Internet, ein Intranet, etc.

ERZEUGUNG EINER SCHLÜSSELHIERARCHIE ZUR VERWENDUNG IN EINER ISOLIERTEN AUSFÜHRUNGSUMGEBUNG

[0053] Die vorliegende Erfindung ist ein Verfahren, eine Einrichtung und ein System zum Erzeugen einer Schlüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung einer geschützten Plattform. Um Geheimnisse an einen bestimmten Code zu binden, der in einer isolierten Ausführung betrieben wird, wird eine Schlüsselhierarchie mit einer Reihe von symmetrischen Schlüsseln für eine symmetrische Standard-Chiffre benutzt.

[0054] Fig. 2 ist ein Schema, das einen Chiffreschlüsselerzeuger **200** gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht. Der Chiffreschlüsselerzeuger enthält einen Schlüsselgenerator **210** und einen Schlüsselauswähler **220**. Softwarecode, der gegenwärtig in der isolierten Ausführungsumgebung aktiv ist (d. h. bereits in den isolierten Speicherbereich geladen ist), weist einen ihm zugeordneten Ladungssoftwarecodeschlüssel **225** auf. Der Ladungssoftwarecode, der gegenwärtig aktiv ist, wird verwendet, um den nachfolgenden Softwarecode zu laden. Softwarecode, der nachfolgend in die isolierte Ausführungsumgebung geladen wird, weist einen ihm zugeordneten ID-Wert auf, der als ID des geladenen Softwarecodes **228** bezeichnet ist. Der ID **228** kann der Hash-Wert des geladenen Softwarecodes sein. Jedoch könnte er auch ein anerkannter ID des geladenen Softwarecodes sein.

[0055] Bei dem Ausführungsbeispiel, bei dem der ID **228** ein anerkannter ID des geladenen Softwarecodes ist, kann der ID 228 an den aktiven ladenden Softwarecode mit Hilfe eines Zertifikats übermittelt werden, daß mit Hilfe eines Zertifikatüberprüfungswurzelschlüssels überprüft wird, der innerhalb des aktiven ladenden Softwarecodes gehalten wird. Bei einem Ausführungsbeispiel besteht das Zertifikat aus dem Zertifikatinhalt und einer digitalen Signatur über den Zertifikatinhalt. Der Zertifikatinhalt kann zumindest aus dem Hash des geladenen Softwarecodes und dem ID-Wert **228** bestehen. Ein Beispiel eines Zertifikats, das benutzt werden könnte, ist ein SPKI-Authorisierungszertifikat (siehe IETF RFC 2693, Internet Engineering Task Force Request for Comments 2693, "SPK1 Certificate Theory", Ellison et al., September 1999). Jedoch ist es Fachleuten klar, daß eine große Vielzahl verschiedener Zertifikatformate benutzt werden könnte.

[0056] Der Schlüssel **225** des ladenden Softwarecodes und der ID des geladenen Softwarecodes **228** werden als Eingaben dem Schlüsselgenerator **210** zur Verfügung gestellt. Der Schlüsselgenerator **210** erzeugt einen Schlüssel **230** für den geladenen Softwarecode. Der Schlüsselgenerator **210** erzeugt den Schlüssel **230** für den geladenen Softwarecode unter

Benutzung einer Funktion, die die Eigenschaften eines kryptographischen Hash aufweist. So kann bei einem Ausführungsbeispiel eine Hashing-Funktion von dem Schlüsselgenerator **210** benutzt werden. Der Schlüsselgenerator erzeugt einen Hash-Wert des Schlüssels **225** des ladenden Softwarecodes und des IDs **228** des geladenen Softwarecodes, um den Schlüssel **230** des geladenen Softwarecodes zu erzeugen. Beispielsweise könnte ein HMAC, der auf SHA-1 basiert, benutzt werden (siehe IETF RFC 2104, Internet Engineering Task Force Request for Comments 2104, "HMAC: Keyed-Hashing for Message Authentication", Krawczyk et al., Februar 1997). Es ist jedoch klar, daß andere allgemein akzeptable kryptographische Hash-Funktionen ebensogut verwendet werden können. Darüber hinaus kann bei einem Ausführungsbeispiel die kryptographische Hash-Einheit **157** (Fig. 1C) verwendet werden, um die Hashing-Funktion auszuführen. Außerdem ist es klar, daß eine breite Vielfalt anderer Arten von Einwegfunktionen verwendet werden kann, um den Schlüssel **230** des geladenen Softwarecodes zu erzeugen.

[0057] Im allgemeinen ist der Schlüssel **230** des geladenen Softwarecodes von der Größe eines Hash-Funktionsausgabewerts, aber er könnte von beliebiger Größe sein, vorausgesetzt, er weist zumindest ausreichende Bits auf, um kryptographisch sicher zu sein. Jedoch könnte der Schlüssel **230** für den geladenen Softwarecode nicht klein genug sein zur Verwendung in einem gewünschten symmetrischen Chiffre, wie beispielsweise Triple-DES (Data Encryption Standard) oder AES (der neue Advanced Encryption Standard, der von dem NIST (National Institute of Standards and Technology) ausgewählt worden ist). Folglich wendet der Schlüsselauswähler **220** eine Schlüsselauswahlfunktion an, um einen Schlüssel geeigneter Größe (z. B. 112, 168, 128 oder 256 Bits) aus dem größeren Schlüssel **230** des geladenen Softwarecodes auszuwählen, um einen symmetrischen Chiffreschlüssel **240** für den geladenen Softwarecode zu erzeugen.

[0058] Der Schlüsselgenerator **210** führt eine Erzeugungsfunktion aus, die mit G() bezeichnet wird, um einen zweiten Schlüssel (mit K2 bezeichnet) aus einem ersten Schlüssel (mit K1 bezeichnet) und einem ID zu erzeugen. Die Gleichung nimmt die Form K2 = G(ID, K1) an. Bei einem Ausführungsbeispiel ist der zweite Schlüssel K2, der erzeugt worden ist, der Schlüssel **230** des geladenen Softwarecodes. So erzeugt die Gleichung K2 = G(ID, K1) den Schlüssel **230** des geladenen Softwarecodes unter Verwendung der G()-Funktion, wobei der ID der ID **228** des geladenen Softwarecodes und K1 der Schlüssel **225** des ladenden Softwarecodes ist. Wie zuvor erörtert, kann die Erzeugungsfunktion G() eine kryptographische Hashing-Funktion sein. Auf die G()-Funktion wird im gesamten Rest des Patents Bezug genommen, um die Erzeugung von Schlüsseln zu beschreiben, wie beispielsweise den Prozessor-Nub-Schlüssel, dem OS-Nub-Schlüssel und Applet-Schlüssel, was später im Detail erörtert wird.

[0059] Der Schlüsselauswähler **220** führt eine Schlüsselauswahlfunktion S() aus, um einen dritten Schlüssel (der mit K3 bezeichnet ist) aus dem zweiten Schlüssel K2 zu erzeugen. Die Gleichung nimmt die Form K3 = S(K2) an. Die Schlüsselauswahlfunktion S() erzeugt K3, so daß K3 ein ausreichend kleiner symmetrischer Chiffreschlüssel ist, der in einem gewünschten symmetrischen Chiffre verwendet werden kann. Bei einem Ausführungsbeispiel ist der erzeugte dritte Schlüssel K3 der symmetrische Chiffreschlüssel **240** für den geladenen Softwarecode. Wie zuvor erörtert, legt der Schlüsselauswähler **220** die Schlüsselauswahlfunktion S() an, um einen symmetrischen Chiffreschlüssel **240** (K3) der geeigneten Größe (z. B. 112, 168, 128 oder 256 Bits) aus dem größeren Schlüssel **230** des geladenen Softwarecodes auszuwählen, um den symmetrischen Chiffreschlüssel **240** für den geladenen Softwarecode zu erzeugen. Auf die S()-Funktion wird im Rest des Patents Bezug genommen, um die Auswahl von Chiffreschlüsseln zu beschreiben, wie beispielsweise des Prozessor-Nub-Chiffreschlüssels, des OS-Nub-Chiffreschlüssels, und von Applet-Chiffreschlüsseln, was später detaillierter erörtert werden wird.

[0060] Jedes geladene Softwaremodul, das sich aktuell unter der Kontrolle der isolierten Ausführungsumgebung befindet, weist folglich wenigstens zwei Schlüssel auf, den Schlüssel **230** des geladenen Softwarecodes und den symmetrischen Chiffreschlüssel **240** für den geladenen Softwarecode. Wenn mehrere Verschlüsselungsalgorithmen von dem Softwaremodul benutzt werden, könnte der Softwaremodul mehrere symmetrische Chiffreschlüssel, die in ihrer Form dem symmetrischen Chiffreschlüssel **240** ähnlich sind, aufweisen. Wenn es nur einen aus dem Schlüssel **230** des geladenen Softwarecodes auszuwählenden Schlüssel gibt, kann diese Auswahl die Exklusiv-ODER-Verknüpfung eines oder mehrerer getrennter Segmente des längeren Schlüssels verwenden, wobei diese Segmente die Länge des gewünschten Schlüssels sind. Wenn mehrere kryptographische Schlüssel erzeugt werden sollen, könnte eine Funktion, wenn beispielsweise ein HMAC für S() unter Verwendung eines großen zufällig erzeugten Werts, der für den Algorithmus, für welchen der Schlüssel erzeugt wird, spezifisch ist, als Schlüssel verwendet werden.

Die Schlüsselhierarchie

[0061] Fig. 3 ist ein Schema, das einen Prozeß **300** zum Erzeugen einer Schlüsselhierarchie **301** gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht. Der Prozeß **300** erzeugt die Schlüsselhie-

rarchie **301**, welche eine Mehrzahl symmetrischer Schlüssel umfaßt, die von dem Chiffreschlüsselerzeuger **200**, wie er zuvor erörtert wurde, erzeugt werden. Die Schlüssel werden benutzt, um Geheimnisse an speziellen Code, der in isolierter Ausführung arbeitet, zu binden.

[0062] Der isolierte Ausführungsbereich (z. B. isolierter Ausführungsring-0 **15** und isolierter Ausführungsring-3 **45**, wie sie in den Fig. 1A und Fig. 1B gezeigt sind) ist in Schichten bestückt. Besonders relevant für die isolierte Ausführung und die nachfolgende Diskussion sind der Prozessor-Nub **18**, der OS-Nub **16** und Applets **46** (die zuvor detailliert unter Bezugnahme auf die Fig. 1A und Fig. 1B der Architekturübersicht des Patents erörtert worden sind). Als erstes wird der Prozessor-Nub **18** geladen und übernimmt die Kontrolle über die isolierte Ausführungsumgebung. Der Prozeß **300** erzeugt zunächst einen Prozessor-Nub-Schlüssel (PNK) **325** mit dem Schlüsselgenerator **210**. Der Schlüsselgenerator **210** verwendet den Anfangsschlüssel (IK; Initial Key) **310** und den Prozessor-Nub-ID (PNID) **320** als Eingaben. Der Schlüsselgenerator **210** führt eine Erzeugungsfunktion wie folgt durch: PNK = G(IK, PNID), wobei IK **310** der für die Plattform spezifische Anfangsschlüssel und der PNID **320** der ID des Prozessor-Nubs ist. Wie zuvor erörtert, kann die Erzeugungsfunktion G() eine kryptographische Hashing-Funktion sein. Der ID kann der Hash des Codes selbst oder irgendein zertifizierter ID-Wert sein, wie oben erörtert wurde. Jedoch muß dann, wenn er ein zertifizierter Wert ist, der Prozessor-Nub-Lader in der Lage sein, das Zertifikat, das diesen ID-Wert hält, zu validieren.

[0063] Als nächstes erzeugt der Prozeß **300** einen Prozessor-Nub-Chiffreschlüssel (PNCK) **327** unter Benutzung des Schlüsselauswählers **220**. Der Schlüsselauswähler **220** führt eine Schlüsselauswahlfunktion wie folgt durch: PNCK = S(PNK), wobei PNK **325** der Prozessor-Nub-Schlüssel ist. Wie zuvor erörtert, erzeugt die Schlüsselauswahlfunktion S() einen Chiffreschlüssel, der klein genug ist, so daß er in einem gewünschten symmetrischen Chiffre benutzt werden kann. Beispielsweise wendet der Schlüsselauswähler **220** die Schlüsselauswahlfunktion S() an, um einen symmetrischen Schlüssel PNCK **327** der geeigneten Größe (z. B. 112, 168, 128 oder 256 Bits) aus dem PNK **325** auszuwählen. Der Prozessor-Nub **18** verwendet den PNCK **327**, um seine Geheimnisse gegenüber dem OS-Nub **16**, den Applets **46** und nicht-isolierten Ausführungscode zu schützen.

[0064] Bei einem Ausführungsbeispiel wird der Anfangsschlüssel (IK) **310** durch einen externen Hardware-Zufallszahlengenerator erzeugt und in Schmelzsicherungen des Eingabe/Ausgabe-Controller-Hub (ICH) **150** (Fig. 1C) zum Zeitpunkt der Herstellung programmiert. Bei einem anderen Ausführungsbeispiel wird der IK **310** von der Plattform **100** selbst erzeugt, wenn die Plattform erstmalig eingeschaltet wird, unter Verwendung des internen Hardware-Zufallszahlengenerators **166** (Fig. 1C) der Plattform und wird dann in dem nicht-flüchtigen Speicher des Kryptographischer-Schlüssel-Speichers **155** in dem ICH **150** (Fig. 1C) gespeichert. Dieses Ausführungsbeispiel reduziert die Chance, daß der IK **310** während der Herstellung gefährdet wird. In beiden Fällen wird der IK **310** in dem geschützten nicht-flüchtigen Speicher des Kryptographischer-Schlüssel-Speichers **155** gespeichert. Es ist jedoch klar, daß der IK **310** in einem geschützten nicht-flüchtigen Speicher in anderen Bereichen der Plattform gespeichert werden könnte.

[0065] Wenn weitere Softwaremodule (z. B. der OS-Nub **16**, Applets **46**, etc.) in den isolierten Ausführungsbereich geladen werden, ersetzen die Schlüssel der neu geladenen Module die Schlüssel des Ladungsmoduls. Jedoch wird der IK **310** nicht ersetzt, sondern durch Setzen eines von Hardware setzbaren Flags unzugreifbar gemacht, was ein Lesen des Speicherplatzes verhindert. Das Flag wird ausschließlich durch die totale Reinitialisierung des isolierten Bereichs gelöscht. Erzeugte Schlüssel werden überschrieben und stehen somit nicht mehr zur Verfügung.

[0066] Wenn der Prozessor-Nub **18** erstmalig geladen wird, veranlaßt der Prozeß **300**, daß der Prozessor-Nub-Schlüssel (PNK) **325** und der Prozessor-Nub-Chiffreschlüssel (PNCK) **327** erzeugt werden, wie zuvor erörtert wurde. Wenn der Prozessor-Nub irgendwelche Geheimnisse speichern muß, auf die nur er zugreifen kann, kann er den PNCK **327** als kryptographischen Schlüssel für diesen Zweck benutzen. Wenn der Prozessor-Nub entscheidet, daß der OS-Nub **16** geladen und die Kontrolle ihm übertragen wird, werden der OS-Nub-Schlüssel (OSNK) **335** und der OS-Nub-Chiffreschlüssel (OSNCK) **337** erzeugt, wobei der gleiche Prozeß **300** verwendet wird, wie er zuvor zum Erzeugen des PNK **325** und des PNCK **327** beschrieben worden ist. Darüber hinaus ersetzen der OSNK **335** und der OSNCK **337** den PNK **325** und den PNCK **327** in dem Kryptographischer-Schlüssel-Speicher. Der OSNCK **337** wird von dem OS-Nub verwendet, um dessen Geheimnisse gegenüber anderen OS-Nubs, Applets und nicht-isoliertem Ausführungscode zu schützen. Wenn es jetzt irgendwelche Geheimnisse gibt, die für eine zukünftige Verwendung durch den OS-Nub verfügbar gemacht werden sollen, können diese durch den OSNCK **337** verschlüsselt werden. Jedoch sind irgendwelche Geheimnisse, die unter dem PNCK **327** verschlüsselt worden sind, nicht für den OS-Nub oder irgendeine andere Software, die auf diese Plattform geladen werden soll, verfügbar.

[0067] Dieser Prozeß **300** kann aufeinanderfolgend für beliebige und sämtliche Ebenen der Softwareco-

deladung wiederholt werden. Fig. 3 zeigt vier Ebenen oder Schichten für: den Prozessor-Nub, den OS-Nub, das Applet-1 und den Applet-1-Sub-Prozeß. Der Prozeß **300** erzeugt den Applet-1-Schlüssel (A1K) **345** und den Applet-1-Chiffreschlüssel (A1CK) **347** für das Applet-1 und den Applet-1-Sub-Prozeß-Schlüssel (A1SK) **355** und den Applet-1-Sub-Prozeß-Chiffreschlüssel (A1SCK) **357** für einen Applet-1-Sub-Prozeß unter Verwendung des gleichen Prozesses **300**, wie er zuvor zum Erzeugen des Prozessor-Nub- und OS-Nub-Schlüssels beschrieben worden ist. Es ist klar, daß die Schlüsselhierarchie **301** und der Prozeß **300** nur Beispiele sind und daß andere Schlüsselhierarchien sowie Prozesse zum Erzeugen dieser Schlüsselhierarchien zum Schützen von Geheimnissen in einer isolierten Ausführungsumgebung möglich sind.

Ein anderes Ausführungsbeispiel für das Back-up und die Wiederherstellung von Daten

[0068] Fig. 4 ist ein Schema, das ein System **400** zum Sichern eines Anfangsschlüssels zum Ermöglichen eines Back-up und der Wiederherstellung von Daten gemäß einem Ausführungsbeispiel der Erfindung veranschaulicht.

[0069] Wie oben beschrieben, ist der Anfangsschlüssel (IK) **310** für jede Plattform einzigartig und in diese Plattform eingeschlossen (d. h., er ist permanent in dem geschützten nicht-flüchtigen Speicher des Kryptographischer-Schlüssel-Speichers **155** (Fig. 1C) gespeichert). Unglücklicherweise gibt es bei diesem Ausführungsbeispiel keine Möglichkeit, den IK **310** zu sichern oder wiederherzustellen. Ein anderes Ausführungsbeispiel der Erfindung gestattet, daß der IK **310** gesichert (backed-up) und wiederhergestellt wird, wie es in Fig. 4 gezeigt ist. Dies ist für einen Benutzer wünschenswert, der wertvolle Geheimnisse an die Plattform bindet und dann seine Hardware zu irgendeinem späteren Zeitpunkt aufzurüsten wünscht. Die Aufrüstung könnte geschehen, weil der Plattformbesitzer beschlossen hat, eine neuere, schnellere Hardware zu verwenden. Alternativ könnte die Aufrüstung geschehen, da die Plattform plötzlich und katastrophal ausfällt und ersetzt werden muß.

[0070] Bei diesem Ausführungsbeispiel gibt es ein Schlüssel-gesichert-Flag **402** in einem nicht-flüchtigen Speicher, das immer dann gelöscht wird, wenn ein neuer Anfangsschlüssel (IK) **310** erzeugt wird (aber nicht, wenn er wiederhergestellt wird). Solange dieses Flag gelöscht ist, ist es dem Verschlüsseler **404** gestattet, den IK **310** zu verschlüsseln. Der Verschlüsseler **404** führt eine Verschlüsselungsfunktion zum Entschlüsseln des IK **310** unter Verwendung eines kryptographischen Schlüssels **420**, der von dem Benutzer (vermutlich dem Plattformbesitzer) zur Verfügung gestellt wird, aus. Das verschlüsselte Ergebnis ist der gesicherte IK **410**. Der gesicherte IK **410** kann durch jedermann gelesen werden. Insbesondere kann der gesicherte IK **410** durch normale Back-up-Werkzeuge gesichert werden. Wenn der gesicherte IK **410** erzeugt wird, wird das Schlüssel-gesichert-Flag **402** gesetzt, was jegliche zukünftige Sicherung des IK **310** verhindert.

[0071] Zu jedem beliebigen zukünftigen Zeitpunkt ist der Benutzer in der Lage, den richtigen kryptographischen Schlüssel **420** und einen früher gesicherten IK **410** einem Entschlüsseler **430** zur Verfügung zu stellen, der eine Entschlüsselungsfunktion durchführt, um den IK **310** zu überschreiben. Dies ermöglicht einem Plattformbesitzer, eine fehlerhafte Plattform einschließlich sämtlicher Geheimnisse, die in den isolierten Bereich der Plattform eingebunden wurden, zu ersetzen. Die Verschlüsselungs- und Entschlüsselungsfunktionen des Verschlüsselers **404** bzw. des Entschlüsselers **430** können allgemein akzeptable symmetrische kryptographische Funktionen sein, wie beispielsweise Triple-DES (Data Encryption Standard) oder AES (der neue Advanced Encryption Standard, der von dem NIST (National Institute of Standards and Technology) ausgewählt wurde).

[0072] Fig. 5 ist ein Ablaufdiagramm, das den Prozeß des Erzeugens einer Schlüsselhierarchie gemäß einem Ausführungsbeispiel der Erfindung näher veranschaulichen.

[0073] Bei START erzeugt der Prozeß **500** einen Anfangsschlüssel, der für die Plattform einzigartig ist (beispielsweise auf der Grundlage einer Zufallszahl) (Block **510**). Dann speichert der Prozeß **500** den Anfangsschlüssel (Block **520**). Als nächstes erzeugt der Prozeß **500** einen Prozessor-Nub-Schlüssel, indem er den Anfangsschlüssel mit dem ID des Prozessor-Nubs einer Hash-Operation unterzieht (Block **530**). Der Prozeß **500** wählt dann einen kleineren Prozessor-Nub-Chiffreschlüssel auf der Grundlage des Prozessor-Nub-Schlüssels aus (Block **540**). Bei Block **550** erzeugt der Prozeß **500** einen OS-Nub-Schlüssel, indem er den Prozessor-Nub-Schlüssel mit dem ID des OS-Nubs einer Hash-Operation unterzieht. Dann wählt der Prozeß **500** einen kleineren OS-Nub-Chiffreschlüssel auf der Grundlage des OS-Nub-Schlüssels aus (Block **560**). Als nächstes erzeugt der Prozeß **500** einen Applet-Schlüssel, indem er den OS-Nub-Schlüssel mit dem ID des Applets einer Hash-Operation unterzieht (Block **570**). Bei Block **580** wählt der Prozeß **500** einen kleineren Applet-Chiffreschlüssel auf der Grundlage des Applet-Schlüssels aus. Der Prozeß **500** wird dann beendet.

[0074] Während diese Erfindung unter Bezugnahme auf veranschaulichende Ausführungsbeispiele beschrieben wurde, soll diese Beschreibung nicht in einem einschränkenden Sinne ausgelegt werden.

Patentansprüche

1. Einrichtung zum Erzeugen einer Schlüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung einer geschützten Plattform (**100**), wobei die geschützte Plattform einen Prozessor aufweist, der entweder in einem normalen Ausführungsmodus oder in einem isolierten Ausführungsmodus konfiguriert ist, wobei die die Einrichtung aufweist:
einen Schlüsselspeicher (**155**) zum Speichern eines für die geschützte Plattform einzigartigen Anfangsschlüssels (**310**); und
einen Chiffreschlüsselerzeuger (**200**) in der geschützten Plattform (**100**), der eine Hierarchie von Schlüsseln (**327**, **337**, **347**, **357**) auf der Grundlage des Anfangsschlüssels (**310**) erzeugt, wobei der Chiffreschlüsselerzeuger (**200**) einen Schlüsselgenerator (**210**) für eine Hash-Operation eines ID (**228**; **320**, **330**, **340**, **350**) eines geladenen Softwarecodes mit dem Anfangsschlüssel (**310**) oder einem Schlüssel (**225**; **325**, **335**, **345**) von ladendem Softwarecode zum Erzeugen eines Schlüssels (**230**; **325**, **335**, **345**, **355**) des geladenen Softwarecodes und einen Schlüsselauswähler (**220**) zum Auswählen eines kleineren symmetrischen Chiffreschlüssels (**240**; **327**, **337**, **347**, **357**) auf der Grundlage des Schlüssels (**230**; **325**, **335**, **345**, **355**) des geladenen Softwarecodes aufweist.

2. Einrichtung nach Anspruch 1, wobei der Anfangsschlüssel (**310**) in durchtrennbaren Leitungsverbindungen eines Eingabe/Ausgabe-Steuer-Hubs (ICH) (**150**) programmiert ist.

3. Einrichtung nach Anspruch 2, wobei der Anfangsschlüssel (**310**) auf einer Zufallszahl basiert.

4. Einrichtung nach Anspruch 1, wobei der Anfangsschlüssel (**310**) von der Plattform (**100**) erzeugt wird, wenn die Plattform (**100**) erstmalig eingeschaltet wird.

5. Einrichtung nach Anspruch 4, wobei der Anfangsschlüssel (**310**) auf einer von einem Zufallszahlengenerator (**166**) der Plattform (**100**) erzeugten Zufallszahl basiert.

6. Einrichtung nach Anspruch 1, wobei der Schlüsselgenerator (**210**) einen Prozessor-Nub-Schlüssel (**325**) erzeugt, indem er den Anfangsschlüssel (**310**) mit einem Prozessor-Nub-ID (**320**) einer Hash-Operation unterzieht und der Schlüsselauswähler (**220**) einen kleineren Prozessor-Nub-Chiffreschlüssel (**327**) auf der Grundlage des Prozessor-Nub-Schlüssels (**325**) auswählt.

7. Einrichtung nach Anspruch 6, wobei der Schlüsselgenerator (**210**) einen OS-Nub-Schlüssel (**335**) erzeugt, indem er den Prozessor-Nub-Schlüssel (**325**) mit einem OS-Nub-ID (**330**) einer Hash-Operation unterzieht und der Schlüsselauswähler (**220**) einen kleineren OS-Nub-Chiffreschlüssel (**327**) auf der Grundlage des OS-Nub-Schlüssels (**335**) auswählt.

8. Einrichtung nach Anspruch 7, wobei der Schlüsselgenerator (**210**) einen Applet-Schlüssel (**345**) erzeugt, indem er den OS-Nub-Schlüssel (**335**) mit einem Applet-ID (**340**) einer Hash-Operation unterzieht und der Schlüsselauswähler (**220**) einen Applet-Chiffreschlüssel (**347**) auf der Grundlage des Applet-Schlüssels (**345**) auswählt.

9. Verfahren zum Erzeugen einer Schlüsselhierarchie zur Verwendung in einer isolierten Ausführungsumgebung einer geschützten Plattform, wobei die geschützte Plattform einen Prozessor aufweist, der entweder in einem normalen Ausführungsmodus oder einem isolierten Ausführungsmodus konfiguriert ist, wobei:
ein Anfangsschlüssel gespeichert wird;
eine Schlüsselhierarchie auf der Grundlage des Anfangsschlüssels in der geschützten Plattform erzeugt wird, indem jeweils ein ID eines geladenen Softwarecodes mit dem Anfangsschlüssel bzw. nachfolgend mit einem Schlüssel eines ladenden Softwarecodes einer Hash-Operation unterzogen wird, um den Schlüssel des geladenen Softwarecodes zu erzeugen, und dann ein kleinerer symmetrischer Chiffreschlüssel aus dem Schlüssel des geladenen Softwarecodes ausgewählt wird.

10. Verfahren nach Anspruch 9, wobei der Anfangsschlüssel für die Plattform einzigartig ist.

11. Verfahren nach Anspruch 10, wobei das Speichern des Anfangsschlüssels das Programmieren des Anfangsschlüssels in Schmelzelemente eines Eingabe/Ausgabe-Steuer-Hubs (ICH) umfaßt.

12. Verfahren nach Anspruch 11, wobei der Anfangsschlüssel auf einer Zufallszahl basiert.

13. Verfahren nach Anspruch 10, wobei der Anfangsschlüssel unter Benutzung der Plattform erzeugt wird, wenn die Plattform erstmalig eingeschaltet wird.

14. Verfahren nach Anspruch 13, wobei das Erzeugen des Anfangsschlüssels umfaßt:
Erzeugen einer Zufallszahl mit einem Zufallszahlengenerator der Plattform;
Erzeugen des Anfangsschlüssels auf der Grundlage der Zufallszahl; und
Speichern des Anfangsschlüssels in dem Schlüsselspeicher.

15. Verfahren nach Anspruch 9, wobei ein Prozessor-Nub-Schlüssel erzeugt wird, indem der Anfangsschlüssel mit einem Prozessor-Nub-ID einer

Hash-Operation unterzogen wird; und ein kleinerer Prozessor-Nub-Chiffreschlüssel aus dem Prozessor-Nub-Schlüssel ausgewählt wird.

16. Verfahren nach Anspruch 15, wobei
ein OS-Nub-Schlüssel erzeugt wird, indem der Prozessor-Nub-Schlüssel mit einem OS-Nub-ID einer Hash-Operation unterzogen wird; und
ein kleinerer OS-Nub-Chiffreschlüssel aus dem OS-Nub-Schlüssel ausgewählt wird.

17. Verfahren nach Anspruch 16, wobei
ein Applet-Schlüssel erzeugt wird, indem der OS-Nub-Schlüssel mit einem Applet-ID einer Hash-Operation unterzogen wird; und
ein kleinerer Applet-Chiffreschlüssel aus dem Applet-Schlüssel ausgewählt wird.

18. Geschützte Plattform (**100**), aufweisend:
einen Chipsatz (**130**, **150**);
einen mit dem Chipsatz (**130**, **150**) gekoppelten Speicher (**140**), der einen isolierten Speicherbereich (**70**) aufweist;
einen mit dem Chipsatz (**130**, **150**) und dem Speicher (**140**) gekoppelten Prozessor (**110**), wobei der Prozessor einen normalen Ausführungsmodus und einen isolierten Ausführungsmodus aufweist, wobei der Prozessor (**110**) auf den isolierten Speicherbereich (**70**) zugreift, wenn sich der Prozessor (**110**) in dem isolierten Ausführungsmodus befindet; und
eine Einrichtung nach einem der Ansprüche 1 bis 8.

Es folgen 7 Blatt Zeichnungen

Anhängende Zeichnungen

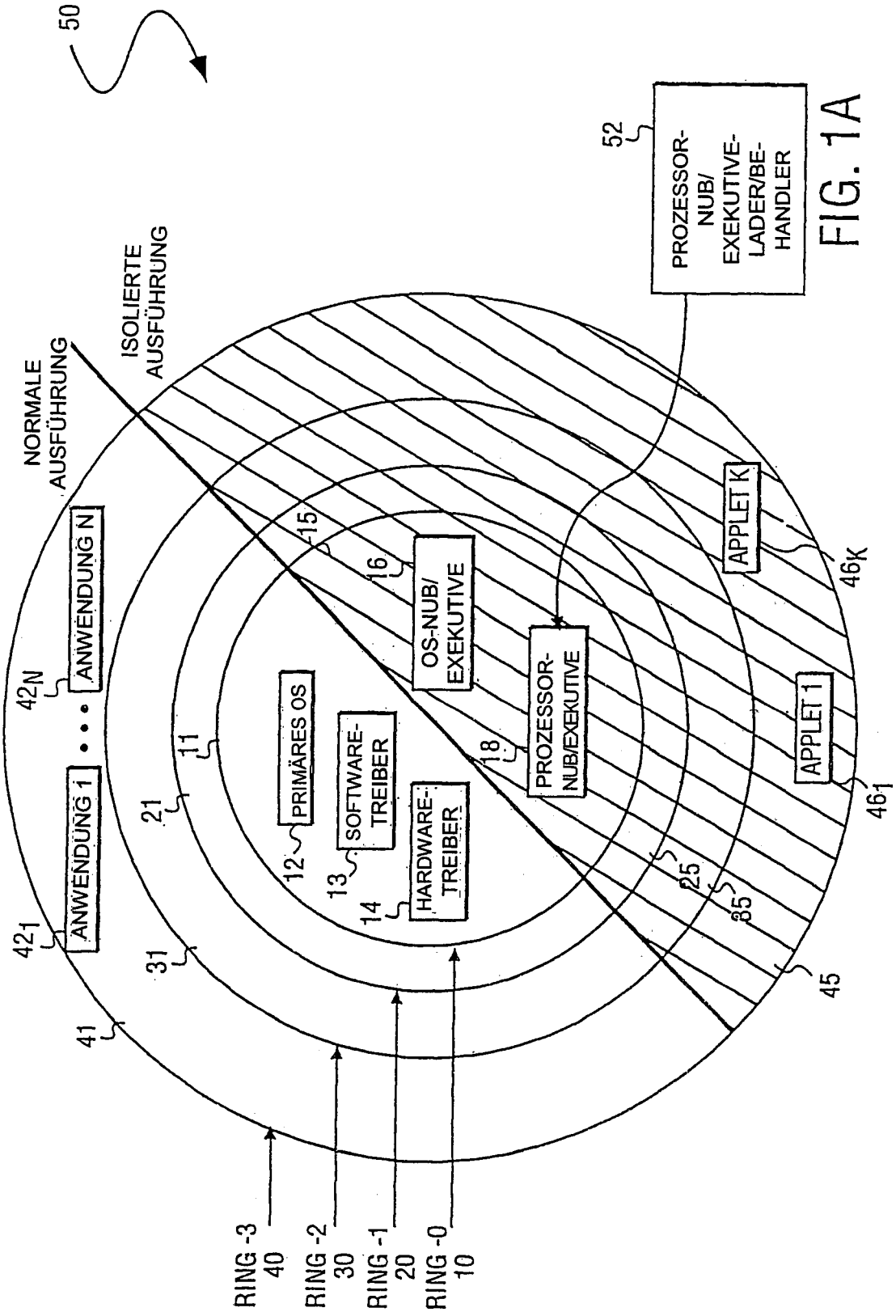


FIG. 1A

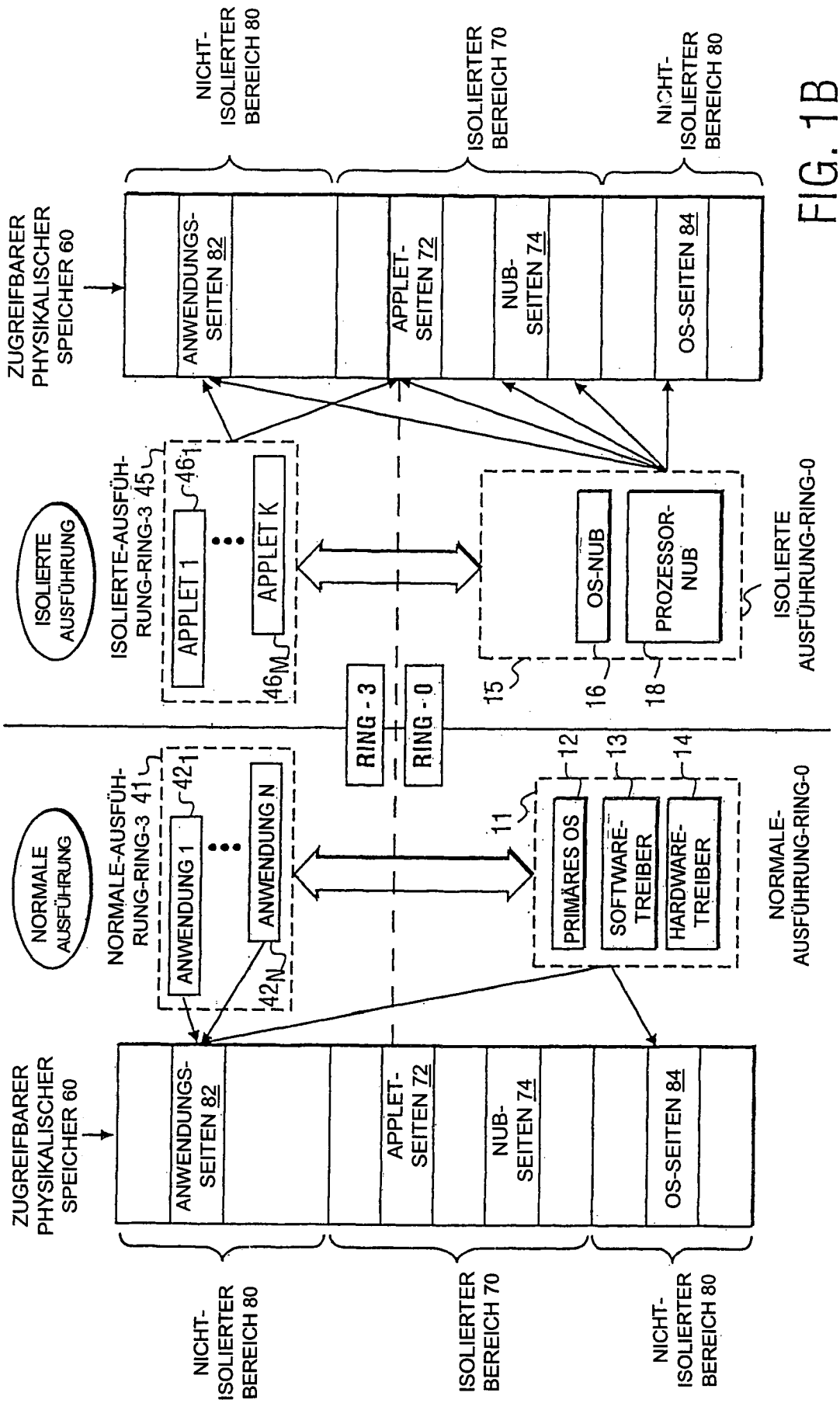

ZUGREIFBARER PHYSIKALISCHER SPEICHER 60
NICHT-ISOLIERTER BEREICH 80
ISOLIERTER BEREICH 70
NICHT-ISOLIERTER BEREICH 80
ANWENDUNGS-SEITEN 82
APPLET-SEITEN 72
NUB-SEITEN 74
OS-SEITEN 84
ISOLIERTE AUSFÜHRUNG
ISOLIERTE-AUSFÜHRUNG-RING-3 45
APPLET 1
APPLET K
46 1
46 M
OS-NUB
PROZESSOR-NUB
ISOLIERTE AUSFÜHRUNG-RING-0
15
16
18
RING - 3
RING - 0
NORMALE AUSFÜHRUNG
NORMALE-AUSFÜHRUNG-RING-3 41
ANWENDUNG 1
ANWENDUNG N
42 1
42 N
11
12
13
14
PRIMÄRES OS
SOFTWARE-TREIBER
HARDWARE-TREIBER
NORMALE-AUSFÜHRUNG-RING-0

FIG. 1B

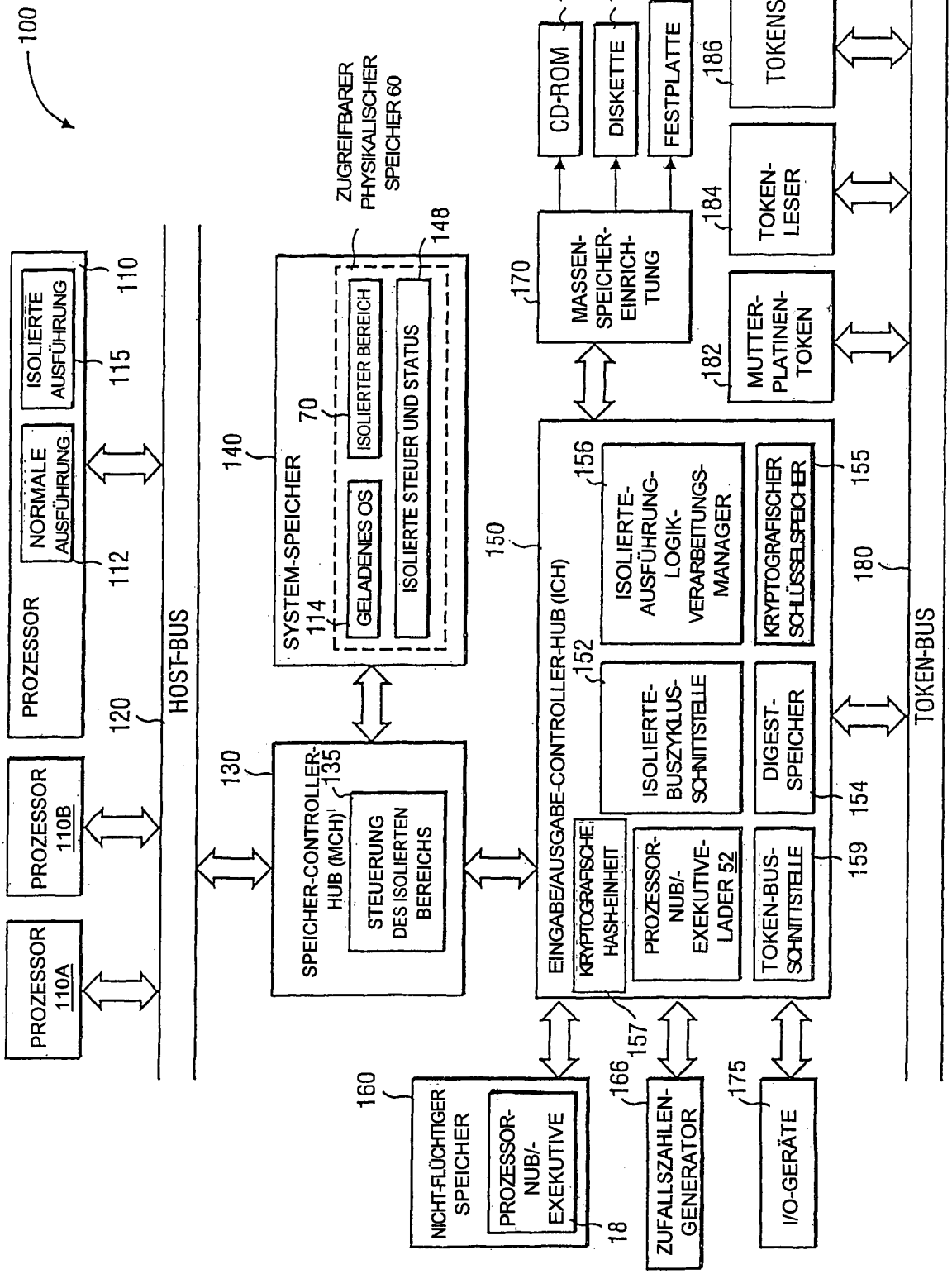

100
PROZESSOR 110A
PROZESSOR 110B
PROZESSOR
NORMALE AUSFÜHRUNG
ISOLIERTE AUSFÜHRUNG
110
112
115
120
HOST-BUS
130
SPEICHER-CONTROLLER-HUB (MCH)
135
STEUERUNG DES ISOLIERTEN BEREICHS
140
SYSTEM-SPEICHER
70
114
GELADENES OS
ISOLIERTER BEREICH
ISOLIERTE STEUER UND STATUS
148
ZUGREIFBARER PHYSIKALISCHER SPEICHER 60
150
EINGABE/AUSGABE-CONTROLLER-HUB (ICH)
152
ISOLIERTE-BUSZYKLUS-SCHNITTSTELLE
156
ISOLIERTE-AUSFÜHRUNG-LOGIK-VERARBEITUNGS-MANAGER
KRYPTOGRAFISCHE HASH-EINHEIT
157
PROZESSOR-NUB/-EXEKUTIVE-LADER 52
DIGEST-SPEICHER
154
KRYPTOGRAFISCHER SCHLÜSSELSPEICHER
155
TOKEN-BUS-SCHNITTSTELLE
159
180
TOKEN-BUS
160
NICHT-FLÜCHTIGER SPEICHER
PROZESSOR-NUB/-EXEKUTIVE
18
166
ZUFALLSZAHLEN-GENERATOR
175
I/O-GERÄTE
170
MASSEN-SPEICHER-EINRICH-TUNG
172
CD-ROM
174
DISKETTE
176
FESTPLATTE
182
MUTTER-PLATINEN-TOKEN
184
TOKEN-LESER
186
TOKENS

FIG. 1C

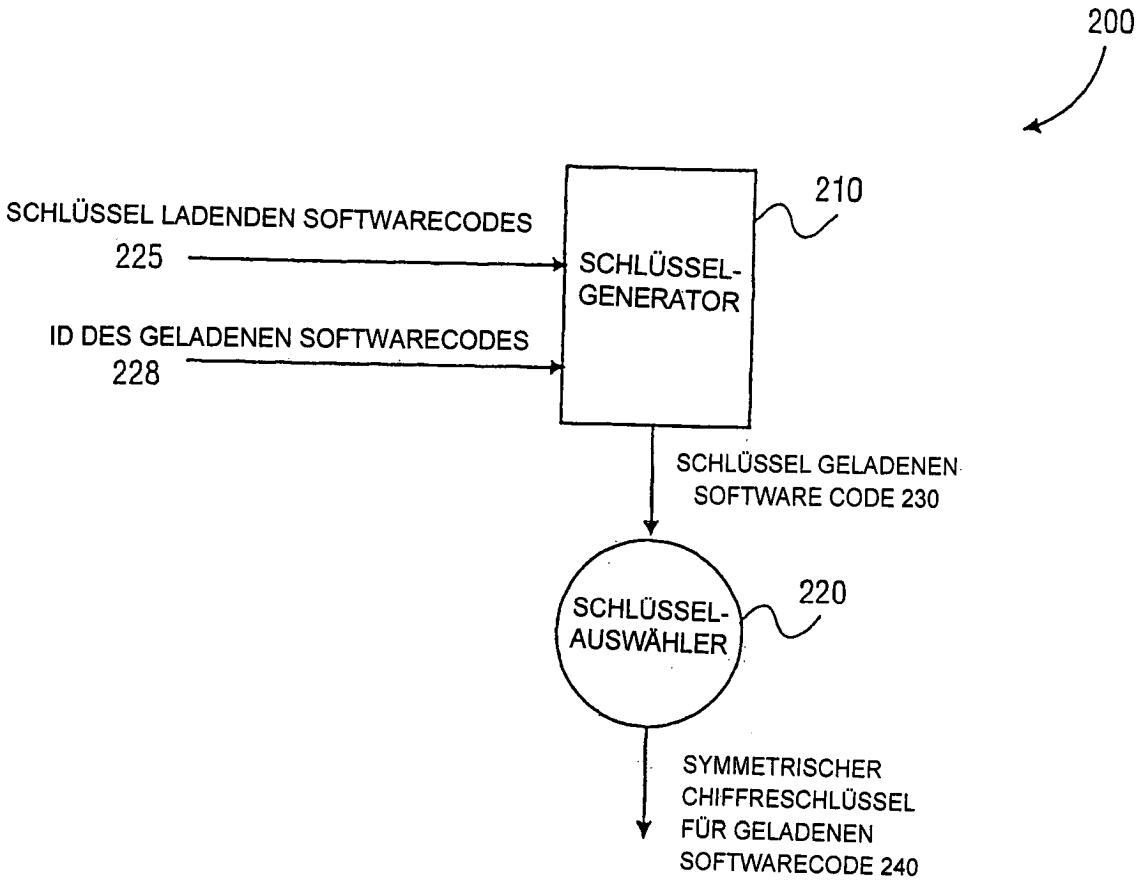

200
SCHLÜSSEL LADENDEN SOFTWARECODES
225
ID DES GELADENEN SOFTWARECODES
228
SCHLÜSSEL-
GENERATOR
210
SCHLÜSSEL GELADENEN
SOFTWARE CODE 230
SCHLÜSSEL-
AUSWÄHLER
220
SYMMETRISCHER
CHIFFRESCHLÜSSEL
FÜR GELADENEN
SOFTWARECODE 240

FIG. 2

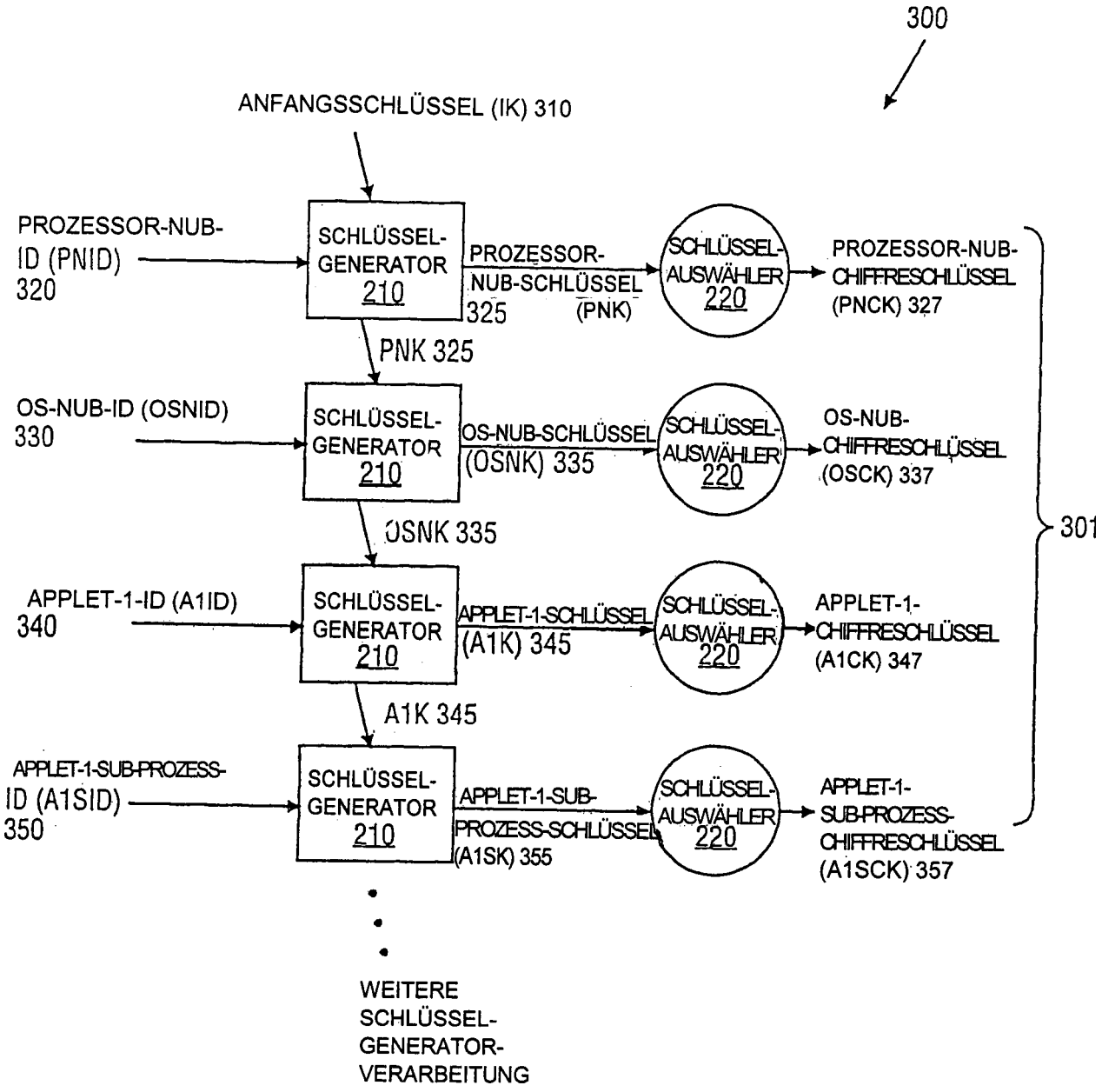

300
ANFANGSSCHLÜSSEL (IK) 310
PROZESSOR-NUB-ID (PNID) 320
SCHLÜSSEL-GENERATOR 210
PROZESSOR-NUB-SCHLÜSSEL (PNK) 325
SCHLÜSSEL-AUSWÄHLER 220
PROZESSOR-NUB-CHIFFRESCHLÜSSEL (PNCK) 327
PNK 325
OS-NUB-ID (OSNID) 330
SCHLÜSSEL-GENERATOR 210
OS-NUB-SCHLÜSSEL (OSNK) 335
SCHLÜSSEL-AUSWÄHLER 220
OS-NUB-CHIFFRESCHLÜSSEL (OSCK) 337
OSNK 335
APPLET-1-ID (A1ID) 340
SCHLÜSSEL-GENERATOR 210
APPLET-1-SCHLÜSSEL (A1K) 345
SCHLÜSSEL-AUSWÄHLER 220
APPLET-1-CHIFFRESCHLÜSSEL (A1CK) 347
A1K 345
APPLET-1-SUB-PROZESS-ID (A1SID) 350
SCHLÜSSEL-GENERATOR 210
APPLET-1-SUB-PROZESS-SCHLÜSSEL (A1SK) 355
SCHLÜSSEL-AUSWÄHLER 220
APPLET-1-SUB-PROZESS-CHIFFRESCHLÜSSEL (A1SCK) 357
301
WEITERE SCHLÜSSEL-GENERATOR-VERARBEITUNG

FIG. 3

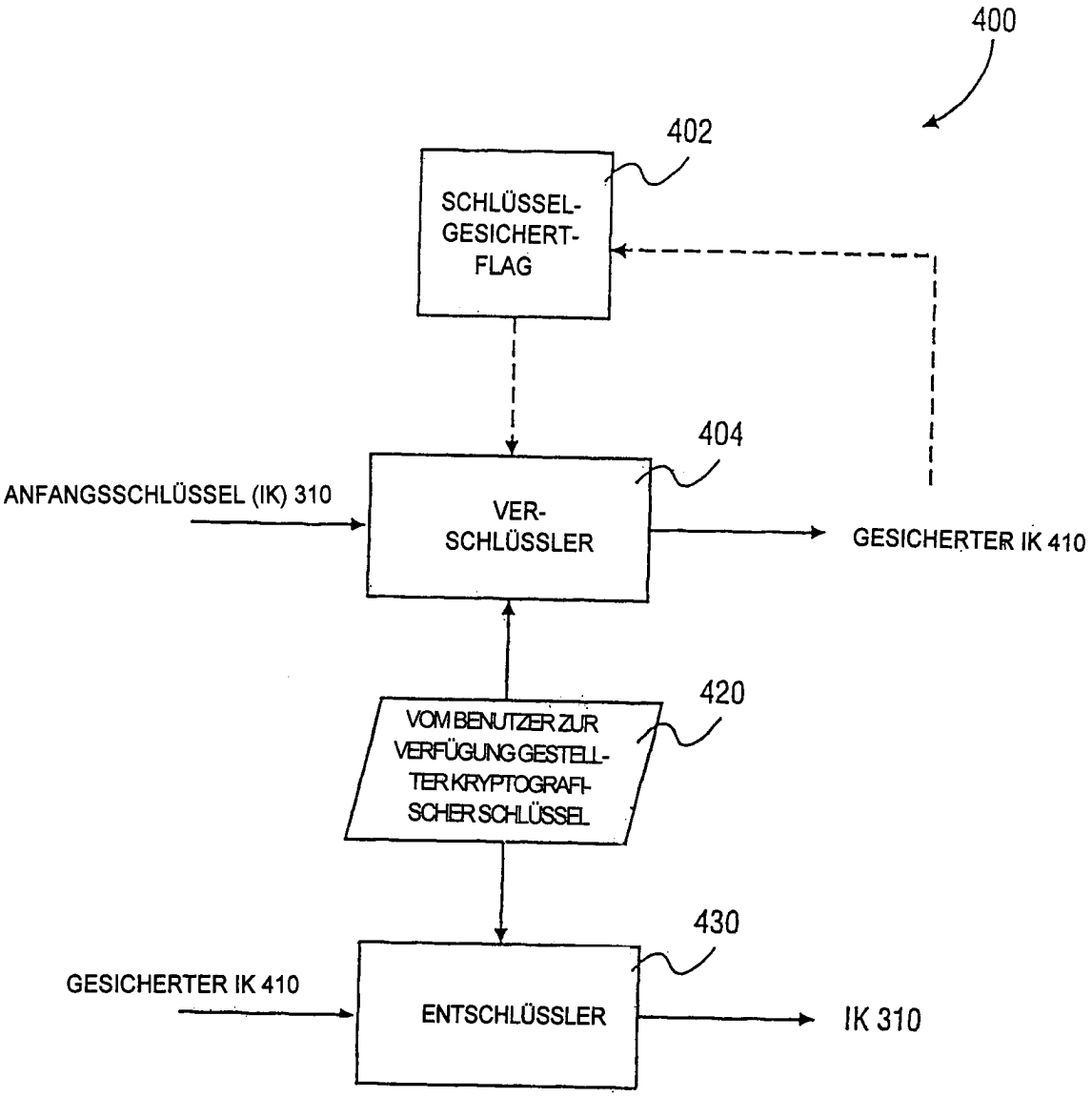

400
402
SCHLÜSSEL-GESICHERT-FLAG
404
ANFANGSSCHLÜSSEL (IK) 310
VER-SCHLÜSSLER
GESICHERTER IK 410
420
VOM BENUTZER ZUR VERFÜGUNG GESTELLTER KRYPTOGRAFISCHER SCHLÜSSEL
430
GESICHERTER IK 410
ENTSCHLÜSSLER
IK 310

FIG. 4

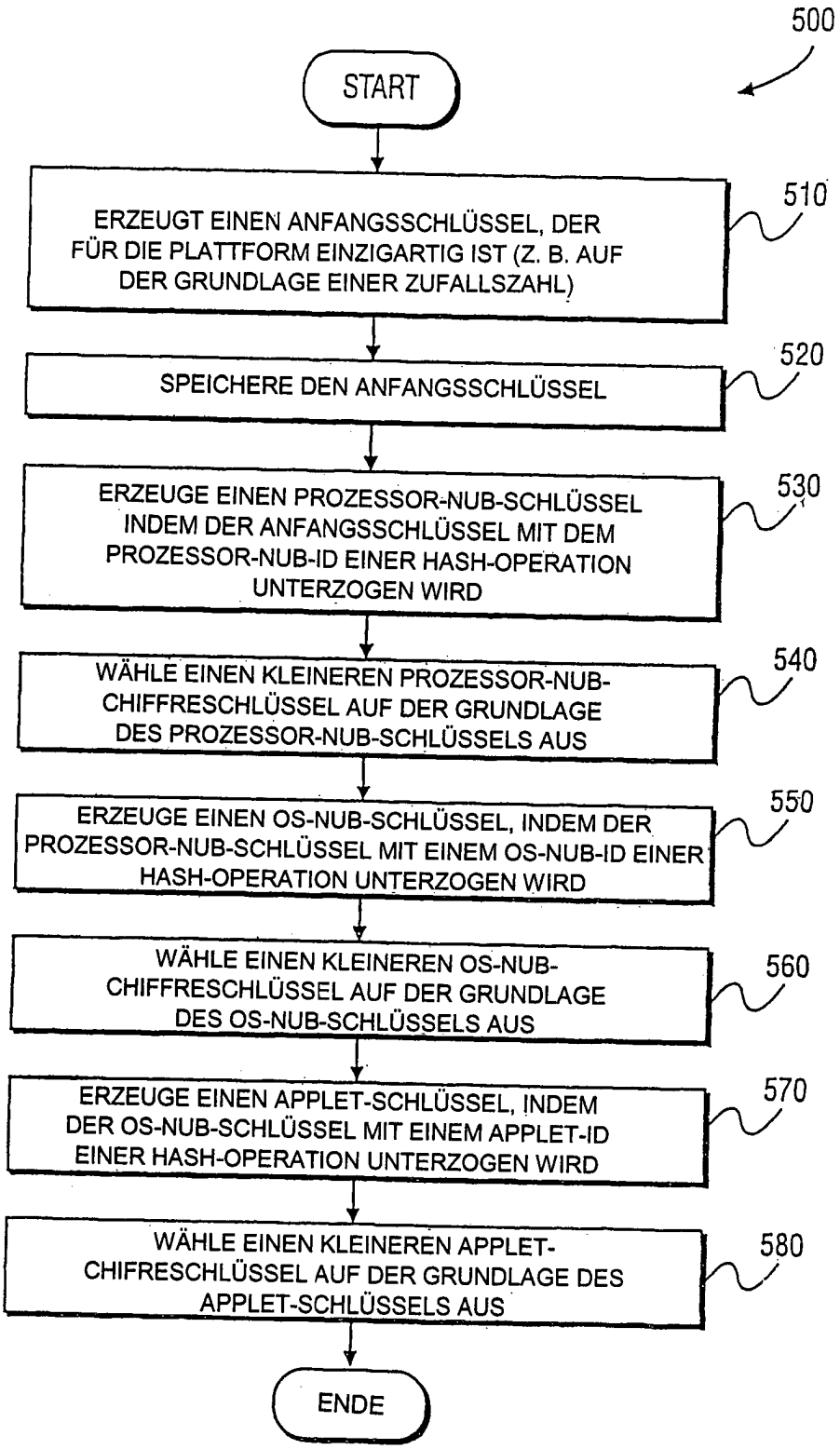

500
START
ERZEUGT EINEN ANFANGSSCHLÜSSEL, DER FÜR DIE PLATTFORM EINZIGARTIG IST (Z. B. AUF DER GRUNDLAGE EINER ZUFALLSZAHL)
510
SPEICHERE DEN ANFANGSSCHLÜSSEL
520
ERZEUGE EINEN PROZESSOR-NUB-SCHLÜSSEL INDEM DER ANFANGSSCHLÜSSEL MIT DEM PROZESSOR-NUB-ID EINER HASH-OPERATION UNTERZOGEN WIRD
530
WÄHLE EINEN KLEINEREN PROZESSOR-NUB-CHIFFRESCHLÜSSEL AUF DER GRUNDLAGE DES PROZESSOR-NUB-SCHLÜSSELS AUS
540
ERZEUGE EINEN OS-NUB-SCHLÜSSEL, INDEM DER PROZESSOR-NUB-SCHLÜSSEL MIT EINEM OS-NUB-ID EINER HASH-OPERATION UNTERZOGEN WIRD
550
WÄHLE EINEN KLEINEREN OS-NUB-CHIFFRESCHLÜSSEL AUF DER GRUNDLAGE DES OS-NUB-SCHLÜSSELS AUS
560
ERZEUGE EINEN APPLET-SCHLÜSSEL, INDEM DER OS-NUB-SCHLÜSSEL MIT EINEM APPLET-ID EINER HASH-OPERATION UNTERZOGEN WIRD
570
WÄHLE EINEN KLEINEREN APPLET-CHIFRESCHLÜSSEL AUF DER GRUNDLAGE DES APPLET-SCHLÜSSELS AUS
580
ENDE

FIG. 5