(19) Deutsches Patent- und Markenamt

(10) **DE 10 2005 030 590 B4** 2011.03.24

(12) # Patentschrift

(21) Aktenzeichen: **10 2005 030 590.3**
(22) Anmeldetag: **30.06.2005**
(43) Offenlegungstag: **04.01.2007**
(45) Veröffentlichungstag der Patenterteilung: **24.03.2011**

(51) Int Cl.[8]: ***H04L 9/30*** (2006.01)
H04L 9/32 (2006.01)
H04L 9/16 (2006.01)

(73) Patentinhaber:
Advanced Micro Devices, Inc., Sunnyvale, Calif., US

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80802 München

(72) Erfinder:
Wachtler, Axel, 01662 Meißen, DE; Findeisen, Ralf, 01127 Dresden, DE; Schuecke, Frank, 01129 Dresden, DE

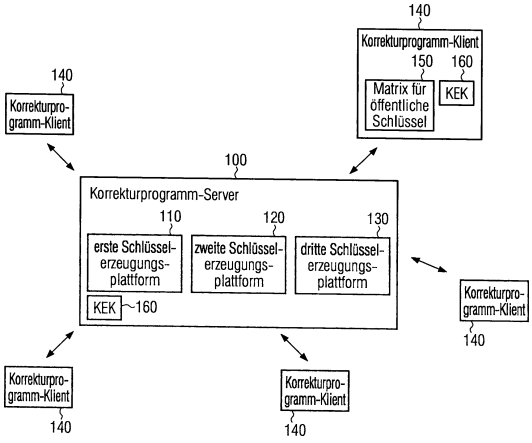
(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:

US 61 57 721 A
WO 02/25 409 A2

Menezes,A.J., van Oorschot,P.C., Vanstone, S.A.: Handbook of Applied Cryptography, CRC Press, 1996, S.321-383 und 543-555, ISBN: 0-8493-8523-7

(54) Bezeichnung: **Sicheres Patchsystem**

(57) Hauptanspruch: Patch-Server (100), der mit einem Patch-Client (140) verbunden ist, um dem Patch-Client einen Patch (1150, 1355, 1365, 1375, 1385) bereitzustellen, und Folgendes umfasst:
eine erste Schlüsselerzeugungsplattform (110), die zum Erzeugen einer ersten Gruppe privater Schlüssel (260), welche eine Vielzahl erster privater Schlüssel (200–215) umfasst, eingerichtet ist;
eine zweite Schlüsselerzeugungsplattform (120), die von der ersten Schlüsselerzeugungsplattform verschieden ist und zum Erzeugen einer zweiten Gruppe privater Schlüssel (270), welche eine Vielzahl zweiter privater Schlüssel (220–235) umfasst, eingerichtet ist;
eine erste Schlüsselauswahleinrichtung, die zum Auswählen (410, 610) eines der ersten privaten Schlüssel aus der ersten Gruppe privater Schlüssel eingerichtet ist;
eine zweite Schlüsselauswahleinrichtung, die zum Auswählen (410, 620) eines der zweiten privaten Schlüssel aus der zweiten Gruppe privater Schlüssel eingerichtet ist;
eine erste Signaturerzeugungseinrichtung, die zum Erzeugen (420, 710, 1010) einer ersten digitalen Signatur (1110, 1310) auf Basis des Patchs und des ersten ausgewählten privaten Schlüssels eingerichtet ist;...



Beschreibung

HINTERGRUND DER ERFINDUNG

1. Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft im Allgemeinen Softwareaktualisierungen und betrifft insbesondere die sichere Verteilung von Softwareaktualisierungen in einem verteilten System.

2. Beschreibung des Stands der Technik

[0002] Viele Computermediensysteme oder Kommunikationssysteme weisen Sicherheitslücken auf, die einen unautorisierten Datenzugriff oder das Einbringen von Computerwürmern ermöglichen. Dadurch kann ein beträchtlicher Schaden hervorgerufen werden. Für gewöhnlich werden derartige Sicherheitslücken mittels sicherheitsrelevanter Softwareaktualisierungen, die auch als Korrekturprogramme oder „Patch" bezeichnet werden, geschlossen. In verteilten Systemen können Korrekturprogramme von einem Korrekturprogrammserver erzeugt und dann an eine Anzahl von Korrekturprogrammklienten, beispielsweise Mobileinheiten eines Kommunikationssystems oder Verbrauchereinrichtungen, die eingebettete Prozessoren aufweisen, verteilt werden.

[0003] Jedoch müssen Korrekturprogramme, die von einem nicht sicheren System heruntergeladen werden, dennoch gegen schädigende Modifizierungen geschützt werden. Ansonsten könnte ein Virus oder ein Wurm weiterhin effektive Angriffe starten. Beispielsweise könnte ein Angriff mit einer Verweigerung von Dienstleistungen (DOS) gegenüber einem GSM (globales System für Mobilkommunikation) Netzwerk ausgeführt werden, wobei lediglich ein infiziertes aktives Gerät pro Funkzelle ausreicht, um das gesamte System zu blockieren.

[0004] In konventionellen Systemen wird häufig die Verschlüsselung mit öffentlichem Schlüssel (PKC), die auch als asymmetrische Verschlüsselung bezeichnet wird, angewendet, um eine Paketverteilung zu schützen, indem Senden geheimer Schlüssel über nicht sichere Netzwerke vermieden wird. Die grundliegende Idee besteht darin, dass es zwei Schlüssel gibt: einen öffentlichen Schlüssel (PK), der nur für die Verschlüsselung verfügbar und öffentlich bekannt ist, und einen privaten Schlüssel, der auch als geheimer Schlüssel (SK) bezeichnet wird, der bekannt sein muss, um Nachrichten zu entschlüsseln. Die Sicherheit beruht auf der Schwierigkeit des Herleitens des privaten Schlüssels aus dem öffentlichen Schlüssel und der Schwierigkeit in der Entschlüsselung einer verschlüsselten Nachricht, ohne dass der private Schlüssel bekannt ist.

[0005] Eine spezielle Anwendung des PKC ist die digitale Signatur. Hierbei wird eine Verschlüsselungsprüfsumme (Hashsumme) über einem gewissen Dokument berechnet und dann wird die Prüfsumme mit dem privaten Schlüssel des Anwenders verschlüsselt, um die digitale Signatur zu erzeugen. Die Signatur wird in einer gewissen Form dem Dokument angefügt. Jeder, der den öffentlichen Schlüssel des Anwenders kennt, kann die Prüfsumme des Dokuments berechnen, die angefügte Signatur mit dem öffentlichen Schlüssel entschlüsseln und das Ergebnis mit dem prüfsummenverschlüsselten Dokument vergleichen. Alternativ kann die digitale Signatur durch Verschlüsseln der Prüfsumme mit dem privaten Schlüssel des Anwenders erzeugt werden. Für die Verifizierung kann der Empfänger des Dokuments wiederum die Prüfsumme berechnen, diese mit dem öffentlichen Schlüssel entschlüsseln und das Ergebnis mit der bereitgestellten Signatur vergleichen.

[0006] Eine korrekte digitale Signatur zeigt, dass ihr Schöpfer den privaten Schlüssel (Authentizität) kannte und dass das Dokument seit der Schaffung der Signatur (Integrität) weder durch Hinzufügen, Löschen oder Modifizieren des Inhalts oder durch Umordnen von Teilen davon modifiziert wurde. Die zuletzt genannte Eigenschaft wird durch die Eigenschaften der verwendeten kryptographischen Prüfsummenfunktionen, beispielsweise MD5 (Nachrichtenverarbeitungsalgorithmus 5), SHA-1 (sicherer Prüfsummenalgorithmus 1) oder RIPEMD-160 (Race Integrity Primitives Evaluation Message-Verarbeitung 160) erreicht. Jedoch kann nicht erkannt werden, ob beispielsweise der private Schlüssel während des Erstellens der Signatur gestohlen war.

[0007] Die asymmetrische Verschlüsselung wird häufig für digitale Signaturen verwendet, da auch nicht vertrauenswürdige Parteien, d. h. ohne Kenntnis von Geheimnissen, in der Lage sind, die digitale Signatur zu prüfen. Jedoch weist PKC den Nachteil auf, dass es anfällig und langsam ist. Des weiteren gibt es Sicherheitslücken in den PKC-Systemen auf Grund möglicher „Mann-in-der-Mitte-"Angriffen. Dies kann verhindert werden, wenn die Authentizität der öffentlichen Schlüssel in irgendeiner Weise nachgewiesen werden kann. In einigen Szenarien reicht es, eine kryptographische Prüfsumme über den öffentlichen Schlüssel auszurechnen, d. h. der sogenannte Fingerabdruck, um diesen dem Empfänger direkt, beispielsweise über Telefon, mitzuteilen. In komplexen und dynamisch sich ändernden Umgebungen ist dies jedoch nicht möglich. Daher werden Infrastrukturen mit öffentlichen Schlüsseln (PKI) verwendet, wo öffentliche Schlüssel digital mittels einer Hierarchie vertrauenswürdiger Parteien unterzeichnet werden. Die Einrichtung und Unterhaltung von PKI-Infrastrukturen ist jedoch kostenaufwendig.

[0008] Des weiteren gibt es eine Reihe weiterer Risiken und Probleme, die sich aus dem PKC ergeben, wenn beispielsweise eine RSA (Rivest-Shamir-Adleman) Verschlüsselung verwendet wird. Zur geringen Geschwindigkeit des PKC kommt hinzu, dass Klartextbereiche mit zufälligen Bits aufgefüllt werden, da ansonsten diverse Angriffe möglich sind. Daher werden Klartexte für gewöhnlich nicht direkt mit öffentlichen Schlüsseln verschlüsselt. Stattdessen wird ein Zufallssitzungsschlüssel erzeugt und für die herkömmliche symmetrische Verschlüsselung, beispielsweise unter Verwendung von Algorithmen, wie AES (fortschrittlicher Verschlüsselungsstandard) oder Two-fish, verwendet. In diesen Protokollen, die als Hybridprotokolle bezeichnet werden, wird lediglich der Sitzungsschlüssel mit dem öffentlichen Schlüssel verschlüsselt und dem Schlüsseltext hinzugefügt.

[0009] Selbst wenn jedoch ein Hybridprotokoll verwendet wird, muss der Sitzungsschlüssel in zufälliger Weise aufgefüllt werden zu beispielsweise 512 oder 1024 Bit-Längen. Eine nicht-ausreichende Auffüllung führt zu einer reduzierten Sicherheit.

[0010] Wenn ferner ein Bit in einem privaten RSA-Schlüssel durch Hardware oder durch einen Angreifer umgedreht wird, und dieser private Schlüssel dann zum Definieren einer Nachricht verwendet wird, kann der öffentliche Schlüssel faktorisiert werden und damit kann die Sicherheit vollständig aufgehoben werden. Diese führt zu erheblichen Risiken bei PKC-Systemen.

[0011] Ferner können Fallen bei der Erzeugung öffentlicher Schlüssel eingebaut sein. Ein Angreifer, der die Modifizierungen des Erzeugungsalgorithmus kennt, ist in der Lage, den privaten Schlüssel leicht aus dem öffentlichen Schlüssel herzuleiten, so dass wiederum die Sicherheit vollständig aufgehoben ist.

[0012] Daher versuchen viele konventionelle Korrektursysteme, PKC zu vermeiden. In Systemen, in denen es einen Dialog zwischen vertrauenswürdigen Dienstleistungen und Klienten, etwa eingebetteten Prozessoren (beispielsweise in GSM-Telefonen), gibt, würde ein seriell gestalteter Schlüssel (möglicherweise in einer Smart-Karte) und ein Frage-Antwort-Protokoll eine einfache und zuverlässige Lösung bieten. Jedoch ist dies in vielen Systemen nicht anwendbar. Wenn beispielsweise der Empfänger während er Initialisierungszeit fixiert und passiv ist, ist eine Serialisierung nicht möglich.

[0013] Alternativ werden in konventionellen Systemen, in denen die Parteien gemeinsame Geheimnisse teilen, kryptographische Prüfsummen, etwa HMAC (verschlüsselte Prüfsummenbildung für Nachrichtenauthentizierung) verwendet. Ein Beispiel dafür ist die Authentizierung und die Schlüsselhandhabung in GSM/UMTS (globales System für Mobilkommunikation/universelles Mobiltelekommunikationssystem) Mobiltelefonen, indem die Schlüssel durch SIM (Teilnehmeridentifikationsmodul) Karten verteilt werden. Diese Lösung ist einfacher, schneller und zuverlässiger als PKC. Obwohl jedoch eine HMAC-kryptographische Prüfsumme die Integrität beweisen könnte, wäre der entsprechende geheime Schlüssel in der Firmware des EP-(eingebetteten Prozessors)Empfänger festgelegt. Ein Bekanntwerden dieses geheimen HMAC-Schlüssels würde daher alle Prüfsummen als wertlos erweisen.

[0014] Daher werden in vielen konventionellen Korrektursystemen digitale Signaturen verwendet. Jedoch ist häufig das Sicherstellen der Integrität nicht ausreichend. Wenn Korrekturprogramme untersucht werden, können dennoch Sicherheitslücken erkannt werden. Ein signiertes Korrekturprogramm beweist lediglich seine Echtheit, aber nicht seine Sicherheit. Die Sicherheit ist nur durch Vertrauen des Autors gewährleistet. Ferner weist das Signieren von Korrekturprogrammen unter Anwendung des gleichen öffentlichen Schlüssels für alle Korrekturprogrammklienten den Nachteil auf, dass dies äußerst unsicher ist. Wenn der private Schlüssel bekannt wird, ist die gesamte Sicherheit aufgehoben. Ein noch wahrscheinlicheres Szenario ist eines, in welchem der private Schlüssel verloren geht. In diesem Falle könnten keine weiteren Korrekturprogramme herausgegeben werden und alle Korrekturprogrammklienten würden zunehmend unsicher werden.

[0015] Aus der WO 02/25409 A2 sind ein Codesignierungssystem und -verfahren insbesondere bei Java-Anwendungen für mobile Kommunikationsgeräte bekannt. Es wird offenbart, dass eine Softwareanwendung mehrere gerätespezifische, bibliothekspezifische oder API-spezifische Signaturen oder eine Kombination solcher Signaturen als Anhang haben kann.

[0016] Die US 6,157,721 beschreibt Systeme und Verfahren, die kryptographische Techniken verwenden, um sichere Rechnerumgebungen zu schützen.

[0017] A. J. Menezes et al., ”Handbook of Applied Cryptography”, CRC Press, 1996, Seiten 321 bis 383 und 543 bis 555 befasst sich mit Hash-Funktionen und Datenintegrität sowie Techniken zum Verwalten von kryptographischen Schlüsseln.

ÜBERBLICK ÜBER DIE ERFINDUNG

[0018] Der Erfindung liegt die Aufgabe zugrunde, verbesserte Korrekturprogrammserver, Korrekturprogrammklienten und entsprechende Verfahren bereitzustellen, die die Nachteile im Stand der Technik überwinden. Entsprechende Ausführungsformen können den Geheimnisschutz verbessern und die To-

leranz für einen Schlüsselverlust erhöhen. Dies kann wiederum einen erhöhten Investitionsschutz bedeuten. Ferner kann der Schutz gegenüber einer Schlüsseloffenlegung verbessert werden. Andere Ausführungsformen können das Risiko schwacher Schlüssel in solchen Fällen reduzieren, in denen die Schlüsselerzeugung noch Sicherheitslücken aufweist.

[0019] Diese Aufgabe wird durch die Gegenstände der unabhängigen Patentansprüche gelöst.

[0020] Bevorzugte Ausgestaltungen sind in den Unteransprüchen angegeben.

[0021] Gemäß einer Ausführungsform ist ein Korrekturprogramm-Anbieter bzw. Server mit einem Korrekturprogramm-Klienten zur Bereitstellung eines Korrekturprogramms für den Klienten verbunden. Der Korrekturprogramm-Server umfasst eine erste Schlüsselerzeugungsplattform, eine zweite Schlüsselerzeugungsplattform, die sich von der ersten Schlüsselerzeugungsplattform unterscheidet, eine erste Schlüsselauswahleinrichtung, eine zweite Schlüsselauswahleinrichtung, einen ersten Signaturerzeuger, einen zweiten Signaturerzeuger und einen Sender. Die erste Schlüsselerzeugungsplattform ist ausgebildet, eine erste private Schlüsselgruppe mit mehreren ersten privaten Schlüsseln zu erzeugen. Die zweite Schlüsselerzeugungsplattform ist ausgebildet, eine zweite private Schlüsselgruppe mit mehreren zweiten privaten Schlüsseln zu erzeugen. Die erste und die zweite Schlüsselauswahleinrichtungen sind ausgebildet, einen der ersten und zweiten privaten Schlüsseln aus den ersten und zweiten privaten Schlüsselgruppen auszuwählen. Der erste Signaturerzeuger ist ausgebildet, eine erste digitale Signatur auf der Grundlage des Korrekturprogramms und des ersten ausgewählten privaten Schlüssels zu erzeugen. Der zweite Signaturerzeuger ist ausgebildet, eine zweite digitale Signatur auf der Grundlage des Korrekturprogramms und der zweiten ausgewählten privaten Schlüssels zu erzeugen. Der Sender ist ausgebildet, das Korrekturprogramm zusammen mit der ersten und der zweiten digitalen Signatur zu dem Korrekturprogramm-Klienten zu senden.

[0022] Gemäß einer weiteren Ausführungsform wird ein Verfahren zum Zuführen eines Korrekturprogramms zu einem Korrekturprogramm-Klienten bereitgestellt. Es wird eine erste private Schlüsselgruppe, die mehrere erste private Schlüssel enthält, unter Anwendung einer ersten Schlüsselerzeugungsplattform erzeugt. Es wird eine zweite private Schlüsselgruppe, die mehrere zweite private Schlüssel enthält, unter Anwendung einer zweiten Schlüsselerzeugungsplattform, die sich von der ersten Schlüsselerzeugungsplattform unterscheidet, erzeugt. Einer der ersten privaten Schlüssel wird aus der ersten privaten Schlüsselgruppe ausgewählt, und einer der zweiten privaten Schlüsseln wird aus der zweiten privaten Schlüsselgruppe ausgewählt. Es wird eine erste digitale Signatur auf der Grundlage des Korrekturprogramms und des ersten ausgewählten privaten Schlüssels erzeugt. Es wird eine zweite digitale Signatur auf der Grundlage des Korrekturprogramms und des zweiten ausgewählten privaten Schlüssels erzeugt. Das Korrekturprogramm wird zusammen mit der ersten und der zweiten digitalen Signatur zu dem Korrekturprogramm-Klienten gesendet.

[0023] Eine weitere Ausführungsform betrifft einen Korrekturprogramm-Klienten, der mit einem Korrekturprogramm-Server verbunden ist, um ein Korrekturprogramm von dem Korrekturprogramm-Server zu empfangen. Der Korrekturprogramm-Klient umfasst eine erste und eine zweite Speichereinrichtung, eine erste und eine zweite Schlüsselauswahleinrichtung und eine erste und eine zweite Signaturverifizierungskomponente. Die erste Speichereinrichtung speichert eine erste öffentliche Schlüsselgruppe, die mehrere erste öffentliche Schlüssel enthält, die durch einer erste Schlüsselerzeugungsplattform erzeugt sind. Die zweite Speichereinrichtung speichert eine zweite öffentliche Schlüsselgruppe, die mehrere zweite öffentliche Schlüssel enthält, die durch eine zweite Schlüsselerzeugungsplattform erzeugt sind, die sich von der ersten Schlüsselerzeugungsplattform unterscheidet. Die erste und die zweite Schlüsselauswahleinrichtung sind ausgebildet, einen der ersten und zweiten öffentlichen Schlüssel aus der ersten und der zweiten öffentlichen Schlüsselgruppe auszuwählen. Die erste Signaturverifizierungskomponente ist ausgebildet, eine erste digitale Signatur, die von dem Korrekturprogramm-Server zusammen mit dem Korrekturprogramm erhalten wird, unter Anwendung des ersten ausgewählten öffentlichen Schlüssels zu verifizieren. Die zweite Signaturverifizierungskomponente ist ausgebildet, eine zweite digitale Signatur, die von dem Korrekturprogramm-Server zusammen mit dem Korrekturprogramm empfangen wird, unter Verwendung des zweiten ausgewählten öffentlichen Schlüssels zu verifizieren. Der Korrekturprogramm-Klient ist ausgebildet, das Korrekturprogramm nur zu installieren, wenn die Ergebnisse des Verifizierens der ersten und der zweiten digitalen Signatur die Authentizität und Integrität der ersten und der zweiten digitalen Signatur anzeigen.

[0024] Gemäß einer noch weiteren Ausführungsform wird ein Verfahren zum Installieren eines Korrekturprogramms in einem Korrekturprogramm-Klienten bereitgestellt. Der Korrekturprogramm wird zusammen mit einer ersten und einer zweiten digitalen Signatur von einem Korrekturprogramm-Server, der mit dem Korrekturprogramm-Klienten verbunden ist, empfangen. Eine erste öffentliche Schlüsselgruppe mit mehreren ersten öffentlichen Schlüsseln wird in dem Korrekturprogramm-Klienten gespeichert, wobei die ersten öffentlichen Schlüssel durch eine erste Schlüsselerzeugungsplattform erzeugt sind. Fer-

ner wird eine zweite öffentliche Schlüsselgruppe mit mehreren zweiten öffentlichen Schlüsseln in dem Korrekturprogramm-Klienten gespeichert, wobei die zweiten öffentlichen Schlüssel mittels einer zweiten Schlüsselerzeugungsplattform, die sich von der ersten Schlüsselerzeugungsplattform unterscheidet, erzeugt sind. Einer der ersten öffentlichen Schlüssel und einer der zweiten öffentlichen Schlüssel wird aus der ersten bzw. der zweiten Schlüsselgruppe ausgewählt. Die erste digitale Signatur wird unter Anwendung des ersten ausgewählten öffentlichen Schlüssels verifiziert, und die zweite digitale Signatur wird unter Anwendung des zweiten ausgewählten öffentlichen Schlüssels verifiziert. Der Korrekturprogramm wird in dem Korrekturprogramm-Klienten nur dann installiert, wenn die Ergebnisse des Verifizierens der ersten und der zweiten digitalen Signatur die Authentizität und Integrität der ersten und der zweiten digitalen Signatur anzeigen.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0025] Die begleitenden Zeichnungen sind hierin mit eingeschlossen und bilden einen Teil der Beschreibung, um die Prinzipien der Erfindung zu erläutern. Die Zeichnungen sollen die Erfindung nicht auf die dargestellten und beschriebenen Beispiele einschränken, wie die Erfindung ausgeführt und angewendet werden kann. Weitere Merkmale und Vorteile gehen aus der folgenden detaillierteren Beschreibung der Erfindung hervor, wie sie auch in den begleitenden Zeichnung dargestellt ist, wobei:

[0026] Fig. 1 eine Blockansicht ist, die die Komponenten eines Korrekturprogramm-Systems gemäß einer Ausführungsform zeigen;

[0027] Fig. 2 eine Blockansicht ist, die eine Verwaltung eines geheimen Schlüssels gemäß einer Ausführungsform zeigt;

[0028] Fig. 3 eine Blockansicht ist, die die Verwaltung eines öffentlichen Schlüssels gemäß einer Ausführungsform zeigt;

[0029] Fig. 4 ein Flussdiagramm ist, das eine Korrekturprogramm-Übertragung gemäß einer Ausführungsform zeigt;

[0030] Fig. 5 die Schritte einer Prüfsummenkette gemäß einer Ausführungsform zeigt;

[0031] Fig. 6 ein Flussdiagramm ist, das die Auswahl des privaten Schlüssels gemäß einer Ausführungsform zeigt;

[0032] Fig. 7 ein Flussdiagramm ist, das die Erzeugung der Signatur gemäß einer Ausführungsform zeigt;

[0033] Fig. 8 die Schritte der KEK-Verschlüsselung gemäß einer Ausführungsform zeigt;

[0034] Fig. 9 ein Flussdiagramm ist, das die Schritte einer Prüfsummenkette gemäß einer weiteren Ausführungsform zeigt;

[0035] Fig. 10 ein Flussdiagramm ist, das die Erzeugung der Signatur gemäß einer weiteren Ausführungsform zeigt;

[0036] Fig. 11 die Zusammensetzung eines Korrekturprogramm-Blockes gemäß einer Ausführungsform zeigt;

[0037] Fig. 12 eine Blockansicht ist, die die Konfiguration eines Schlüsselindikators gemäß einer Ausführungsform zeigt;

[0038] Fig. 13 eine Blockansicht ist, die die Anordnung eines Korrekturprogramm-Blocks gemäß einer weiteren Ausführungsform zeigt;

[0039] Fig. 14 eine Blockansicht ist, die einen Übertragungsblock gemäß einer Ausführungsform zeigt;

[0040] Fig. 15 ein Flussdiagramm ist, das einen Korrekturprogramm-Installationsprozess gemäß einer Ausführungsform zeigt;

[0041] Fig. 16 ein Flussdiagramm ist, das die Auswahl eines öffentlichen Schlüssels gemäß einer Ausführungsform zeigt;

[0042] Fig. 17 ein Flussdiagramm ist, dass die Signaturverifizierung gemäß einer Ausführungsform zeigt;

[0043] Fig. 18 ein Flussdiagramm ist, das die Korrekturprogramm-Installation gemäß einer weiteren Ausführungsform zeigt;

[0044] Fig. 19 die Signaturverifizierung gemäß der weiteren Ausführungsform zeigt;

[0045] Fig. 20 ein Flussdiagramm ist, das die blockweise Korrekturprogramm-Entschlüsselung gemäß der weiteren Ausführungsform zeigt.

DETAILLIERTE BESCHREIBUNG DER ERFINDUNG

[0046] Die anschaulichen Ausführungsformen der vorliegenden Erfindung werden nunmehr mit Bezugnahme zu den Zeichnungen beschrieben. Die Software eines Korrekturprogramm-Klienten, beispielsweise eines eingebetteten Prozessors eines Geräts, kann Korrekturprogramme von einem nicht sicheren System beziehen. Die Ausführungsformen stellen sicher, dass diese Korrekturprogramme unmodifiziert

sind, wenn sie angewendet werden. Beispielsweise könnten bösartige Korrekturprogramme aus Viren oder Würmern resultieren. Es kann auch möglich sein, dass unentdeckte Sicherheitslücken in der Software oder in den Korrekturprogrammen selbst die gleiche Wirkung aufweisen. Die Ausführungsformen können den Korrekturprogramm-Klienten vor den negativen Auswirkungen derartiger Szenarien schützen.

[0047] In Fig. 1 sind Komponenten eines Korrekturprogramm-Systems gemäß einer Ausführungsform gezeigt. Ein Korrekturprogramm-Server **100** ist mit mehreren Korrekturprogramm-Klienten **140** verbunden, um die Korrekturprogramm-Klienten mit sicherheitsbezogenen Softwareaktualisierungen, d. h. Korrekturprogramm zu versorgen. Die Korrekturprogramm-Klienten können beispielsweise eingebettete Systeme, Personalcomputer oder Medien/Kommunikationsgeräte sein. Diese können mit dem Korrekturprogramm-Server **100** über eine beliebige Art einer geeigneten Verbindung, beispielsweise drahtlose oder verdrahtete Verbindungen verbunden sein. Der Korrekturprogramm-Server **100** kann ein Computer oder ein verteiltes Computersystem sein.

[0048] Gemäß der dargestellten Ausführungsform umfasst der Korrekturprogramm-Server **100** drei Schlüsselerzeugungsplattformen **110**, **120**, **130**. Die Plattformen **110**, **120**, **130** können getrennt voneinander sein und können sowohl der Schlüsselerzeugung als auch der Handhabung dienen. Im Allgemeinen beschreibt eine Plattform eine gewisse Art einer Plattform, die in Hardware oder Software oder beidem ausgeführt sein kann, und die es ermöglicht, eine Software zu betreiben. Die Schlüsselerzeugungsplattformen **110**, **120**, **130** der vorliegenden Ausführungsform basieren auf unterschiedlichen Anwendungen und es kann Hardware parallel verwendet werden. Dies kann die Sicherheit im Hinblick auf Fallen verbessern, die in den Schlüsselerzeugungseinrichtungen implementiert sein können. In Ausführungsformen, in denen HSM (Hardwaresicherheitsmodule) als die einzige Basis akzeptierbar sind, kann die Verwendung unterschiedlicher Software und Hardware für die Schlüsselerzeugung jedoch unnötig sein.

[0049] Gemäß der Ausführungsform erzeugt die erste Schlüsselerzeugungsplattform **110** Schlüssel mit der Hilfe einer gewissen – nAbschirm(nShield)-HSM aus nVerschlüsselung (nCipher) und speichert diese Schlüssel. Die zweite Schlüsselerzeugungsplattform **120** kann Schlüssel erzeugen und verwenden unter Knoppix Linux. Da Knoppix Linux nur im RAM (Speicher mit wahlfreiem Zugriff) operiert, bleiben keine Spuren auf einer Festplatte zurück. Die erzeugten Schlüssel können extern in verschlüsselter Form gespeichert werden. Die Schlüsselhandhabung in der zweiten Schlüsselerzeugungsplattform **120** kann durch OpenSSL erreicht werden. Ferner kann eine lange Kennwortphrase so geteilt werden, dass diese von zwei unterschiedlichen Personen nacheinander geschrieben werden muss, um private Schlüssel für die Anwendung freizugeben. Die dritte Schlüsselerzeugungsplattform **130** kann Schlüssel mittels OpenSSL traditionell aber unter einem SELinux (sicherheitsverbesserten Linux) System oder einem beliebigen gleichwertigen Hochsicherheitsbetriebssystem anwenden. Alternativ können Verschlüsselungskarten in der dritten Schlüsselerzeugungsplattform **130** verwendet sein.

[0050] Um die Zuverlässigkeit des Verhinderns des Bekanntwerdens bekannter Schlüssel zu erhöhen, kann eine Geheimnisteilung („k aus n Operatorkarten") in der n-Abschirmung verwendet werden. Es können unabhängige Parteien in den Signierungsprozess eingeschlossen werden und Schlüssel können zwischen zwei Administratoren oder Gruppen aus Administratoren der ersten Schlüsselerzeugungsplattform **110** aufgeteilt werden.

[0051] Das Korrekturprogramm-System der vorliegenden Ausführungsform signiert Korrekturprogramme mit mehreren Signaturen unter Verwendung variabler Teilmengen aus geheimen Schlüsseln, die von den Schlüsselerzeugungsplattformen **110**, **120**, **130** erzeugt und verwaltet werden. Die entsprechenden öffentlichen Schlüssel können auch durch die Schlüsselerzeugungsplattformen **110**, **120**, **130** jeweils erzeugt werden und in die Korrekturprogramm-Klienten **140** während ihrer Erzeugung eingespeist werden. Die öffentlichen Schlüssel können in den Korrekturprogramm-Klienten **140** in einer Matrix für öffentliche Schlüssel **150** abgelegt werden.

[0052] Vor dem Senden zu den Korrekturprogramm-Klienten **140** können die Korrekturprogrammes symmetrisch unter Anwendung des Zufallssitzungsschlüssels verschlüsselt werden. Der Zufallssitzungsschlüssel kann wiederum unter Anwendung eines geheimen Hauptschlüssels verschlüsselt werden, der gemeinsam für alle Korrekturprogramm-Klienten **140** verwendet werden kann, und der auch als der Schlüsselverschlüsselungsschlüssel (KEK) **160** bezeichnet wird. Dies führt zu erhöhter Geschwindigkeit und Einfachheit im Hinblick auf konventionelle PKC-Systeme. Des weiteren kann damit ein Schutz gegen unbekannte Schwachstellen des Verschlüsselungsalgorithmus erhöht werden. Diese Schwachstellen können von einem Angreifer nicht so leicht ausgenutzt werden, da keiner der Bereiche des Klartexts mit dem gleichen Schlüssel verschlüsselt ist. Der KEK-Schlüssel **160** kann in sicherer Weise von dem Korrekturprogramm-Server **100** gespeichert werden.

[0053] In dem Korrekturprogramm-Klienten **140** kann der KEK-Schlüssel **160** in versteckter Form zum Zwecke der höheren Sicherheit gespeichert sein. Da-

zu können Hardware und Softwaremittel kombiniert werden. Beispielsweise kann ein 128-Bit-Schlüssel mittels einer XOR-(exklusiv und/oder)Funktion aus mehreren 128 Bit-Bereichen (geheime Aufteilung) gebildet werden. Die Daten können über das Programm verteilt werden und sind schwer aufzufinden. Ferner kann die Verwendung von Funktionszeigern, die über eine „verrückte" Berechnung dynamisch zugeordnet sind, und die beispielsweise durch Makros an diversen sehr unterschiedlichen Stellen in dem Programm durchgeführt werden, eine Gegenmaßnahme gegenüber einer nachträglichen Untersuchung sowie gegenüber einer außer Kraft setzenden Verwendung von Debuggern bereitstellen. Eine derartige geheime Aufteilung kann in einer Weise angewendet werden, in der praktisch niemand den Schlüssel kennt, lediglich ein Programm während der Laufzeit kann diesen Schlüssel temporär nachbilden.

[0054] Sowohl die Matrix für den öffentlichen Schlüssel **150** als auch der Schlüsselverschlüsselungsschlüssel **160** können in den Korrekturprogramm-Klienten **140** durch den Anbieter eingesteckt werden, bevor der entsprechende Korrekturprogramm-Klient **140** an einen Anwender verkauft wird. Die entsprechenden geheimen Schlüssel können beim Anbieter in sicheren Plätzen verbleiben. Wie in der vorliegenden Ausführungsform sind die öffentlichen Schlüssel festgelegt, d. h. eine Änderung der privaten Schlüssel ist nicht möglich – lediglich ein Aufrufen ist möglich. Somit können die privaten Schlüssel in dem Korrekturprogramm-Server **100** geweckt werden.

[0055] Jeder Korrekturprogramm-Klient **100** kann einen nicht zurücksetzbaren Zähler aufweisen, der eine Sequenzzahl des letzten empfangenen Korrekturprogramms speichert, um Angriffe durch erneutes Ablaufen zu vermeiden. Dies kann dafür sorgen, zu vermeiden wird, dass ältere Korrekturprogramme mit bekannten Sicherheitslücken angewendet werden. Dazu können beispielsweise die Korrekturprogramm-Klienten **140** eine Zeitmarkierung, die zusammen mit einem Korrekturprogramm empfangen wird, gegenüber einer funkempfangen Zeit überprüfen.

[0056] Jedes Korrekturprogramm kann eine Reihe von Korrekturprogramm-Blöcken aufweisen. Um Angriffe durch Anwenden des Überlaufs zu vermeiden, kann die Anzahl und die Größe der Korrekturprogramm-Blöcke durch Software begrenzt werden. Die Korrekturprogrammblöcke können gegebenenfalls nicht aktiviert werden, solange das Aktualisieren des Korrekturprogramms durchgeführt wird. Daher ist es unter Umständen nicht notwendig, Prüfsummen einzelner Korrekturprogramm-Blöcke zu berechnen.

[0057] Gemäß der vorliegenden Ausführungsform werden 12 Schlüsselpaare verwendet, um die Korrekturprogramme zu signieren. Jedes Korrekturprogramm kann drei Signaturen enthalten, wobei jede Signatur auf einem Schlüssel aus einer Gruppe von vier basiert, wie dies nachfolgend detaillierter beschrieben ist. Das zugrunde liegende Prinzip ist in den Fig. 2 und Fig. 3 gezeigt.

[0058] Fig. 2 zeigt einen Satz aus privaten Schlüsseln **200** bis **255**, die von dem Korrekturprogramm-Server **100** erzeugt und von diesen verwaltet werden. Eine erste private Schlüsselgruppe **260** kann vier private Schlüssel **200** bis **215** aufweisen. In ähnlicher Weise kann eine zweite private Schlüsselgruppe **270** vier andere private Schlüssel **220** bis **235** aufweisen und eine dritte private Schlüsselgruppe **280** kann vier weitere private Schlüssel **240** bis **250** enthalten. Die erste, die zweite und dritte private Schlüsselgruppe **260**, **270**, **280** können entsprechend durch die erste, zweite und dritte Schlüsselerzeugungsplattform **110**, **120**, **130** erzeugt und gehandhabt werden. Gemäß einer Ausführungsform können die privaten Schlüsselgruppen **260**, **270**, **280** unterschiedliche Vertrauensebenen besitzen.

[0059] In den Schlüsselerzeugungsplattformen **110**, **120**, **130** können HSMs verwendet werden, um eine nicht legale Herauslösung privater Schlüssel **200** bis **255** zu verhindern. Alternativ können „gemischte Schlüssel", beispielsweise auf der Grundlage von HSM und Knoppix/OpenSSL verwendet werden, die den Vorteil der sicheren Speicherung sowie des vollen Vertrauensschutzes bereitstellen. Ferner kann dies eine Steuerung der Schlüsselerzeugung und Wiedergewinnung der Schlüssel ermöglichen, ohne dass eine weitere HSM-Einrichtung erforderlich ist. Die Zugriffsrechte zu den geheimen Schlüssel **200** bis **255** können unter den nicht kooperierenden Gruppen aufgeteilt sein. Ferner kann das Prinzip der geheimen Aufteilung („3 aus 5 Operatorkarten" in dem Falle von HSMs) angewendet werden.

[0060] Die entsprechenden öffentlichen Schlüssel können in den Korrekturprogramm-Klienten **140** in der Matrix **150** gespeichert werden, wie dies detaillierter in Fig. 3 gezeigt ist.

[0061] Eine erste öffentliche Schlüsselgruppe **360** kann vier öffentliche Schlüssel **300** bis **315** halten, die den privaten Schlüsseln **200** bis **215** der ersten privaten Schlüsselgruppe **260** entsprechen. Die erste öffentliche Schlüsselgruppe **360** kann durch die erste Schlüsselerzeugungsplattform **110** erzeugt sein und in den Korrekturprogramm-Klienten **140** während der Herstellung eingespeist werden. Eine zweite öffentliche Schlüsselgruppe **370**, die durch die zweite Schlüsselerzeugungsplattform **120** erzeugt sein kann und in den Korrekturprogramm-Klienten **140** durch den Vertreiber eingespeist sein kann, kann vier öffentliche Schlüssel **320** bis **335** aufweisen, die den privaten Schlüsseln der zweiten privaten Schlüsselgruppe **270** entsprechen. Schließlich kann eine dritte öffentliche Schlüsselgruppe **380** aus vier öffentli-

chen Schlüsseln **340** bis **355** aufgebaut sein, die den vier privaten Schlüsseln **240** bis **255** der dritten privaten Schlüsselgruppe **280** entsprechen. Die dritte öffentliche Schlüsselgruppe **380** kann durch die dritte Schlüsselerzeugungsplattform **380** erzeugt sein und in den Korrekturprogramm-Klienten **140** vor dem Verkauf gespeichert sein.

[0062] Die privaten Schlüsseln **200** bis **255** können in einem standardisierten Format erzeugt sein. Dies kann für eine Schlüsselsicherung und eine Wiedergewinnung bei Ausfall ausgenutzt werden. Ferner kann die weitere Schlüsselerzeugung unabhängig von einem beliebigen Sicherheitsserver sein. In Ausführungsformen, in denen eine RSA-Verschlüsselung angewendet wird, können die öffentlichen Schlüssel **300** bis **355** direkt als C-Kopfzeilen mittels Modul (d. h. Produkt p × q geheimen Primzahlen) und einem öffentlichen Exponenten gehandhabt werden. Die Signaturen können in PKCS#1 (öffentlicher Schlüsselentschlüsselungsstandard Nr. 1) Format erzeugt werden, das von RSA CryptoCME und BSAFE-Bibliotheken sowie von OpenSSL, das mit HSM-Einrichtungen kooperiert, unterstützt wird.

[0063] In einer Ausführungsform können die öffentlichen Schlüssel **300** bis **355** 1024 Bit lang sein. In anderen Ausführungsformen können jedoch andere Schlüssellängen verwendet werden. Beispielsweise können 512-Bit RSA- oder DAS (digitaler Signaturalgorithmus) Schlüssel verwendet werden, um Speicherplatz und Rechenzeit bei der Signaturprüfung einzusparen. Im Gegensatz zu den KEK-Schlüssel **160** müssen die öffentlichen Schlüssel **300** bis **355** nicht in den Korrekturprogramm-Klienten **140** verborgen sein. Gemäß der vorliegenden Ausführungsform ist der KEK-Schlüssel **160** ein Hauptschlüssel für alle Korrekturprogramm-Klienten **140** und besitzt eine Bit-Länge von 128 Bits. Beispielsweise kann der KEK-Schlüssel **160** ein 128-Bit-AES-Schlüssel sein. Alternativ kann ein 128 Bit-Twofish-Schlüssel oder ein 256-Bit-AES-Schlüssel oder ein beliebiger anderer geeigneter Schlüssel für den KEK-Schlüssel **160** verwendet werden.

[0064] Zu beachten ist, dass die Verwendung von 12 Schlüsselpaaren zur Signierung der Korrekturprogramme lediglich ein spezielles Beispiel ist. Alternativ können weniger oder mehr Schlüsselgruppen eingesetzt werden (der Korrekturprogramm-Server **100** enthält dann weniger oder mehr Schlüsselerzeugungsplattformen), und jede öffentliche/private Schlüsselgruppe kann weniger oder mehr als vier öffentliche/private Schlüssel aufweisen. Des weiteren wurde die Anordnung der Schlüssel in einer Matrix lediglich zu Anschauungszwecken gewählt. Es können diverse andere Formen zum Speichern der öffentlichen und privaten Schlüssel verwendet werden. insbesondere können die drei privaten Schlüsselgruppen **260**, **270**, **280** separat gespeichert und gehandhabt werden, wobei die drei Schlüsselerzeugungsplattformen **110**, **120**, **130** eingesetzt werden. Aus Sicherheitsgründen könne sogar die einzelnen privaten Schlüssel **220** bis **255** jeder privaten Schlüsselgruppe **260**, **270**, **280** separat gespeichert werden.

[0065] Mit Bezug zu den Fig. 4 bis Fig. 8, Fig. 11, Fig. 12 und Fig. 14 wird die Funktion des Korrekturprogramm-Servers **100** gemäß einer ersten Ausführungsform beschrieben. In dieser Ausführungsform sind die Korrekturprogramm-Klienten in der Lage, die von dem Korrekturprogramm-Server **100** bereitgestellten Signaturen zusammen mit dem Korrekturprogramm nach dem Lesen des gesamten Korrekturprogramms zu verifizieren.

[0066] Fig. 4 zeigt die Gesamtfunktion des Korrekturprogramm-Servers **100** gemäß der Ausführungsform. Im Schritt **400** wird eine Prüfsummenkette ausgeführt. Die Prüfsummenkette **400** kann vier Schritte aufweisen, wie dies in Fig. 5 gezeigt ist.

[0067] Zunächst kann eine grundlegende Prüfsumme H berechnet werden, indem Korrekturprogramm im Schritt **510** der Prüfung unterzogen wird. Danach kann eine erste Prüfsumme H_1 berechnet werden, indem eine Verkettung (H|"0") der grundlegenden Prüfsumme H mit 1 Byte mit dem Wert 0 im Schritt **520** als Kontrolle summiert wird (hierbei bedeutet „|" eine Verkettung und „0" ist das Byte mit dem Wert 0). Anschließend wird im Schritt **530** eine zweite Prüfsumme H_2 berechnet, indem eine Verkettung (H_1|"1") der ersten Prüfsumme H_1 die im Schritt **520** berechnet wurde, mit einem Byte „1" mit dem Wert 1 ausgeführt wird. Schließlich wird eine dritte Prüfsumme H_3 berechnet, indem eine Verkettung (H_2|"2") der zweiten Prüfsumme H_2, die sich aus dem Schritt **530** ergibt, mit einem Byte „2" mit dem Wert 2 zur Kontrolle summiert wird (Schritt **540**).

[0068] Anzumerken ist, dass gemäß der vorliegenden Ausführungsform jedes Korrekturprogramm drei Signaturen enthält, die im Schritt **420** (siehe im Folgenden) auf der Grundlage der drei Prüfsummen H_1, H_2 und H_3 berechnet werden. In anderen Ausführungsformen können die Korrekturprogramme oder mehr Signaturen enthalten. In diesen Ausführungsformen kann die Prüfsummenkette **400** entsprechend kürzer oder länger sein.

[0069] Gemäß Fig. 4 können nunmehr private Schlüssel im Schritt **410** ausgewählt werden, um die Signaturen im Schritt **420** zu erzeugen. Die einzelnen Schritte der Auswahl **410** der privaten Schlüssel gemäß der Ausführungsform sind in Fig. 6 gezeigt.

[0070] Es kann ein erster privater Schlüssel aus der ersten privaten Schlüsselgruppe **260** im Schritt **610** ausgewählt werden. Anschließend wird ein zweiter privater Schlüssel aus der zweiten privaten Schlüs-

selgruppe **270** im Schritt **620** ausgewählt. In ähnlicher Weise wird ein dritter privater Schlüssel aus der dritten privaten Schlüsselgruppe **630** ausgewählt.

[0071] Selbstverständlich ist die dargestellte Sequenz aus Schritten nur anschaulich. Selbstverständlich können die privaten Schlüssel in einer beliebig anderen Reihenfolge ausgewählt werden. Alternativ können einige oder alle private Schlüssel ausgewählt werden, bevor die Prüfsummenkette im Schritt **400** ausgeführt wird. Ferner können in Ausführungsformen, in denen eine andere Anzahl an privaten Schlüsselgruppen **260** bis **280** verwendet wird, die Schritte für das Ausbilden der privaten Schlüssel **410** weniger oder mehr Auswahlschritte als die Schritte **600** bis **630**, die in Fig. 6 gezeigt sind, aufweisen.

[0072] Anschließend an die Auswahl der privaten Schlüssel **410** können drei digitale Signaturen D_1 bis D_3 im Schritt **420** erzeugt werden, die später zu einem Korrekturprogramm hinzugefügt werden, der zu einem Korrekturprogramm **140** gesendet wird, um die Authentizitätsprüfung und Integritätsprüfung zu ermöglichen Die Auswahl der privaten Schlüssel **410** ist in Fig. 7 detaillierter gezeigt.

[0073] Im Schritt **710** wird eine erste digitale Signatur D_1 durch Signieren der ersten Prüfsumme H_1, die im Schritt **520** ausgerechnet wurde, unter Anwendung des ersten im Schritt **610** ausgewählten privaten Schlüssels berechnet. Anschließend wird im Schritt **720** eine zweite digitale Signatur D_2 berechnet, indem die zweite Prüfsumme H_2, die sich aus dem Schritt **530** ergibt, unter Anwendung des aus der zweiten privaten Schlüsselgruppe im Schritt **620** ausgewählten zweiten privaten Schlüssels signiert. Schließlich wird im Schritt **730** eine dritte digitale Signatur D_3 berechnet, indem die in dem Schritt **540** berechnete dritte Prüfsumme H_3 unter Anwendung des im Schritt **630** ausgewählten dritten privaten Schlüssels signiert wird. Somit verwendet die vorliegende Erfindung Signaturtriplets, die auf einem Schlüssel pro Art basiert.

[0074] Dabei kann das Signieren der Prüfsummen H_1 bis H_3 einen weiteren Vorgang beinhalten, an dem sich eine Verschlüsselung (oder Verschlüsselung abhängig im angewendeten Signierungsalgorithmus) anschließt unter Anwendung des entsprechenden ausgewählten privaten Schlüssels. Alternativ können die digitalen Signaturen D_1 bis D_3 einfach durch Verschlüsseln oder Entschlüsseln der entsprechenden Prüfsummen H_1 bis H_3 mit dem entsprechenden ausgewählten privaten Schlüssel berechnet werden. Abhängig von der verwendeten Implementierung für die Schlüsselerzeugungsplattformen **110** bis **120** können andere Algorithmen zum Erzeugen der digitalen Signaturen D_1 bis D_3 verwendet werden.

[0075] Wiederum besitzt die spezielle Reihenfolge der in Fig. 7 gezeigten Schritte nur einen anschaulichen Charakter. In anderen Ausführungsformen können die digitalen Signaturen D_1 bis D_3 in einer beliebigen anderen Reihenfolge erzeugt werden, oder die Schritte des Erzeugens **420** der Signatur können abwechselnd mit Schritten der Prüfsummenkette **400** erfolgen und/oder abwechselnd mit der Auswahl **410** des privaten Schlüssels. Beispielsweise kann der Schritt **710** zum Berechnen der ersten digitalen Signatur D_1 ausgeführt werden, sobald die erste Prüfsumme H_1 berechnet ist (Schritt **520**) und der erste private Schlüssel ausgewählt ist (Schritt **610**). Die Berechnung **720** der zweiten digitalen Signatur D_2 kann in ähnlicher Weise vonstatten gehen. Ferner kann in Ausführungsformen, in denen mehr oder weniger digitale Signaturen dem Korrekturprogramm hinzugeführt werden, das Erzeugen **420** der Signaturen entsprechend mehr oder weniger Berechnungsschritte umfassen.

[0076] Nachdem die Signaturen im Schritt **420** erzeugt sind, kann im Schritt **430** bestimmt werden, ob eine der Signaturen eine Platzhalter- bzw. Dummysignatur ist. Das Verwenden von Platzhaltersignaturen kann es ermöglichen, dass geschädigte oder verlorene Schlüssel sicher verworfen werden. Wenn beispielsweise in einem der Schritte **610** bis **630** ein privater Schlüssel ausgewählt ist, der als geschädigt bekannt ist, so kann im Schritt **430** bestimmt werden, dass die entsprechende digitale Signatur eine Platzhaltersignatur ist. Aus diesem Grunde kann der Korrekturprogramm-Server **100** in geeigneter Weise speichern, welcher der privaten Schlüssel **200** bis **255** gestohlen oder verloren wurde, d. h., dass dieser oder diese nicht mehr zu verwenden sind. Dies kann beispielsweise erreicht werden, indem eine entsprechende Nachschlagtabelle geführt wird. Somit können gemäß der vorliegenden Ausführungsform verlorene oder gestohlene Schlüssel in sicherer Weise übersprungen werden. Es kann notwendig sein, diese als nichtig zu erklären oder sie zu ersetzen.

[0077] Wenn eine Platzhaltersignatur zu verwenden ist, kann die entsprechende digitale Signatur, die im Schritt **420** erzeugt wurde, durch einen Platzhalter im Schritt **440** ersetzt werden. Alternativ kann das Bestimmen **430**, ob eine Signatur eine Platzhaltersignatur ist, durchgeführt werden, bevor die Signatur erzeugt wird, und im Falle, dass eine Signatur eine Platzhaltersignatur ist, kann der entsprechende Schritt zum Erzeugen der Signatur **710**, **720**, **730** übersprungen werden, wodurch direkt ein Platzhalter für die entsprechende Signatur verwendet wird.

[0078] Im Schritt **450** kann ein Schlüsselindikator erzeugt werden. Der Schlüsselindikator kann angeben, welche Schlüssel im Schritt **410** aus den drei privaten Schlüsselgruppen **260**, **270**, **280** ausgewählt wurden. Eine beispielhafte Konfiguration des Schlüsselindikators ist in Fig. 12 gezeigt.

[0079] Gemäß der vorliegenden Ausführungsform ist der Schlüsselindikator ein 8 Bit-Ganzzahlwert. Die ersten beiden Bits können einen ersten Schlüsselindikator **1210** repräsentieren, der angibt, welcher der vier privaten Schlüssel **200** bis **215** der ersten privaten Schlüsselgruppe **260** ausgewählt ist, um die erste digitale Signatur D_1 zu erzeugen. Wenn beispielsweise der Wert der ersten beiden Bits 3 beträgt, kann dies anzeigen, dass der dritte private Schlüssel **210** im Schritt **610** gewählt ist und im Schritt **710** angewendet wird. Die nächsten beiden Bits können einen zweiten Schlüsselindikator **1220** bilden, der angezeigt, welcher der vier privaten Schlüssel **220** bis **235** der zweiten privaten Schlüsselgruppe **270** ausgewählt ist (Schritt **620**), um die zweite digitale Signatur zu erzeugen (Schritt **720**). In ähnlicher Weise können das fünfte und das sechste Bit für einen dritten Schlüsselindikator **1230** verwendet werden, um anzugeben, welcher der vier privaten Schlüssel **240** bis **255** im Schritt **630** ausgewählt ist, um die dritte digitale Signatur im Schritt **730** zu erzeugen.

[0080] In Ausführungsformen, in denen mehr oder weniger private Schlüsselgruppen verwendet werden, kann der Schlüsselindikator mehr oder weniger einzelne Schlüsselindikatoren **1210** bis **1230** enthalten. In anderen Ausführungsformen kann jede der privaten Schlüsselgruppen **260**, **270**, **280** mehr oder weniger als vier private Schlüssel enthalten. Somit können der erste bis dritte Schlüsselindikator **1210** bis **1230** länger oder kürzer sein. Des weiteren können die einzelnen Schlüsselindikatoren **1210** bis **1230** unterschiedlich angeordnet sein.

[0081] Der Schlüsselindikator kann ferner angeben, ob eine Platzhaltersignatur verwendet ist und wenn dies der Fall ist, welche der Signaturen die Platzhaltersignatur ist. Dazu repräsentieren die beiden letzten Bits des in Fig. 12 gezeigten Schlüsselindikators eine Platzhaltindikator **1240**. Gemäß der vorliegenden Ausführungsform gibt der Platzhalterindikator **1240** an, dass alle drei Signaturen D_1 bis D_3 zulässige Signaturen sind, wenn der Wert seiner beiden Bits Null beträgt. Ein Wert von 1, 2 oder 3 kann anzeigen, dass die erste, die zweite oder die dritte Signatur eine Platzhaltersignatur ist. Alternativ können die Werte des Platzhalterindikators selbstverständlich in anderer Weise zugeordnet sein.

[0082] In anderen Ausführungsformen können mehr oder weniger als zwei Signaturen verwendet werden, um das Korrekturprogramm zu signieren. Der Platzhalterindikator **1240** kann dann entsprechend länger oder kürzer sein als 2 Bits. Des weiteren können mehrere Signaturen Platzhaltersignaturen sein. In derartigen Ausführungsformen kann der Platzhalterindikator **1240** länger als 2 Bits sein.

[0083] Sobald der Schlüsselindikator erzeugt ist, kann ein Korrekturprogramm-Block im Schritt **460** zusammengestellt werden. Gemäß der folgenden Ausführungsform besitzt der Korrekturprogramm-Block das in Fig. 11 gezeigte Format.

[0084] Wie in Fig. 11 gezeigt ist, kann der Korrekturprogramm-Block **1100** mit der ersten digitalen Signatur **1110**, die im Schritt **710** erzeugt ist, beginnen, woran sich die zweite digitale Signatur **1120** und die dritte digitale Signatur **1130**, die in den Schritten **720** bzw. **730** erzeugt wurden, anschließen. Im Anschluss an die Signatur **1110** bis **1130** kann der Korrekturprogramm-Block **1100** den Schlüsselindikator **1140** enthalten, der sich aus dem Schritt **450** ergibt. Schließlich kann das Korrekturprogramm **1150** selbst in dem Korrekturprogramm-Block **1140** enthalten sein. Die spezielle Zusammensetzung des Korrekturprogramm-Blocks **1100**, wie er in Fig. 11 gezeigt wird, sollte nicht als beschränkend für die Erfindung betrachtet werden. In anderen Ausführungsformen kann der Korrekturprogramm-Block **1100** in unterschiedlicher Weise angeordnet sein.

[0085] Im Anschluss an das Zusammenstellen **460** des Korrekturprogramm-Blocks kann eine KEK-Verschlüsselung im Schritt **470** durchgeführt werden. Die KEK-Verschlüsselung **470** gemäß der vorliegenden Ausführungsform ist detaillierter in Fig. 8 gezeigt.

[0086] Im Schritt **810** wird ein Zufallsitzungsschlüssel der auch als der symmetrische Schlüssel im Folgenden bezeichnet ist, erzeugt. Die Erzeugung des Sitzungsschlüssels kann sicher gestaltet werden, indem Teile von Kennwortphrasen, beispielsweise durch gemeinsame Geheimnisse, auf Operartor-Karten von n-Verschlüsselungs HSM-Einrichtungen geteilt werden. In anderen Ausführungsformen wird der symmetrische Schlüssel nicht während des Ablaufs der Korrekturprogramm-Übertragung erzeugt, sondern wird stattdessen zuvor erzeugt und im Korrekturprogramm-Server **100** gespeichert. In derartigen Ausführungsformen kann der symmetrische Schlüssel in der Hardware versteckt werden. Dies kann auf diverse Arten verwirklicht werden. Beispielsweise kann eine Schlüsselerzeugung aus verteilten Quellen dazu dienen, da die Codierung für gewöhnlich durch viele Personen überprüft wird, so dass niemand alle Details kennt, wie der Schlüssel erzeug wird, aber die Gesamtinformation kann bei Bedarf wiedergewonnen werden. Alternativ kann der symmetrische Schlüssel innerhalb einiger HSM versteckt werden. Des weiteren können die Korrekturprogramm-Server **100** derartiger Ausführungsformen ein gewisses Verfahren enthalten, um den symmetrischen Schlüssel im Falle des Verlusts zu rekonstruieren.

[0087] Der Zufallssitzungsschlüssel kann im Schritt **820** verwendet werden, um den im Schritt **460** zusammengestellten Korrekturprogramm-Bock **1100** zu verschlüsseln. Dies kann unter Anwendung einer AES-Verschlüsselung erreicht werden. Schließlich

kann der Zufallssitzungsschlüssel im Schritt **830** unter Anwendung des KEK-Schlüssels **160** verschlüsselt werden. In anderen Ausführungsformen können die Schritte **820** und **830** in umgekehrter Reihenfolge ausgeführt werden.

[0088] Während der KEK-Verschlüsselung **470** können alle Informationen (außer den Kopfzeilen, die aus gewissen Gründen im Klartext sein müssen) in dem Ausgangsrückkopplungsmodus (OFD) verschlüsselt werden. Somit kann das Korrekturprogramm besser in dem Korrekturprogramm-Klienten **140** als Datenstrom entschlüsselt werden. Ein Anreichern ist nicht unter Umständen notwendig und ein blockweises Entschlüsseln kann problemlos durchgeführt werden. In alternativen Ausführungsformen kann der Verschlüsselungsrückkopplungsmodus (CFB) stattdessen für die Stromverschlüsselung verwendet werden. Der CFB-Modus ist klartextabhängig und Fehler werden über mindestens einen Block verteilt. Wenn jedoch der OFB-Modus verwendet wird, besitzt lediglich ein einziges Bit, das von einem Angreifer geändert oder hinzugefügt wird, die gleiche Verteilerwirkung wie ein zerstörter Block, und eine digitale Signaturverifizierung in dem Korrekturprogramm-Klienten **140** schlägt in beiden Fällen fehl. Somit wird durch Anwendung des OFB-Modus die Sicherheit erhöht.

[0089] Da die digitalen Signaturen D_1 bis D_3 an dem Korrekturprogramm-Block **1100** (Schritt **460**) vor dem Ausführen des Schritts **470** zur KEK-Verschlüsselung durchgeführt wurden, sind auch die digitalen Signaturen durch Verschlüsseln gemäß der vorliegenden Ausführungsform geschützt. Dies kann weiter das Risiko eines Auffindens einer Sicherheitslücke durch einen Angreifer verringern.

[0090] Um die symmetrische KEK-Verschlüsselung **470** noch sicherer zu gestalten, kann der Initialisierungsvektor, der im OFB-Modus angewendet wird, in einzigartiger Weise ausgewählt werden, d. h., so, dass sich dieser nie unter allen Korrekturprogrammen wiederholt. Dies kann beispielsweise durch Einschließen einer Sequenzzahl in einen fixierten Teil des Initialisierungsvektors erreicht werden. Alternativ kann eine Zeitmarkierung zu diesem Zweck verwendet werden.

[0091] Es sei nun wieder auf Fig. 4 verwiesen; ein Übertragungsblock kann im Schritt **480** zusammengestellt werden. Die Zusammensetzung des Übertragungsblocks der vorliegenden Ausführungsform ist in Fig. 14 gezeigt.

[0092] Insbesondere kann der Übertragungsblock **1400** aus dem verschlüsselten Sitzungsschlüssel **1410** bestehen, an den sich das verschlüsselte Korrekturprogramm **1420** anschließt, das aus den Schritten **830** und **820** resultiert. Gemäß der vorliegenden Ausführungsform ist der verschlüsselte Sitzungsschlüssel 128 Bits lang. Alternativ können andere Längen für den Sitzungsschlüssel in anderen Ausführungsformen verwendet werden.

[0093] Schließlich wird im Schritt **490** der Übertragungsblock zu dem Korrekturprogramm-Klienten **140** übertragen.

[0094] Wie bereits erwähnt ist, kann ein Hardwarefehler während des Signierens den privaten Schlüssel offen legen. Um diese Gefahr zu vermeiden, können die im Schritt **420** erzeugten Signaturen vor dem Senden zu dem Korrekturprogramm-Klienten **140** (geprüft werden im Schritt **490**). Somit kann jede Signatur D_1 bis D_3 durch ein unterschiedliches Programm überprüft werden. Beispielsweise kann durch den Vertreiber verifiziert werden, ob ein (noch nicht verkaufter) Korrekturprogramm-Klient **140** mit dem tatsächlichen Korrekturprogramm initialisiert wird, der die zu prüfende Signatur enthält.

[0095] Gemäß der zuvor beschriebenen Ausführungsform ist die Verifizierung der digitalen Signaturen D_1 bis D_3 in den Korrekturprogramm-Klienten **140** nur nach dem Lesen des vollständigen Korrekturprogramms **1150** möglich. In anderen Ausführungsformen kann es vorteilhaft sein, die Authentizitäts- und Integritätsprüfung für jeden Block in den Korrekturprogramm unmittelbar nach dessen Entschlüsselung durchzuführen. Dies ist in einer zweiten Ausführungsform möglich, die nunmehr mit Bezugnahme zu den Fig. 9, Fig. 10 und Fig. 13 beschrieben wird.

[0096] In dieser Ausführungsform kann der die Gesamtfunktion des Korrekturprogramm-Servers **100** derjenigen entsprechen, die in Fig. 4 gezeigt ist, und die Auswahl des privaten Schlüssels **410**, die Handhabung **430**, **440** für Platzhalter, die Erzeugung **450** des Schlüsselindikators, die Verschlüsselung **470** des KEK und die Zusammensetzung und Übertragung **480**, **490** des Übertragungsblocks können gleich sein, wie dies zuvor beschrieben ist. Jedoch wird eine modifizierte Prüfsummenkette **400** und eine Erzeugung **420** der Signatur angewendet. Des weiteren kann die Korrekturprogramm-Blockzusammenfügung **460** zu einem Korrekturprogramm-Block führen, der eine andere Zusammensetzung hat als dies zuvor beschrieben ist.

[0097] Es sei zunächst der Korrekturprogramm-Block, seine Zusammensetzung gemäß der vorliegenden Ausführungsform in Fig. 13 gezeigt.

[0098] Ähnlich zu dem Korrekturprogramm-Block **1100** der ersten Ausführungsform beginnt der Korrekturprogramm-Block **1300** mit drei digitalen Signaturen (D_1 bis D_3), **1310**, **1310**, **1330** an die sich ein Schlüsselindikator **1340** anschließt. Der Schlüsselindikator **1340** kann dem Schlüsselindikator **1140** entsprechen, wie er zuvor mit Bezug zu den Fig. 11

und Fig. 12 beschrieben ist. Die digitalen Signaturen **1310**, **1320**, **1330** werden jedoch in unterschiedlicher Weise im Vergleich zu den digitalen Signaturen der ersten Ausführungsform beschrieben, wie dies nachfolgend mit Bezug zu Fig. 10 erläutert ist. Ferner kann jedem Teilblock (R_1 bis R_n) **1355**, **1365**, **1375**, **1385** des Korrekturprogramms eine Verschlüsselungsprüfsumme (H_1 bis H_n) **1350**, **1360**, **1370**, **1380** vorausgehen.

[0099] Die Berechnung der Prüfsummen **1350**, **1360**, **1370**, **1380** gemäß der vorliegenden Ausführungsform wird im Schritt **400** aus Fig. 4 ausgeführt und ist detaillierter in Fig. 9 gezeigt.

[0100] Zunächst wird eine Prüfsumme H_n **1380** im Schritt **910** durch Kontrollsummieren einer Verkettung ($R_n|0$) des n-ten Teilblocks R_n **1385** des zu sendenden Korrekturprogramms mit einem Prüfsummenwert „0", der aus 0 Bits besteht, berechnet. In alternativen Ausführungsformen wird lediglich der Teilblock R_n im Schritt **910** der Kontrolle summiert. Sodann kann eine Prüfsumme H_{n-1} **1370** im Schritt **920** berechnet werden, indem die Verkettung ($R_{n-1}|H_n$) des $(n-1)^{ten}$ Teilblocks R_{n-1} **1375** mit der zuvor berechneten Prüfsumme H_n **1380** zur Prüfung berechnet wird. Im Folgenden werden H_{n-2} bis H_1 in analoger Weise in den Schritten **930** bis **940** berechnet. In anderen Ausführungsformen kann der letzte Schritt **940** weggelassen werden, und der erste Teilblock R_1 kann in den folgenden Schritten anstelle der Prüfsumme H_1, die sich aus dem Schritt **940** ergibt, verwendet werden. Die in Fig. 9 gezeigte Ausführungsform ermöglicht jedoch eine einfachere und zuverlässigere Programmierung.

[0101] Die Erzeugung **420** der modifizierten Signatur gemäß der vorliegenden Ausführungsform ist in Fig. 10 gezeigt.

[0102] Im Schritt **1010** wird eine erste digitale Signatur D_1 **1310** durch Signieren (d. h. Prüfsummenbildung und Entschlüsselung/Verschlüsselung oder einfach Entschlüsselung/Verschlüsselung, wie dies zuvor mit Bezug zu Fig. 7 erläutert ist) der ersten Prüfsumme H_1 **1350** berechnet werden, wobei der erste im Schritt **610** aus Fig. 6 ausgewählte private Schlüssel verwendet wird. Eine zweite digitale Signatur D_2 **1320** kann dann im Schritt **1020** entsprechend berechnet werden, indem die erste Prüfsumme H_1 **1350** unter Anwendung des im Schritt **620** ausgewählten zweiten privaten Schlüssels signiert wird. Schließlich wird im Schritt **1030** eine dritte digitale Signatur D_3 **1330** erzeugt, indem die erste Prüfsumme H_1 **1350** unter Anwendung des sich aus dem Schritt **630** ergebenden dritten privaten Schlüssels signiert wird.

[0103] Die dargestellte Reihenfolge aus Schritten sollte als nicht beschränkend für die vorliegende Erfindung erachtet werden. Beispielsweise können die digitalen Signaturen D_1 bis D_3 in unterschiedlicher Reihenfolge berechnet werden. Des weiteren können die Berechnungsschritte **1010** bis **1030** abwechselnd mit den Schritten zur Auswahl des privaten Schlüssels, wie sie in Fig. 6 gezeigt sind, und/oder abwechselnd mit der Prüfsummenverkettung, die in Fig. 9 gezeigt ist, ausgeführt werden. Beispielsweise können die digitalen Signaturen D_1 bis D_3 berechnet werden, sobald die erste Prüfsumme H_1 berechnet ist und sobald die entsprechenden privaten Schlüssel ausgewählt sind.

[0104] Sobald der Korrekturprogramm-Server einen Übertragungsblock **1400** mit einem verschlüsselten Korrekturprogramm zu den Korrekturprogramm gehend in **140** übertragen hat, kann der Korrekturprogramm sicher in dem Korrekturprogramm-Klienten **140** unter Anwendung der Matrix **150** für öffentliche Schlüssel und dem KEK-Schlüssel **160** installiert werden. Es wird nun ein sicherer Korrekturprogramm-Installisierungsprozess, der von dem Korrekturprogramm-Klienten **140** durchgeführt wird, gemäß einer Ausführungsform mit Bezugnahme zu den Fig. 15 bis Fig. 17 beschrieben. Dieser Korrekturprogramm-Installierungsprozess kann in einer Ausführungsform eingesetzt werden, in der die digitalen Signaturen D_1 bis D_3 **1110**, **1120**, **1130** lediglich nach dem Entschlüsseln des gesamten Korrekturprogramms verifiziert werden.

[0105] Zunächst sei auf Fig. 15 verwiesen; der Übertragungsblock **1400** wird im Korrekturprogramm-Klienten **140** im Schritt **1500** empfangen. Anschließend wird der verschlüsselte Zufallssitzungsschlüssel **1410** im Schritt **1510** unter Anwendung des KEK-Schlüssels **160** entschlüsselt. Im Schritt **1520** wird der verschlüsselte Korrekturprogramm-Block **1420** durch den AES-Algorithmus unter Anwendung des zuvor im Schritt **1510** gewonnen Zufallsitzungsschlüssels entschlüsselt. Die Entschlüsselung in den Schritten **1510** und **1520** kann unter Anwendung des OFB-Modus, der zuvor mit Bezugnahme zu Fig. 8 beschrieben ist, erreicht werden.

[0106] Im Schritt **1530** können die öffentlichen Schlüssel, die zum Verifizieren der Signaturen zu verwenden sind, ausgewählt werden. Dies ist detaillierter in Fig. 16 gezeigt.

[0107] Zunächst wird im Schritt **1610** ein erster öffentlicher Schlüssel aus der ersten öffentlichen Schlüsselgruppe **360** unter Verwendung des zuvor im Schritt **1520** entschlüsselten ersten Schlüsselindikators **1210** ausgewählt. Insbesondere kann der Schlüssel unter den öffentlichen Schlüsseln **300** bis **315**, auf den der Schlüsselindikator **1210** zeigt, als der erste öffentliche Schlüssel ausgewählt werden. Der ausgewählte erste öffentliche Schlüssel kann dem ersten privaten Schlüssel entsprechen, der von dem Korrekturprogramm-Server **100** im Schritt **610**

ausgewählt wurde (siehe Fig. 6). Folglich können die Schritt **1620** und **1630** das Auswählen des zweiten und des dritten öffentlichen Schlüssels aus der zweiten bzw. der dritten öffentlichen Schlüsselgruppe **370**, **380** beinhalten, wobei der zweite bzw. der dritte Schlüsselindikator **1220**, **1230** verwendet wird. Der zweite und der dritte öffentliche Schlüssel können dem zweiten bzw. dem dritten privaten Schlüssel entsprechen, der von dem Korrekturprogramm-Server **100** in den Schritten **620** bzw. **630** ausgewählt wurde.

[0108] Selbstverständlich können die Schritte zum Auswählen der öffentlichen Schlüssel **1610** bis **1630** in einer anderen Reihenfolge durchgeführt werden. Ferner können in Ausführungsformen, in denen mehr oder weniger als drei öffentliche Schlüsselgruppen **360** bis **380** (und folglich mehr oder weniger als drei private Schlüsselgruppen **260** bis **280**) verwendet werden, die Auswahlschritte **1230** für das Auswählen des öffentlichen Schlüssels entsprechend mehr oder weniger Auswahlschritte beinhalten.

[0109] Nach Auswahl der öffentlichen Schlüssel können die Signaturen **1110** bis **1130** im Schritt **1540** verifiziert werden. Die Teilschritte der Signaturverifizierung gemäß der vorliegenden Ausführungsform sind in Fig. 17 gezeigt.

[0110] Zunächst wird eine erste Prüfsummenkette im Schritt **700** ausgeführt. Gemäß der Ausführungsform entspricht von dem Korrekturprogramm-Klienten **140** ausgeführte Prüfsummenkette **700** der Prüfsummenkette, die von dem Korrekturprogramm-Server **100** gebildet wird, wie dies zuvor mit Bezug zu Fig. 5 beschrieben ist.

[0111] Anschließend werden im Schritt **710** die Prüfsummen H_1, H_2 und H_3, die sich aus den Schritten **520**, **530** und **540** ergeben, erneut zur Kontrolle aufsummiert. In Ausführungsformen, in denen das von dem Korrekturprogramm-Server **100** in den Schritten **710** bis **730** ausgeführte Signieren keine weitere Prüfsummenberechnung ergibt, sondern eine Einschlüsselung/Verschlüssung beinhaltet, kann der Schritt **1710** weggelassen werden.

[0112] Anschließend zum Schritt **1710** werden die digitalen Signaturen (D_1 bis D_3) **1110**, **1120**, **1130**, die beim Einschlüsseln des verschlüsselten Korrekturprogramm-Blocks **1420** im Schritt **1520** erhalten werden, unter Anwendung des ersten, des zweiten bzw. des dritten öffentlichen Schlüssels entschlüsselt.

[0113] Sodann werden in den Schritten **1750** bis **1770** die entschlüsselten digitalen Signaturen, die sich aus den Schritten **1720** bis **1740** ergeben, mit den summierten Prüfsummen H_1 bis H_3, die im Schritt **1710** ermittelt wurden, verglichen. In Ausführungsformen, in denen Schritt **1710** weggelassen ist, können die entschlüsselten digitalen Signaturen direkt mit den prüfsummierten Summen H_1 bis H_3 verglichen werden.

[0114] Im Schritt **1780** wird bestimmt, ob es Platzhaltersignaturen und den digitalen Signaturen D_1 bis D_3 **1110**, **1120**, **1130** gibt. Dies kann bewerkstelligt werden, indem der Platzhalterindikator **1240** des Schlüsselindikators **1140** überprüft wird.

[0115] Wenn dies der Fall ist, wird im Schritt **1790** bestimmt, dass die von dem Platzhalterindikator **1240** als eine Platzhaltersignatur gekennzeichnete digitale Signatur während des Rests des sicheren Korrekturprogramm-Installationsprozesses unberücksichtigt bleibt. In anderen Ausführungsformen können mehr als eine digitale Signatur eine Platzhaltersignatur sein, wie dies zuvor erläutert ist. In derartigen Ausführungsformen können alle Platzhaltersignaturen während des weiteren Korrekturprogramm-Installierungsprozesses ignoriert werden.

[0116] Wenn Schritt **1780** zeigt, dass der Platzhalterindikator **1240** angibt, dass alle digitalen Signaturen **1110** bis **1130** zulässige Signaturen sind, d. h. keine Platzhaltersignaturen sind, so wird Schritt **1790** nicht ausgeführt und alle digitalen Signaturen **1110**, **1120**, **1130** können während der folgenden Schritte der sicheren Korrekturprogramm-Installation berücksichtigt werden.

[0117] Sobald die Signaturen im Schritt **1540** verifiziert sind, wird im Schritt **1550** bestimmt, ob alle digitalen Signaturen **1110**, **1120**, **1130** geordnet sind. Gemäß der vorliegenden Ausführungsform ist dies der Fall, wenn die Vergleichsschritte **1750** bis **1770** die Gleichheit anzeigen. Wenn dies der Fall ist, kann das Korrekturprogramm in dem Korrekturprogramm-Klienten **140** im Schritt **1560** installiert werden. Der Schritt **1560** kann das Bereitstellen eines Berichts der erfolgreichen Korrekturprogramm-Installation für den Anwender des Korrekturprogramm-Klienten **140** und/oder das Informieren des Korrekturprogramm-Servers **100** in entsprechender Weise beinhalten.

[0118] Wenn jedoch zumindest einer der Schritte **1750** bis **1770** anzeigt, dass eine entschlüsselte digitale Signatur nicht identisch ist zu der entsprechenden (kontrollsummierten) Prüfsumme, wird im Schritt **1570** bestimmt, dass der empfangene Korrekturprogramm nicht in dem Korrekturprogramm-Klienten **140** zu installieren ist. Dies kann beispielsweise beinhalten, dass der Anwender eine Fehlernachricht erhält und/oder dass der Korrekturprogramm-Server **100** ihm mitteilt, dass die Korrekturprogramm-Installierung fehlgeschlagen hat.

[0119] Es ist zu beachten, dass die spezielle Reihenfolge der in den Fig. 15 bis Fig. 17 gezeigten

Schritte nur der Darstellung wegen ausgewählt ist. In anderen Ausführungsformen können die einzelnen Teilschritte in unterschiedlicher Reihenfolge ausgeführt werden; beispielsweise können die Entschlüsselungsschritte **1720** bis **1740** abwechselnd mit den Vergleichsschritten **1750** bis **1770** ausgeführt werden. Des weiteren kann der Prüfsummierungsschritt **1710** in drei einzelne Teilschritte unterteilt werden, die ebenso abwechselnd mit der Entschlüsselung in den Vergleichsschritten **1720** bis **1770** ausgeführt werden können. Des weiteren können die einzelnen Schritte **510** bis **540** der Prüfsummenkette **1700** auf die Prüfsummeneinschlüsselungs- und Vergleichsschritte **1710** bis **1770** aufgeteilt werden.

[0120] Des weiteren können in Ausführungsformen, in denen mehr oder weniger private und öffentliche Schlüsselgruppen verwendet werden, die Signaturverifizierungsschritte **1540** entsprechend mehr oder weniger prüfsummierende, Entschlüsselungs- und Vergleichsschritte **1710** bis **1770** umfassen.

[0121] Des weiteren kann die Handhabung der Platzhalter in den Schritten **1780** und **1790** vor dem Entschlüsselungsschritt **1720** oder sogar zu Beginn der Signaturverifizierung **1540** oder vor der Auswahl **1530** des öffentlichen Schlüssels ausgeführt werden. Wenn in den Schritten **1780** derartige Ausführungsformen bestimmt wird, dass eine spezielle Signatur **1110**, **1120**, **1130** eine Platzhaltersignatur ist, können die entsprechenden Schritte der Auswahl **1530** öffentlicher Schlüssel und der Verifizierung **1700** bis **1770** der Signatur weggelassen werden. Wenn beispielsweise die dritte digitale Signatur D_3 eine Platzhaltersignatur ist, kann es unter Umständen nicht erforderlich sein, die dritte Prüfsumme H_3 im Schritt **540** auszurechnen, einen dritten öffentlichen Schlüssel im Schritt **1630** auszuwählen, die dritte Prüfsumme H_3 im Schritt **1710** kontroll zu summieren, die dritte digitale Signatur im Schritt **1740** zu entschlüsseln und/oder den Vergleich im Schritt **1770** auszuführen.

[0122] Wie zuvor bereits erwähnt ist, ermöglicht es das Korrekturprogramm-Installationsschema der vorliegenden Ausführungsform, dass die Signaturen nur nach dem vollständigen Entschlüsseln des gesamten verschlüsselten Korrekturprogramm-Blocks **1420** verifiziert werden. Jedoch gibt es Ausführungsformen, in denen es vorteilhaft ist, zu verifizieren, ob das Korrekturprogramm unmittelbar nach dem Entschlüsseln einzelner Teilblöcke des Korrekturprogramms zu installieren ist. Die Funktionsweise des Korrekturprogramm-Servers **100** in einer derartigen Ausführungsform ist zuvor mit Bezug den Fig. 9, Fig. 10 und Fig. 13 beschrieben. Eine entsprechende sichere Korrekturprogramm-Installation, die von dem Korrekturprogramm-Klienten **140** auszuführen ist, wird nunmehr mit Bezugnahme zu den Fig. 18 bis Fig. 20 beschrieben.

[0123] Im Schritt **1800** aus Fig. 18 wird ein Übertragungsblock **1400** in dem Korrekturprogramm-Klienten **140** empfangen. Im Schritt **1810** wird der in dem Übertragungsblock **1400** enthaltene verschlüsselte Zufallsitzungsschlüssel **1410** unter Anwendung des KEK-Schlüssels **160** entschlüsselt. Dieser Schritt entspricht dem Schritt **1510** aus Fig. 15.

[0124] Nachfolgend wird im Schritt **1820** der im Schritt **1810** gewonnen Zufallssitzungsschlüssel verwendet, um die in dem verschlüsselten Korrekturprogramm-Block **1420** enthaltene Information zu entschlüsseln. Die im Schritt **1820** ausgeführte Entschlüsselung kann in gleicher Weise erfolgen, wie die im Schritt **1520** durchgeführte Entschlüsselung. Jedoch werden gemäß der vorliegenden Ausführungsform lediglich die verschlüsselten Signaturen und der verschlüsselte Schlüsselindikator im Schritt **1820** entschlüsselt. Der Rest des verschlüsselten Korrekturprogramm-Blocks **1420** wird später teilblockweise in den Schritten **1840** und **1850** entschlüsselt.

[0125] Im Schritt **1830** werden die öffentlichen Schlüssel auf der Grundlage des Schlüsselindikators **1340**, der im Schritt **1820** gewonnen wird, ausgewählt. Dies entspricht der Auswahl **1530** öffentlicher Schlüssel, wie dies zuvor mit Bezug zu Fig. 15 beschrieben ist. Anschließend können im Schritt **1840** die digitalen Signaturen (D_1 bis D_3) **1310**, **1320**, **1330**, die Schritt **1820** ermittelt wurden, verifiziert werden. Dies ist detaillierter in Fig. 19 gezeigt.

[0126] Zunächst werden im Schritt **1900** die verschlüsselte erste Prüfsumme und der verschlüsselte erste Teilblock des Korrekturprogramms, die beide in dem Übertragungsblock **1400** empfangen werden, unter Verwendung des im Schritt **1810** gewonnenen Zufallssitzungsschlüssels entschlüsselt. Dies kann in gleicher Weise geschehen wie die Entschlüsselung im Schritt **1520**, die zuvor mit Bezug zu Fig. 15 beschrieben ist. Danach wird die erste Prüfsumme H_1 **1350**, die im Schritt **1900** wiedergewonnen wird, im Schritt **1910** kontrollsummiert. In anderen Ausführungsformen, insbesondere in Ausführungsformen, in denen die von dem Korrekturprogramm-Server **100** in den Schritten **1010** bis **1030** durchgeführte Signierung keine Prüfsummenbildung enthält, kann der Schritt **1910** weggelassen werden.

[0127] Anschließend werden in den Schritten **1920** bis **1940** die digitalen Signaturen (D_1 bis D_3) **1310** bis **1330**, die sich aus dem Schritt **1820** ergeben, unter Anwendung des ersten bis dritten öffentlichen Schlüssels entschlüsselt, wobei diese im Schritt **1830** ausgewählt wurden. Jedes der Ergebnisse der Schritte **1920** bis **1940** kann dann mit dem Ergebnis des Schritts **1910** in den Schritten **1950** bis **1970** verglichen werden. In Ausführungsformen, in denen der Schritt **1910** weggelassen wird, können die sich aus den Schritten **1920** bis **1940** ergebenden entschlüs-

selten Signaturen mit der Prüfsumme H_1, die sich aus dem Schritt **1900** ergibt, stattdessen verglichen werden.

[0128] Schließlich kann die Handhabung von Platzhaltersignaturen in den Schritten **1980** und **1990** ausgeführt werden. Dies entspricht der Platzhaltersignaturhandhabung, wie sie zuvor mit Bezug zu den Schritten **1780** und **1790** aus Fig. 17 beschrieben ist.

[0129] Wiederum sind die in den Fig. 18 und Fig. 19 gezeigten Schritte in dieser speziellen Abfolge lediglich anschaulicher Natur und sollen nicht als Einschränkung der Erfindung verstanden werden. In anderen Ausführungsformen können die entsprechenden Schritte unterschiedlich, beispielsweise abwechselnd, ausgeführt werden. Ferner kann die Handhabung der Platzhaltesignaturen in den Schritten **1980** und **1990** beispielsweise zu Beginn der Signaturverifizierung **1840** oder vor der Auswahl **1830** öffentlicher Schlüssel ausgeführt werden. In derartigen Ausführungsformen können alle Schritte der Auswahl **1830** öffentlicher Schlüssel und der Signaturverifizierung **1900** bis **1970**, die sich auf eine von dem Platzhalterindikator **1240** als eine Platzhaltersignatur gekennzeichnete digitale Signatur beziehen, weggelassen werden.

[0130] Es sei nun auf Fig. 18 verwiesen; der Rest des verschlüsselten Korrekturprogramm-Blocks **1420** kann dann teilblockweise im Schritt **1850** entschlüsselt werden. Ein teilblockweises Korrekturprogramm-Entschlüsselungsschema gemäß der vorliegenden Ausführungsform ist in Fig. 20 gezeigt.

[0131] Dabei kann zuerst im Schritt **2000** überprüft werden, ob alle Signaturen D_1 bis D_3 vorhanden sind. Der Schritt **2000** kann das Bestimmen beinhalten, ob alle Vergleichsschritte **1950** bis **1970** die Gleichheit ergeben. Wie zuvor erläutert ist, werden mögliche Platzhaltersignaturen für dieses Bestimmen nicht berücksichtigt.

[0132] Wenn dies nicht der Fall ist, d. h., wenn mindestens eine der entschlüsselten digitalen Signaturen D_1 bis D_3 nicht identisch zu der (kontrollsummierten) Prüfsumme H_1 ist, wird im Schritt **2050** bestimmt, dass das vorliegende Korrekturprogramm nicht zu installieren ist. Dies entspricht dem Schritt **1570** aus Fig. 15.

[0133] Wenn jedoch alle Signaturen in Ordnung sind, können die zweite verschlüsselte Prüfsumme und der zweite verschlüsselte Korrekturprogramm-Teilblock, die in dem verschlüsselten Korrekturprogramm-Block **1420** enthalten sind, unter Anwendung des im Schritt **1810** gewonnenen Zufallssitzungsschlüssels entschlüsselt werden. Die Entschlüsselung im Schritt **2005** kann in gleicher Weise ausgeführt werden, wie die zuvor erläuterte Entschlüsselung im Schritt **1820**. Anschließend wird im Schritt **2010** die Verkettung von $(R_1|H_2)$ des ersten Teilblocks R_1 **1355** (der bereits im Schritt **1900** erhalten wurde) mit der zweiten Prüfsumme H_2 **1360**, die im Schritt **2005** erhalten wurde, im Schritt **2010** kontrollsummiert.

[0134] Das Ergebnis kann mit der Prüfsumme H_1 **1350**, die zuvor im (im Entschlüsselungsschritt **1900**) ermittelt wurde. Wenn im Schritt **2020** bestimmt wird, dass diese beiden Prüfsummen nicht identisch sind, kann das Korrekturprogramm-Entschlüsselungsschema zum Schritt **2015** weiterschreiten, um zu bestimmen, dass das Korrekturprogramm nicht zu installieren ist. Ansonsten können die Schritte **2005** bis **2020** entsprechend für die dritte bis letzte Prüfsumme und wird in dem verschlüsselten Korrekturprogramm-Block **1420** wiederholt.

[0135] Wenn der Vergleich im Schritt **2020** positiv ist für alle entschlüsselten Prüfsummen und Teilblöcke, kann die Verkettung $(R_n|0)$ des letzten Korrekturprogramm-Teilblocks R_n **1385** mit dem Prüfsummenwert „0", das aus 0 Bits besteht, im Schritt **2030** berechnet werden. Dieser Wert kann mit der letzten Prüfsumme H_n **1380** des Korrekturprogramm-Blocks **1300** im Schritt **2035** verglichen werden.

[0136] Wenn im Schritt **2040** bestimmt wird, dass die beiden Werte identisch sind, wird im Schritt **2045** der Korrekturprogramm installiert, und der Anwender und/oder der Korrekturprogramm-Server **100** können über die erfolgreiche Korrekturprogramm-Installation unterrichtet werden. Ansonsten wird im Schritt **2040** bestimmt, dass der Korrekturprogramm nicht zu installieren ist.

[0137] Gemäß der in den Fig. 18 bis Fig. 20 gezeigten Ausführungsform wird ein empfangenes Korrekturprogramm entweder vollständig in Schritt **2045** installiert oder gar nicht installiert, selbst wenn lediglich nur ein Korrekturprogramm-Teilblock schadhaft ist. In einer derartigen Ausführungsform reicht ein ungedrehtes Bit oder ein gelöschtes/eingefügtes Byte in dem Korrekturprogramm aus, um eine Initialisierung des Korrekturprogramm-Klienten **140** zu verhindern. Dies kann eine erhöhte Sicherheit bieten, wobei beispielsweise vor Würmer geschützt wird, die Korrekturprogramme in einer Weise modifizieren, dass ein Anwender den Eindruck erhält, er hätte lediglich ein minderwertig hergestelltes Korrekturprogramm erhalten.

[0138] In anderen Ausführungsformen, in denen jedoch eine Sicherheit in diesem Ausmaße nicht erforderlich ist, können die Teilblöcke **1355**, **1365**, **1375**, **1385** des Korrekturprogramms, für die in den Schritten **2000**, **2020** oder **2040** eine positive Antwort erhalten wurde, installiert werden. Somit führt eine negative Antwort in den Schritten **2000**, **2020**

oder **2040** nicht notwendigerweise zu einem Verwerfen des gesamten Korrekturprogramms im Schritt **2050**, sondern lediglich des Teilblocks **1355**, **1375**, **1385**, der aktuell geprüft wird. In derartigen Systemen kann die Sicherheit dennoch verbessert werden, indem der Anwender gezwungen wird, beispielsweise eine MD5-Summe zu prüfen, wenn er ein Korrekturprogramm erhält, oder eine gewisse Prüfung bei der Korrekturprogramm-Installierung einzubauen. Das Berücksichtigen lediglich einzelner Teilblöcke **1355**, **1365**, **1375**, **1385** des Korrekturprogramms ist durch die Anwendung des OFD-Verschlüsselungsmodus zulässig, da der OFD-Modus Fehler lokalisiert.

[0139] Gemäß den beschriebenen Ausführungsformen wird ein sicherer Korrekturprogramm-Installationsprozess durch den Korrekturprogramm-Klienten **140** automatisch erreicht, und der Anwender hat keinen Einfluss auf den Prozess. Des weiteren besitzt der Anwender keine Möglichkeit, zu erkennen, was innerhalb des Korrekturprogramm-Klienten **140** abläuft. Jedoch kann ein Anwender mit einer Fehlernachricht im Schritt **1570** oder **2050** oder einem Licht versorgt werden, dass der Korrekturprogramm im Schritt **1560** oder **2045** korrekt installiert wurde.

[0140] Ferner gibt es gemäß der vorliegenden Ausführungsform keine Möglichkeit, die beschriebenen Sicherheitsfunktionen auszuschalten. Dies kann Angriffe auf Grund von „Schatten"-Variablen verhindern, die über den Zustand des Sicherheitsabschaltanschlusses Bericht erstatten. In Ausführungsformen, in denen das Ausschalten der Sicherheitsfunktion aus Leistungsgründen gewünscht ist, kann damit sichergestellt werden, dass diese Variable durch Software nicht modifizierbar ist. Dies liegt darin begründet, dass die Software, die eine derartige Variable einstellt, ebenso angegriffen werden könnte.

[0141] Wie aus der obigen Beschreibung der Ausführungsformen hervorgeht, werden Verfahren und Systeme zum Aktualisieren von Software mit einem erhöhten Geheimnisschutz und einer verbesserten Schlüsselverlusttoleranz bereitgestellt. Insbesondere die Sicherheit kann deutlich erhöht werden, indem unterschiedliche Schlüsselerzeugungsplattformen **110**, **120**, **130** verwendet werden.

[0142] Durch Kombinieren von Schlüsseln von den unterschiedlichen Plattformen **110**, **120**, **130** wird das Risiko schwacher Signaturschlüssel verringert, was sich ergibt, wenn eine Plattform **110**, **120**, **130**, die zur Schlüsselerzeugung verwendet wird, Sicherheitslücken aufweist. Für gewöhnlich führt dieses Risiko zum Problem von Hardwareänderungen in Halbleiterchips, um darin neue Schlüssel einzubetten, was mit hohen Kosten verbunden ist. Somit verringert die vorgeschlagene Kombination von Schlüsseln, die in unterschiedlichen Plattformen **110**, **120**, **130** erzeugt werden, auch die Produkt- und Wartungskosten.

[0143] Ein Schutz gegenüber Schlüsselverlust, d. h. in Fällen, in denen Schlüssel nicht mehr verfügbar sind, kann erreicht werden, indem ein Bit-Indikator **1140**, **1340** verwendet wird, der auf die entsprechenden Schlüssel zeigt, die aus einer Matrix **150** auszuwählen sind. Dadurch wird die Notwendigkeit einer Hardwareänderung oder der Verwendung von Gültigkeitslisten im Falle eines Schlüsselverlusts vermieden.

[0144] Ferner kann die vorgeschlagene Anwendung dreier Schlüssel aus einer Matrix, die 12 Schlüssel der drei digitalen Signaturen D_1 bis D_3 umfasst, den Schutz im Hinblick auf Situationen verbessern, in denen Schlüssel offengelegt/gestohlen werden, und damit öffentlich verfügbar sind. Dies kann es für Angreifer schwer machen, manipulierte Korrekturprogramme einzuführen. Es muss zumindest ein vollständiger Satz an drei geheimen Schlüsseln (die durch unterschiedliche Verfahren und Einheiten geschützt werden können) gestohlen werden, die einen Schlüsselsatz aus drei Spalten **260**, **270**, **280** der Schlüsselmatrix bilden, um einen Angriff starten zu können. Selbst in einem derartigen Falle können lediglich ungefähr 1/64 aller Korrekturprogramm-Klienten **140** durch manipulierte Einheiten infiziert werden, vorausgesetzt, dass die Manipulation nicht bekannt ist. Für bekannte Manipulationen können theoretisch bis zu 10 Schlüssel manipuliert werden, ohne die Sicherheit zu gefährden: Der Platzhalterindikator **1240** mit den beiden Bits des erwähnten Schlüsselindikators **1140**, **1340** kann es ermöglichen, dass alle vier Schlüssel einer der privaten Schlüsselgruppen **260**, **270**, **280** manipuliert werden. In alternativen Ausführungsformen, in denen vier Signaturen verwendet werden und keine Platzhaltersignaturen zulässig sind, kann die Wahrscheinlichkeit von 1/64 sogar auf 1/256 abfallen (wobei mindestens vier private Schlüssel öffentlich werden müssen). Jedoch muss in derartigen Ausführungsformen mindestens ein gültiger Schlüssel aus jeder privaten Schlüsselgruppe vorhanden sein, um die Sicherheit aufrecht zu erhalten.

[0145] Die zuvor mit Bezug zu den Fig. 5 und Fig. 9 beschriebenen Prüfsummenketten ermöglichen es, unterschiedliche Parteien in den Signierungsprozess einzubinden, wobei keine davon in der Lage ist, anzugeben, ob sie das gleiche Korrekturprogramm-Paket signiert haben. Dies kann weitere Sicherheit in einigen Szenarien bereitstellen, wobei die Kosten gleich bleiben.

[0146] Ferner liefern die in den Fig. 5 und Fig. 9 gezeigten Prüfsummenketten einen wichtigen Sicherheitsgewinn. Neueste Ergebnisse deuten darauf hin, dass Fehler in den kryptographischen Prüfsummenfunktionen, etwa MD5 und SHA-O (sicherer Prüfsum-

menalgorithmus 0) enthalten sind. Es nicht klar, ob SHA-1 gebrochen werden kann und ob dieses je in einer verwertbaren Weise gebrochen werden kann. Wenn jedoch selbst unterschiedliche Texte mit dem gleichen SHA-1-Prüfsummenwert konstruiert werden können, ist es vollständig unrealistisch, dass dies für zwei oder sogar mehr verkettete Prüfsummen der Fall sein könnte, wie dies durch die in den Fig. 5 und Fig. 9 gezeigten Prüfsummenketten erreicht wird.

[0147] Das vorgeschlagene Konzept garantiert Korrekturprogramm-Integrität auf sehr hohem Niveau unter Anwendung einiger digitaler Signaturen D_1 bis D_3, wobei heterogene öffentliche Schlüssel in Verbindung mit einer sorgfältigen Schlüsselverwaltung verwendet werden. Selbst in Ausführungsformen, in denen lediglich HSM-Schlüssel verwendet werden, bleibt die Sicherheit hoch. Die Korrekturprogramm-Inhalte können durch Verschlüsseln mit einem KEK-Schlüssel geschützt werden, der in der Firmware des Korrekturprogramm-Klienten versteckt werden kann.

[0148] In Ausführungsformen, in denen eine vertrauensvolle dritte Partei (TTP) in dem Sicherheitsprozess beteiligt ist, müssen lediglich Prüfsummenwerte signiert werden. Dadurch kann die Sicherheit noch weiter erhöht werden, da das Risiko anzeigend der Korrekturprogramm-Quelle vermieden werden kann.

[0149] Die Kosten für die vorgeschlagene Lösung sind im Vergleich zu den erreichten Vorteilen gering. Die Korrekturprogramm-Entwicklung und Verteilung ist wesentlich teurer als die Schlüsselhandhabung, die Sicherung, das digitale Signieren und das Verschlüsseln, selbst in Ausführungsformen, in denen mehrere Instanzen bei der Prozessierung beteiligt sind. Die Herstellungskosten für die Korrekturprogramm-Klienten **140** sind nahezu fest und ändert sich lediglich marginal durch Hinzufügen der beschriebenen sicheren Korrekturprogramm-Funktionalität. Die Verschlüsselung und die Signaturprüfung erfordert lediglich eine minimale Zeit während des Hochlaufens der Korrekturprogramm-Klienten **140**. Der Sicherheitszugewinn gegenüber konventionellen reinen Korrekturprogramm-Systemen ist jedoch beachtlich. Somit können die vorliegenden Ausführungsformen die Sicherheit, die Zuverlässigkeit und die Effizienz von Korrekturprogramm-Systemen deutlich verbessern, ohne die entsprechenden Kosten wesentlich zu erhöhen.

[0150] Obwohl die Erfindung in Bezug auf die physikalischen Ausführungsformen beschrieben ist, die entsprechend der Erfindung gestaltet sind, erkennt der Fachmann, dass diverse Modifizierungen, Variationen und Verbesserungen der vorliegenden Erfindung im Lichte der obigen Lehren und innerhalb des Bereichs der angefügten Patentansprüche durchgeführt werden können, ohne von dem Schutzbereich der Erfindung abzuweichen. Ferner sind jene Bereiche, in denen angenommen wird, dass der Fachmann damit vertraut ist, nicht beschrieben, um die Erfindung nicht unnötig zu verdunkeln. Selbstverständlich ist die Erfindung nicht auf die speziellen anschaulichen Ausführungsformen einzuschränken, sondern die Erfindung ist lediglich durch den Schutzbereich der angefügten Patentansprüche definiert.

Patentansprüche

1. Patch-Server (**100**), der mit einem Patch-Client (**140**) verbunden ist, um dem Patch-Client einen Patch (**1150**, **1355**, **1365**, **1375**, **1385**) bereitzustellen, und Folgendes umfasst:
eine erste Schlüsselerzeugungsplattform (**110**), die zum Erzeugen einer ersten Gruppe privater Schlüssel (**260**), welche eine Vielzahl erster privater Schlüssel (**200**–**215**) umfasst, eingerichtet ist;
eine zweite Schlüsselerzeugungsplattform (**120**), die von der ersten Schlüsselerzeugungsplattform verschieden ist und zum Erzeugen einer zweiten Gruppe privater Schlüssel (**270**), welche eine Vielzahl zweiter privater Schlüssel (**220**–**235**) umfasst, eingerichtet ist;
eine erste Schlüsselauswahleinrichtung, die zum Auswählen (**410**, **610**) eines der ersten privaten Schlüssel aus der ersten Gruppe privater Schlüssel eingerichtet ist;
eine zweite Schlüsselauswahleinrichtung, die zum Auswählen (**410**, **620**) eines der zweiten privaten Schlüssel aus der zweiten Gruppe privater Schlüssel eingerichtet ist;
eine erste Signaturerzeugungseinrichtung, die zum Erzeugen (**420**, **710**, **1010**) einer ersten digitalen Signatur (**1110**, **1310**) auf Basis des Patchs und des ersten ausgewählten privaten Schlüssels eingerichtet ist;
eine zweite Signaturerzeugungseinrichtung, die zum Erzeugen (**420**, **720**, **1020**) einer zweiten digitalen Signatur (**1120**, **1320**) auf Basis des Patchs und des zweiten ausgewählten privaten Schlüssels eingerichtet ist; und
einen Transmitter, der zum Übertragen (**490**) des Patchs zusammen mit der ersten und zweiten digitalen Signatur an den Patch-Client eingerichtet ist.

2. Patch-Server nach Anspruch 1, ferner umfassend:
eine erste Hash-Einrichtung, die zum Berechnen (**400**, **510**) einer ersten Hash-Summe auf Basis des Patchs eingerichtet ist; und
eine zweite Hash-Einrichtung, die zum Berechnen (**400**, **520**) einer zweiten Hash-Summe auf Basis der ersten Hash-Summe eingerichtet ist;
wobei die erste Signaturerzeugungseinrichtung ferner zum Erzeugen (**420**, **710**) der ersten digitalen Signatur auf Basis der zweiten Hash-Summe eingerichtet ist.

3. Patch-Server nach Anspruch 2, ferner umfassend:
eine dritte Hash-Einrichtung, die zum Berechnen (**400**, **530**) einer dritten Hash-Summe auf Basis der zweiten Hash-Summe eingerichtet ist;
wobei die zweite Signaturerzeugungseinrichtung ferner zum Erzeugen (**420**, **720**) der zweiten digitalen Signatur auf Basis der dritten Hash-Summe eingerichtet ist.

4. Patch-Server nach Anspruch 1, ferner umfassend:
eine Hash-Einrichtung, die zum Berechnen (**400**, **910**–**940**) einer Vielzahl von Hash-Summen eingerichtet ist;
wobei der Patch eine Vielzahl von Datensätzen (**1355**, **1365**, **1375**, **1385**) umfasst;
wobei die Hash-Einrichtung ferner zum Berechnen (**400**, **910**) einer ersten Hash-Summe der Vielzahl von Hash-Summen auf Basis des letzten in dem Patch enthaltenen Datensatzes (**1385**) eingerichtet ist; und
wobei die Hash-Einrichtung ferner zum Berechnen (**400**, **920**–**940**) jeder weiteren Hash-Summe der Vielzahl von Hash-Summen auf Basis eines jeweils nächstletzten in dem Patch enthaltenen Datensatzes und einer jeweils zuletzt berechneten Hash-Summe der Vielzahl von Hash-Summen eingerichtet ist.

5. Patch-Server nach Anspruch 4, wobei die erste Signaturerzeugungseinrichtung ferner zum Erzeugen (**420**, **1010**) der ersten digitalen Signatur auf Basis der zuletzt berechneten Hash-Summe der Vielzahl von Hash-Summen eingerichtet ist; und wobei der Transmitter ferner zum Übertragen des Patchs zusammen mit der ersten und zweiten digitalen Signatur und der Vielzahl von Hash-Summen an den Patch-Client eingerichtet ist.

6. Patch-Server nach Anspruch 4 oder 5, wobei die zweite Signaturerzeugungseinrichtung ferner zum Erzeugen (**420**, **1020**) der zweiten digitalen Signatur auf Basis der zuletzt berechneten Hash-Summe der Vielzahl von Hash-Summen eingerichtet ist; und wobei der Transmitter ferner zum Übertragen des Patchs zusammen mit der ersten und zweiten digitalen Signatur und der Vielzahl von Hash-Summen an den Patch-Client eingerichtet ist.

7. Patch-Server nach einem der Ansprüche 1 bis 6, ferner umfassend:
eine Schlüsselindikator-Erzeugungseinrichtung, die zum Erzeugen eines Schlüsselindikators (**1140**, **1340**) eingerichtet ist, welcher einen ersten Schlüsselindikator (**1210**), der angibt, welcher erste private Schlüssel aus der ersten Gruppe privater Schlüssel ausgewählt wurde, und einen zweiten Schlüsselindikator (**1220**), der angibt, welcher zweite private Schlüssel aus der zweiten Gruppe privater Schlüssel ausgewählt wurde, umfasst;
wobei der Transmitter ferner zum Übertragen des Patchs zusammen mit der ersten und zweiten digitalen Signatur und dem Schlüsselindikator an den Patch-Client eingerichtet ist.

8. Patch-Server nach Anspruch 7, wobei die Schlüsselindikator-Erzeugungseinrichtung ferner zum Erzeugen des Schlüsselindikators, welcher ferner einen Dummy-Indikator (**1240**), der eine der ersten und zweiten digitalen Signaturen als eine Dummy-Signatur identifiziert, umfasst, eingerichtet ist.

9. Patch-Server nach einem der Ansprüche 1 bis 8, ferner umfassend:
eine Sitzungsschlüssel-Erzeugungseinrichtung, die zum Erzeugen (**470**, **810**) eines zufälligen Sitzungsschlüssels eingerichtet ist;
eine erste Verschlüsselungskomponente, die zum Verschlüsseln (**470**, **820**) des Patchs mit dem zufälligen Sitzungsschlüssel unter Benutzung eines symmetrischen Verschlüsselungsalgorithmus eingerichtet ist; und
eine zweite Verschlüsselungskomponente, die zum Verschlüsseln (**470**, **830**) des zufälligen Sitzungsschlüssels mit einem Masterschlüssel eingerichtet ist;
wobei der Transmitter ferner zum Übertragen des Patchs in verschlüsselter Form zusammen mit der ersten und zweiten digitalen Signatur und dem verschlüsselten zufälligen Sitzungsschlüssel an den Patch-Client eingerichtet ist.

10. Patch-Server nach Anspruch 9, wobei die erste Verschlüsselungskomponente ferner zum Verschlüsseln (**470**, **820**) der ersten und zweiten digitalen Signatur mit dem zufälligen Sitzungsschlüssel unter Benutzung des symmetrischen Verschlüsselungsalgorithmus eingerichtet ist; und wobei der Transmitter ferner zum Übertragen des Patchs in verschlüsselter Form zusammen mit der ersten und zweiten digitalen Signatur in verschlüsselter Form und dem verschlüsselten zufälligen Sitzungsschlüssel an den Patch-Client eingerichtet ist.

11. Verfahren zum Bereitstellen eines Patchs (**1150**, **1355**, **1365**, **1375**, **1385**) an einen Patch-Client (**140**), umfassend:
Erzeugen einer ersten Gruppe privater Schlüssel (**260**), welche eine Vielzahl erster privater Schlüssel (**200**–**215**) umfasst, unter Benutzung einer ersten Schlüsselerzeugungsplattform (**110**);
Erzeugen einer zweiten Gruppe privater Schlüssel (**270**), welche eine Vielzahl zweiter privater Schlüssel (**220**–**235**) umfasst, unter Benutzung einer zweiten Schlüsselerzeugungsplattform (**120**), die von der ersten Schlüsselerzeugungsplattform verschieden ist;
Auswählen (**410**, **610**) eines der ersten privaten Schlüssel aus der ersten Gruppe privater Schlüssel;
Auswählen (**410**, **620**) eines der zweiten privaten Schlüssel aus der zweiten Gruppe privater Schlüssel;

Erzeugen (**420**, **710**, **1010**) einer ersten digitalen Signatur (**1110**, **1310**) auf Basis des Patchs und des ersten ausgewählten privaten Schlüssels;
Erzeugen (**420**, **720**, **1020**) einer zweiten digitalen Signatur (**1120**, **1320**) auf Basis des Patchs und des zweiten ausgewählten privaten Schlüssels; und
Übertragen des Patchs zusammen mit der ersten und zweiten digitalen Signatur an den Patch-Client.

12. Verfahren nach Anspruch 11, ferner umfassend:
Berechnen (**400**, **510**) einer ersten Hash-Summe auf Basis des Patchs; und
Berechnen (**400**, **520**) einer zweiten Hash-Summe auf Basis der ersten Hash-Summe;
wobei die Erzeugung der ersten digitalen Signatur eine Erzeugung (**420**, **710**) der ersten digitalen Signatur auf Basis der zweiten Hash-Summe und des ersten ausgewählten privaten Schlüssels umfasst.

13. Verfahren nach Anspruch 12, ferner umfassend:
Berechnen (**400**, **530**) einer dritten Hash-Summe auf Basis der zweiten Hash-Summe;
wobei die Erzeugung der zweiten digitalen Signatur eine Erzeugung (**420**, **720**) der zweiten digitalen Signatur auf Basis der dritten Hash-Summe und des zweiten ausgewählten privaten Schlüssels umfasst.

14. Verfahren nach Anspruch 11, ferner umfassend:
Berechnen (**400**, **910**–**940**) einer Vielzahl von Hash-Summen;
wobei der Patch eine Vielzahl von Datensätzen (**1355**, **1365**, **1375**, **1385**) umfasst;
wobei die Berechnung der Vielzahl von Hash-Summen eine Berechnung (**400**, **910**) einer ersten Hash-Summe der Vielzahl von Hash-Summen auf Basis des letzten in dem Patch enthaltenen Datensatzes (**1385**) umfasst; und
wobei die Berechnung der Vielzahl von Hash-Summen ferner eine Berechnung (**400**, **920**–**940**) jeder weiteren Hash-Summe der Vielzahl von Hash-Summen auf Basis eines jeweils nächstletzten in dem Patch enthaltenen Datensatzes und einer jeweils zuletzt berechneten Hash-Summe der Vielzahl von Hash-Summen umfasst.

15. Verfahren nach Anspruch 14, wobei die Erzeugung der ersten digitalen Signatur eine Erzeugung (**420**, **1010**) der ersten digitalen Signatur auf Basis der zuletzt berechneten Hash-Summe der Vielzahl von Hash-Summen umfasst; und wobei die Übertragung des Patchs eine Übertragung des Patchs zusammen mit der ersten und zweiten digitalen Signatur und der Vielzahl von Hash-Summen an den Patch-Client umfasst.

16. Verfahren nach Anspruch 14 oder 15, wobei die Erzeugung der zweiten digitalen Signatur eine Erzeugung (**420**, **1020**) der zweiten digitalen Signatur auf Basis der zuletzt berechneten Hash-Summe der Vielzahl von Hash-Summen umfasst; und wobei die Übertragung des Patchs eine Übertragung des Patchs zusammen mit der ersten und zweiten digitalen Signatur und der Vielzahl von Hash-Summen an den Patch-Client umfasst.

17. Verfahren nach einem der Ansprüche 11 bis 16, ferner umfassend:
Erzeugen eines Schlüsselindikators (**1140**, **1340**), welcher einen ersten Schlüsselindikator (**1210**), der angibt, welcher erste private Schlüssel aus der ersten Gruppe privater Schlüssel ausgewählt wurde, und einen zweiten Schlüsselindikator (**1220**), der angibt, welcher zweite private Schlüssel aus der zweiten Gruppe privater Schlüssel ausgewählt wurde, umfasst;
wobei die Übertragung des Patchs eine Übertragung des Patchs zusammen mit der ersten und zweiten digitalen Signatur und dem Schlüsselindikator an den Patch-Client umfasst.

18. Verfahren nach Anspruch 17, wobei die Erzeugung des Schlüsselindikators eine Erzeugung des Schlüsselindikators umfasst, welcher ferner einen Dummy-Indikator (**1240**) umfasst, der eine der ersten und zweiten digitalen Signaturen als eine Dummy-Signatur identifiziert.

19. Verfahren nach einem der Ansprüche 11 bis 18, ferner umfassend:
Erzeugen (**470**, **810**) eines zufälligen Sitzungsschlüssels;
Verschlüsseln (**470**, **820**) des Patchs mit dem zufälligen Sitzungsschlüssel unter Benutzung eines symmetrischen Verschlüsselungsalgorithmus; und
Verschlüsseln (**470**, **830**) des zufälligen Sitzungsschlüssels mit einem Masterschlüssel;
wobei die Übertragung des Patchs eine Übertragung des Patchs in verschlüsselter Form zusammen mit der ersten und zweiten digitalen Signatur und dem verschlüsselten zufälligen Sitzungsschlüssel an den Patch-Client umfasst.

20. Verfahren nach Anspruch 19, ferner umfassend:
Verschlüsseln (**470**, **820**) der ersten und zweiten digitalen Signatur mit dem zufälligen Sitzungsschlüssel unter Benutzung des symmetrischen Verschlüsselungsalgorithmus;
wobei die Übertragung des Patchs eine Übertragung des Patchs in verschlüsselter Form zusammen mit der ersten und zweiten digitalen Signatur in verschlüsselter Form und dem verschlüsselten zufälligen Sitzungsschlüssel an den Patch-Client umfasst.

21. Patch-Client (**140**), der mit einem Patch-Server (**100**) zum Empfangen (**1500**, **1800**) eines Patchs

(**1150**, **1355**, **1365**, **1375**, **1385**) von dem Patch-Server verbunden ist und umfasst:
eine erste Speichereinrichtung, welche eine erste Gruppe öffentlicher Schlüssel (**360**) speichert, welche eine Vielzahl erster öffentlicher Schlüssel (**300**–**315**) umfasst, welche durch eine erste Schlüsselerzeugungsplattform (**110**) erzeugt wurden;
eine zweite Speichereinrichtung, welche eine zweite Gruppe öffentlicher Schlüssel (**370**) speichert, welche eine Vielzahl zweiter öffentlicher Schlüssel (**320**–**335**) umfasst, welche durch eine zweite Schlüsselerzeugungsplattform (**120**), die von der ersten Schlüsselerzeugungsplattform verschieden ist, erzeugt wurden;
eine erste Schlüsselauswahleinrichtung, die zum Auswählen (**1530**, **1610**, **1830**) eines der ersten öffentlichen Schlüssel aus der ersten Gruppe öffentlicher Schlüssel eingerichtet ist;
eine zweite Schlüsselauswahleinrichtung, die zum Auswählen (**1530**, **1620**, **1830**) eines der zweiten öffentlichen Schlüssel aus der Gruppe der öffentlichen Schlüssel eingerichtet ist;
eine erste Signaturüberprüfungskomponente, die zum Überprüfen (**1540**, **1720**, **1750**, **1840**, **1920**, **1950**) einer ersten digitalen Signatur (**1110**, **1310**), welche von dem Patch-Server zusammen mit dem Patch empfangen wurde, unter Benutzung des ersten ausgewählten öffentlichen Schlüssels eingerichtet ist; und
eine zweite Signaturüberprüfungskomponente, die zum Überprüfen (**1540**, **1730**, **1750**, **1840**, **1930**, **1960**) einer zweiten digitalen Signatur (**1120**, **1320**), die von dem Patch-Server zusammen mit dem Patch empfangen wurde, unter Benutzung des zweiten ausgewählten öffentlichen Schlüssels eingerichtet ist;
wobei der Patch-Client dazu eingerichtet ist, den Patch nur dann zu installieren (**1540**, **2045**), wenn die Ergebnisse der Überprüfung der ersten und zweiten digitalen Signatur Authentizität und Integrität der ersten bzw. digitalen Signatur angeben.

22. Patch-Client nach Anspruch 21, ferner umfassend:
eine erste Hash-Einrichtung, die zum Berechnen (**1700**, **510**) einer ersten Hash-Summe auf Basis des Patchs eingerichtet ist; und
eine zweite Hash-Einrichtung, die zum Berechnen (**1700**, **520**) einer zweiten Hash-Summe auf Basis der ersten Hash-Summe eingerichtet ist;
wobei die erste Signaturüberprüfungskomponente ferner zum Überprüfen (**1540**, **1750**) der ersten digitalen Signatur auf Basis der zweiten Hash-Summe eingerichtet ist.

23. Patch-Client nach Anspruch 22, ferner umfassend:
eine dritte Hash-Einrichtung, die zum Berechnen (**1700**, **530**) einer dritten Hash-Summe auf Basis der zweiten Hash-Summe eingerichtet ist;
wobei die zweite Signaturüberprüfungskomponente ferner zum Überprüfen (**1540**, **1760**) der zweiten digitalen Signatur auf Basis der dritten Hash-Summe eingerichtet ist.

24. Patch-Client nach Anspruch 21, wobei die erste Signaturüberprüfungskomponente ferner zum Überprüfen (**1840**, **1950**) der ersten digitalen Signatur auf Basis einer Hash-Summe (**1350**), die von dem Patch-Server zusammen mit dem Patch empfangen wurde, eingerichtet ist; und wobei die zweite Signaturüberprüfungskomponente ferner zum Überprüfen (**1840**, **1960**) der zweiten digitalen Signatur auf Basis der ersten Hash-Summe eingerichtet ist.

25. Patch-Client nach einem der Ansprüche 21 bis 24, wobei die erste Schlüsselauswahleinrichtung ferner zum Auswählen des einen der ersten öffentlichen Schlüssel gemäß einem ersten Schlüsselindikator (**1210**), der von dem Patch-Server zusammen mit dem Patch empfangen wurde, eingerichtet ist; und wobei die zweite Schlüsselauswahleinrichtung ferner zum Auswählen des einen der zweiten öffentlichen Schlüssel gemäß einem zweiten Schlüsselindikator (**1220**), der von dem Patch-Server zusammen mit dem Patch empfangen wurde, eingerichtet ist.

26. Patch-Client nach einem der Ansprüche 21 bis 25, der ferner zum Außerachtlassen (**1790**, **1990**) des Ergebnisses der Überprüfung der ersten oder zweiten digitalen Signatur, wenn ein Dummy-Indikator (**1240**), der von dem Patch-Server zusammen mit dem Patch empfangen wurde, die erste bzw. zweite digitale Signatur als eine Dummy-Signatur identifiziert, eingerichtet ist.

27. Patch-Client nach einem der Ansprüche 21 bis 26, ferner umfassend:
eine dritte Speichereinrichtung, welche einen Masterschlüssel (**160**) speichert;
eine erste Entschlüsselungskomponente, die zum Entschlüsseln (**1510**, **1810**) eines verschlüsselten zufälligen Sitzungsschlüssels (**1410**), der von dem Patch-Server zusammen mit dem Patch empfangen wurde, unter Benutzung des Masterschlüssels eingerichtet ist, um den zufälligen Sitzungsschlüssel zu erhalten;
eine zweite Entschlüsselungskomponente, die zum Entschlüsseln (**1520**, **1850**, **1900**, **2005**) des Patchs durch Anwendung eines symmetrischen Entschlüsselungsalgorithmus unter Benutzung des zufälligen Sitzungsschlüssels eingerichtet ist.

28. Patch-Client nach Anspruch 27, wobei die zweite Entschlüsselungskomponente ferner zum Entschlüsseln (**1520**, **1820**) der ersten und zweiten digitalen Signatur durch Anwendung des symmetrischen Entschlüsselungsalgorithmus auf Basis des zufälligen Sitzungsschlüssels eingerichtet ist.

29. Patch-Client nach Anspruch 27 oder 28, wobei der Masterschlüssel in der dritten Speichereinrichtung versteckt unter anderen in der dritten Speichereinrichtung gespeicherten Informationen gespeichert ist.

30. Patch-Client nach einem der Ansprüche 27 bis 29, wobei der Masterschlüssel während des Produktionsprozesses des Patch-Clients in die dritte Speichereinrichtung eingegeben wurde.

31. Patch-Client nach einem der Ansprüche 21 bis 30, wobei die erste Gruppe öffentlicher Schlüssel und die zweite Gruppe öffentlicher Schlüssel während des Produktionsprozesses des Patch-Clients in die erste bzw. zweite Speichereinrichtung eingegeben wurden.

32. Verfahren zum Installieren eines Patchs (**1150**, **1355**, **1365**, **1375**, **1385**) in einem Patch-Client (Korrekturclient) (**140**), umfassend:
Empfangen (**1500**, **1800**) des Patchs zusammen mit einer ersten digitalen Signatur (**1110**, **1310**) und einer zweiten digitalen Signatur (**1120**, **1320**) von einem Patch-Server (**100**), der mit dem Patch-Client verbunden ist;
Speichern einer ersten Gruppe öffentlicher Schlüssel (**360**), welche eine Vielzahl erster öffentlicher Schlüssel (**300**–**315**) umfasst, in dem Patch-Client, wobei die ersten öffentlichen Schlüssel durch eine erste Schlüsselerzeugungsplattform (**110**) erzeugt wurden;
Speichern einer zweiten Gruppe öffentlicher Schlüssel (**370**), welche eine Vielzahl zweiter öffentlicher Schlüssel (**320**–**335**) umfasst, in dem Patch-Client, wobei die zweiten öffentlichen Schlüssel durch eine zweite Schlüsselerzeugungsplattform (**120**) erzeugt wurden, die von der ersten Schlüsselerzeugungsplattform verschieden ist;
Auswählen (**1530**, **1610**, **1830**) eines der ersten öffentlichen Schlüssel aus der ersten Gruppe öffentlicher Schlüssel;
Auswählen (**1530**, **1620**, **1830**) eines der zweiten öffentlichen Schlüssel aus der zweiten Gruppe öffentlicher Schlüssel;
Überprüfen (**1540**, **1720**, **1750**, **1840**, **1920**, **1950**) der ersten digitalen Signatur unter Benutzung des ersten ausgewählten öffentlichen Schlüssels;
Überprüfen (**1540**, **1730**, **1760**, **1840**, **1930**, **1960**) der zweiten digitalen Signatur unter Benutzung des zweiten ausgewählten öffentlichen Schlüssels; und
Installieren (**1560**, **2045**) des Patchs in dem Patch-Client nur dann, wenn die Ergebnisse der Überprüfung der ersten und zweiten digitalen Signatur Authentizität und Integrität der ersten bzw. zweiten digitalen Signatur angeben.

33. Verfahren nach Anspruch 32, ferner umfassend:
Berechnen (**1700**, **510**) einer ersten Hash-Summe auf Basis des Patchs; und
Berechnen (**1700**, **520**) einer zweiten Hash-Summe auf Basis der ersten Hash-Summe;
wobei die Überprüfung der ersten digitalen Signatur eine Überprüfung (**1540**, **1750**) der ersten digitalen Signatur auf Basis der zweiten Hash-Summe umfasst.

34. Verfahren nach Anspruch 33, ferner umfassend:
Berechnen (**1700**, **530**) einer dritten Hash-Summe auf Basis der zweiten Hash-Summe;
wobei die Überprüfung der zweiten digitalen Signatur eine Überprüfung (**1540**, **1760**) der zweiten digitalen Signatur auf Basis der dritten Hash-Summe umfasst.

35. Verfahren nach Anspruch 32, ferner umfassend:
Empfangen (**1500**, **1800**) einer ersten Hash-Summe (**1350**) von dem Patch-Server zusammen mit dem Patch;
wobei die Überprüfung der ersten digitalen Signatur eine Überprüfung (**1840**, **1950**) der ersten digitalen Signatur auf Basis der ersten Hash-Summe umfasst; und
wobei die Überprüfung der zweiten digitalen Signatur eine Überprüfung (**1840**, **1960**) der zweiten digitalen Signatur auf Basis der ersten Hash-Summe umfasst.

36. Verfahren nach einem der Ansprüche 32 bis 35, ferner umfassend:
Empfangen (**1500**, **1800**) eines ersten Schlüsselindikators (**1210**) und eines zweiten Schlüsselindikators (**1220**) von dem Patch-Server zusammen mit dem Patch;
wobei die Auswahl des einen der ersten öffentlichen Schlüssel eine Auswahl des einen der ersten öffentlichen Schlüssel gemäß dem ersten Schlüsselindikator umfasst;
wobei die Auswahl des einen der zweiten öffentlichen Schlüssel eine Auswahl des einen der zweiten öffentlichen Schlüssel gemäß dem zweiten Schlüsselindikator umfasst.

37. Verfahren nach einem der Ansprüche 32 bis 36, ferner umfassend:
Empfangen (**1500**, **1800**) eines Dummy-Indikators (**1240**) von dem Patch-Server zusammen mit dem Patch; und
Außerachtlassen (**1790**, **1990**) des Ergebnisses der Überprüfung der ersten oder zweiten digitalen Signatur, wenn der Dummy-Indikator die erste bzw. zweite digitale Signatur als eine Dummy-Signatur identifiziert.

38. Verfahren nach einem der Ansprüche 32 bis 37, ferner umfassend:
Empfangen (**1500**, **1800**) eines verschlüsselten zufälligen Sitzungsschlüssels (**1410**) von dem Patch-Server zusammen mit dem Patch;

Speichern eines Masterschlüssels (**160**) in dem Patch-Client;
Entschlüsseln (**1510**, **1810**) des verschlüsselten zufälligen Sitzungsschlüssels unter Benutzung des Masterschlüssels, um den zufälligen Sitzungsschlüssel zu erhalten; und
Entschlüsseln (**1520**, **1850**, **1900**, **2005**) des Patchs durch Anwendung eines symmetrischen Entschlüsselungsalgorithmus unter Benutzung des zufälligen Sitzungsschlüssels.

39. Verfahren nach Anspruch 38, ferner umfassend:
Entschlüsseln (**1520**, **1820**) der ersten und zweiten digitalen Signatur durch Anwendung des symmetrischen Entschlüsselungsalgorithmus unter Benutzung des zufälligen Sitzungsschlüssels.

40. Verfahren nach Anspruch 38 oder 39, wobei die Speicherung des Masterschlüssels in dem Patch-Client eine Speicherung des Masterschlüssels versteckt unter anderen in dem Patch-Client gespeicherten Informationen umfasst.

41. Verfahren nach einem der Ansprüche 38 bis 40, ferner umfassend:
Eingeben des Masterschlüssels in den Patch-Client während des Produktionsprozesses des Patch-Clients.

42. Verfahren nach einem der Ansprüche 32 bis 41, ferner umfassend:
Eingeben der ersten Gruppe öffentlicher Schlüssel und der zweiten Gruppe öffentlicher Schlüssel in den Patch-Client während des Produktionsprozesses des Patch-Clients.

Es folgen 13 Blatt Zeichnungen

Anhängende Zeichnungen

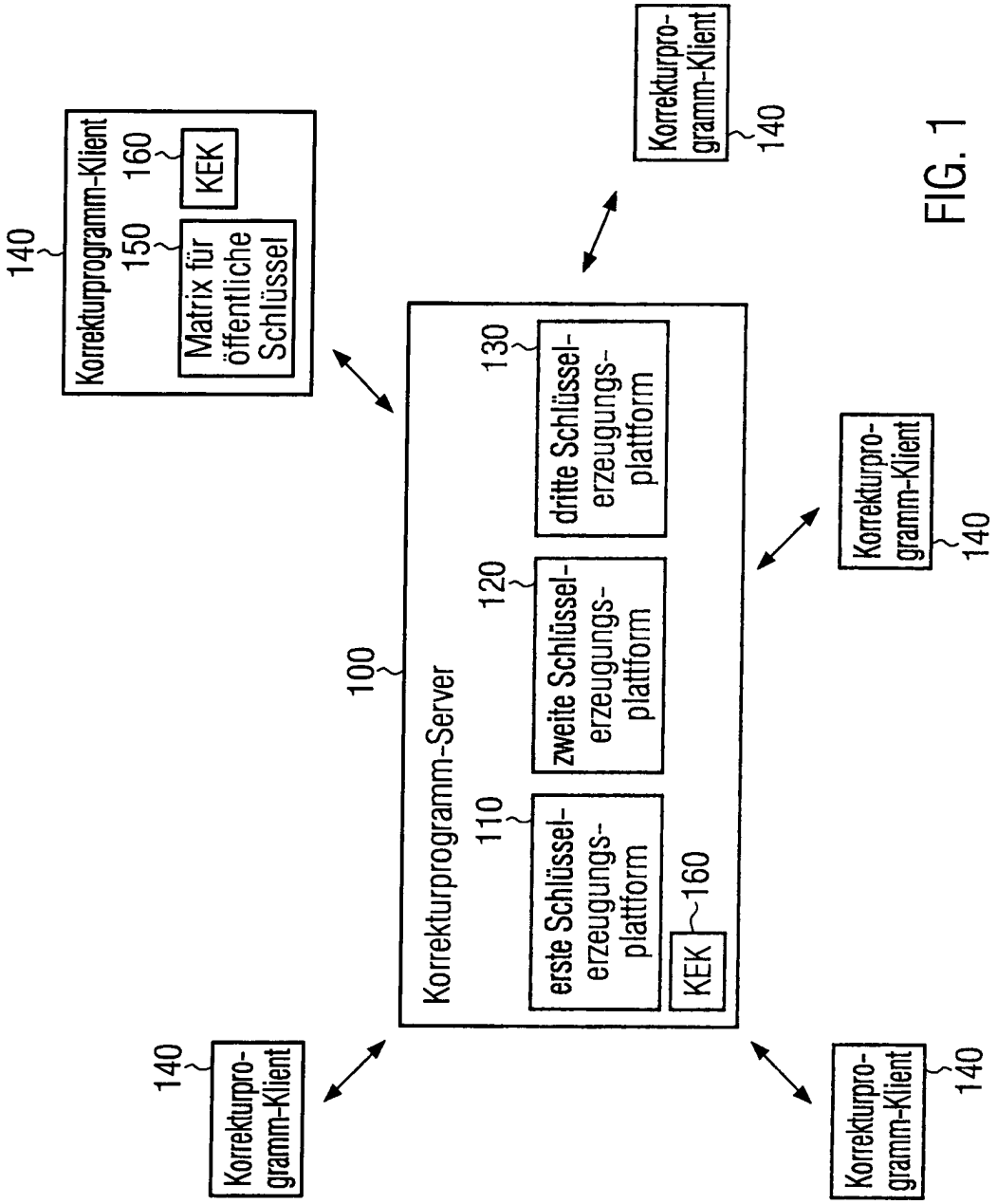


FIG. 1

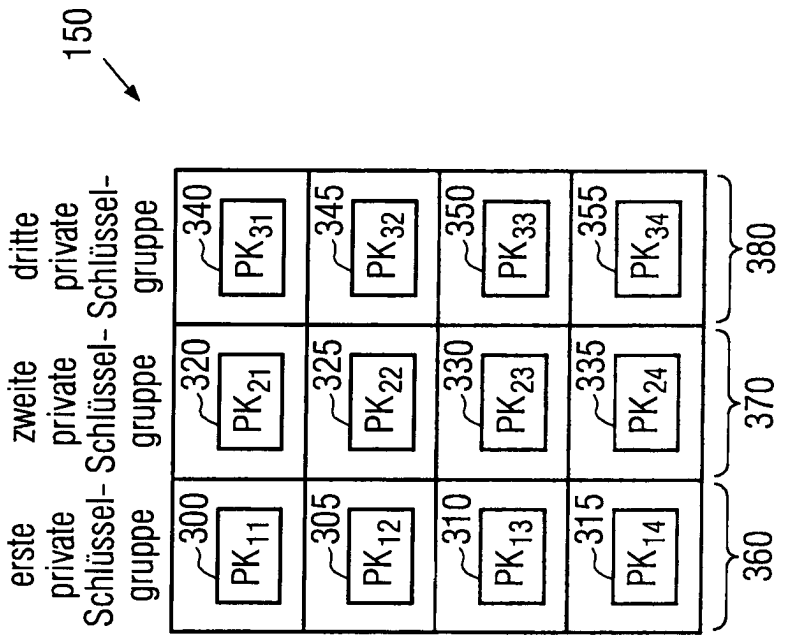

150
erste private Schlüssel-gruppe
zweite private Schlüssel-gruppe
dritte private Schlüssel-gruppe
300
PK11
320
PK21
340
PK31
305
PK12
325
PK22
345
PK32
310
PK13
330
PK23
350
PK33
315
PK14
335
PK24
355
PK34
360
370
380

FIG. 3

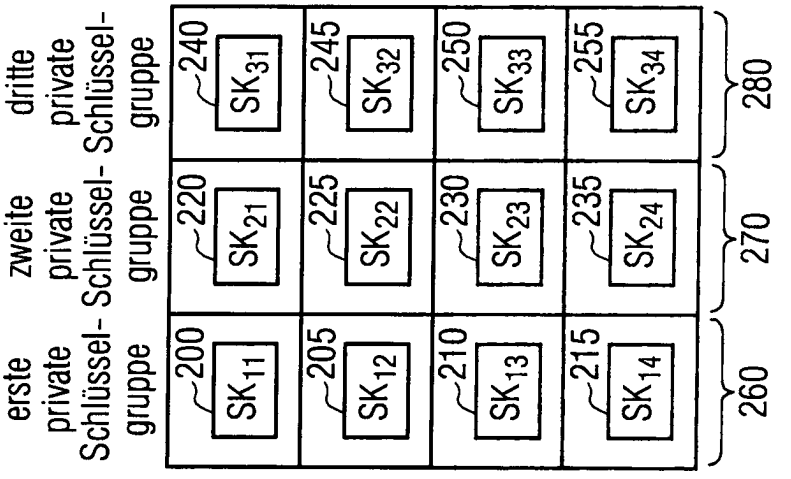

erste private Schlüssel-gruppe
zweite private Schlüssel-gruppe
dritte private Schlüssel-gruppe
200
SK11
220
SK21
240
SK31
205
SK12
225
SK22
245
SK32
210
SK13
230
SK23
250
SK33
215
SK14
235
SK24
255
SK34
260
270
280

FIG. 2

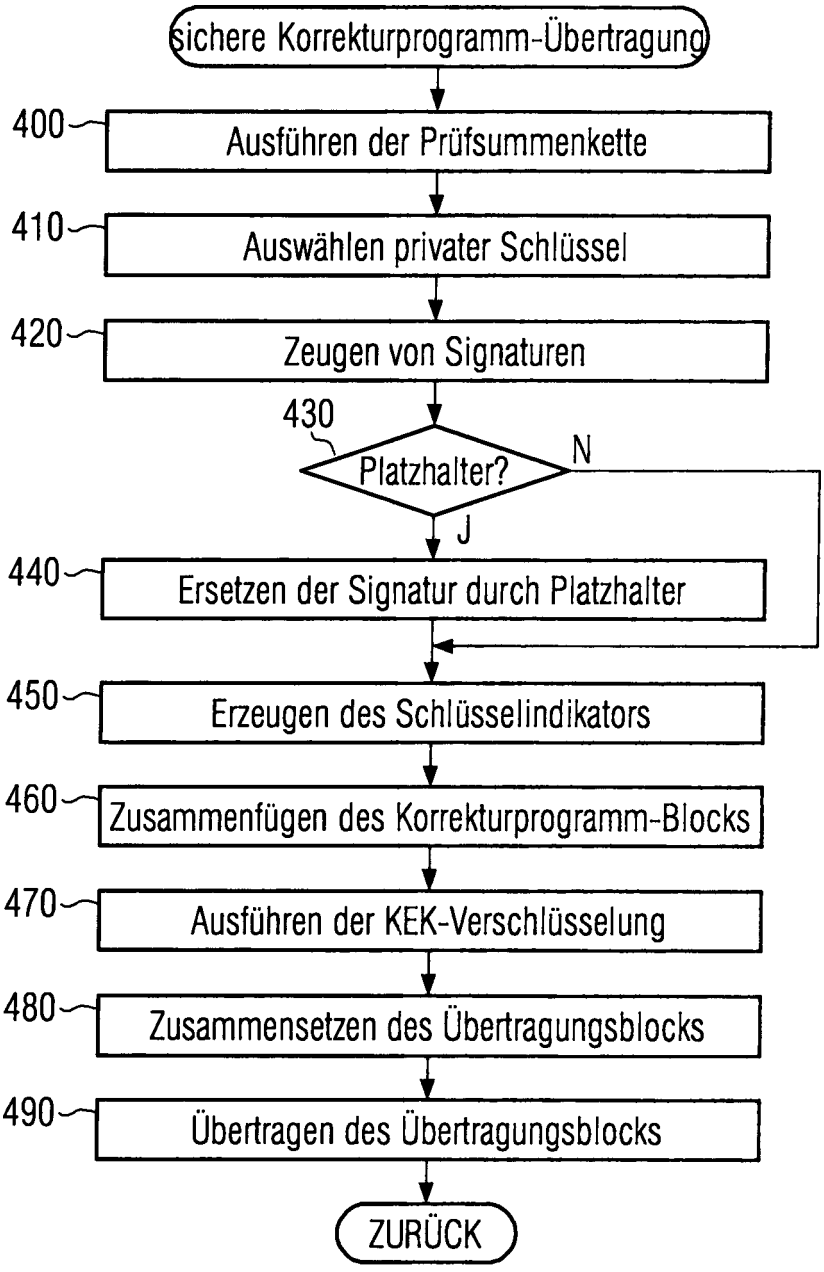

sichere Korrekturprogramm-Übertragung
400
Ausführen der Prüfsummenkette
410
Auswählen privater Schlüssel
420
Zeugen von Signaturen
430
Platzhalter?
N
J
440
Ersetzen der Signatur durch Platzhalter
450
Erzeugen des Schlüsselindikators
460
Zusammenfügen des Korrekturprogramm-Blocks
470
Ausführen der KEK-Verschlüsselung
480
Zusammensetzen des Übertragungsblocks
490
Übertragen des Übertragungsblocks
ZURÜCK

FIG. 4

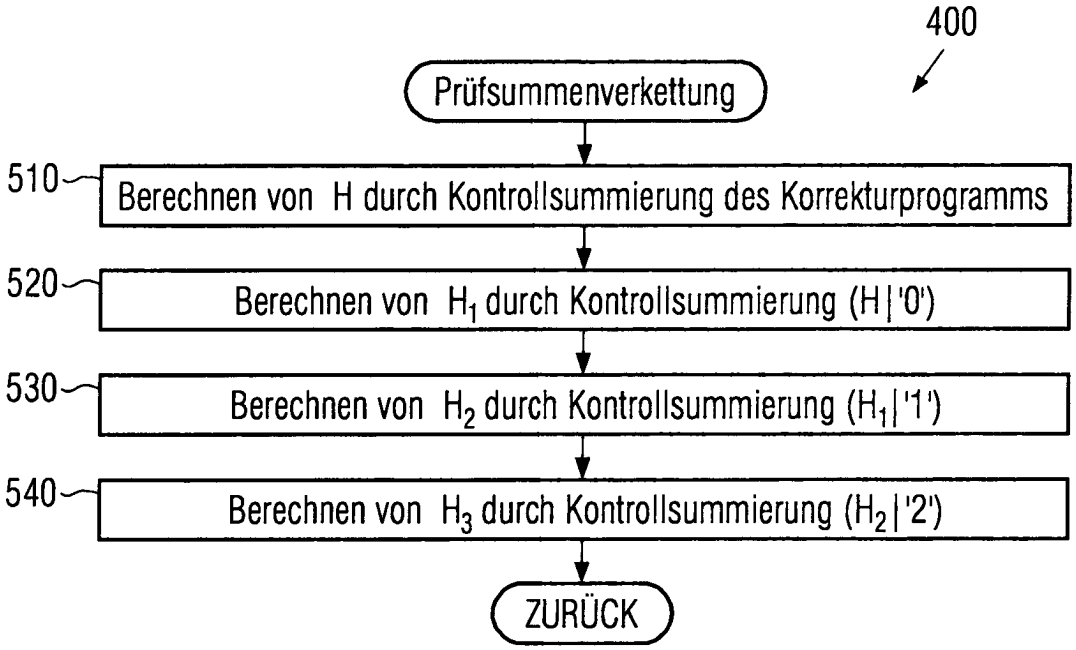

400
Prüfsummenverkettung
510
Berechnen von H durch Kontrollsummierung des Korrekturprogramms
520
Berechnen von H_1 durch Kontrollsummierung (H|'0')
530
Berechnen von H_2 durch Kontrollsummierung (H_1|'1')
540
Berechnen von H_3 durch Kontrollsummierung (H_2|'2')
ZURÜCK

FIG. 5

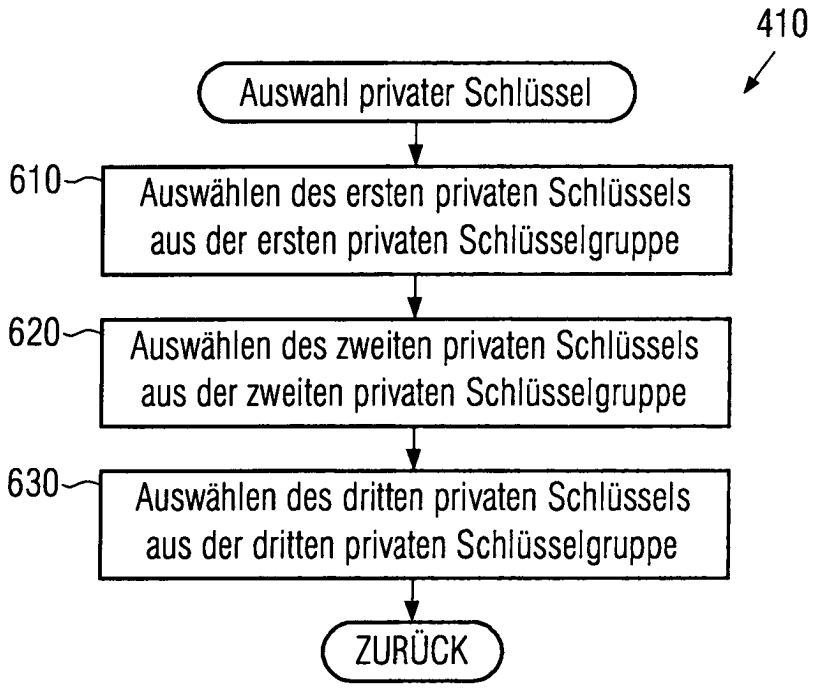

410
Auswahl privater Schlüssel
610
Auswählen des ersten privaten Schlüssels aus der ersten privaten Schlüsselgruppe
620
Auswählen des zweiten privaten Schlüssels aus der zweiten privaten Schlüsselgruppe
630
Auswählen des dritten privaten Schlüssels aus der dritten privaten Schlüsselgruppe
ZURÜCK

FIG. 6

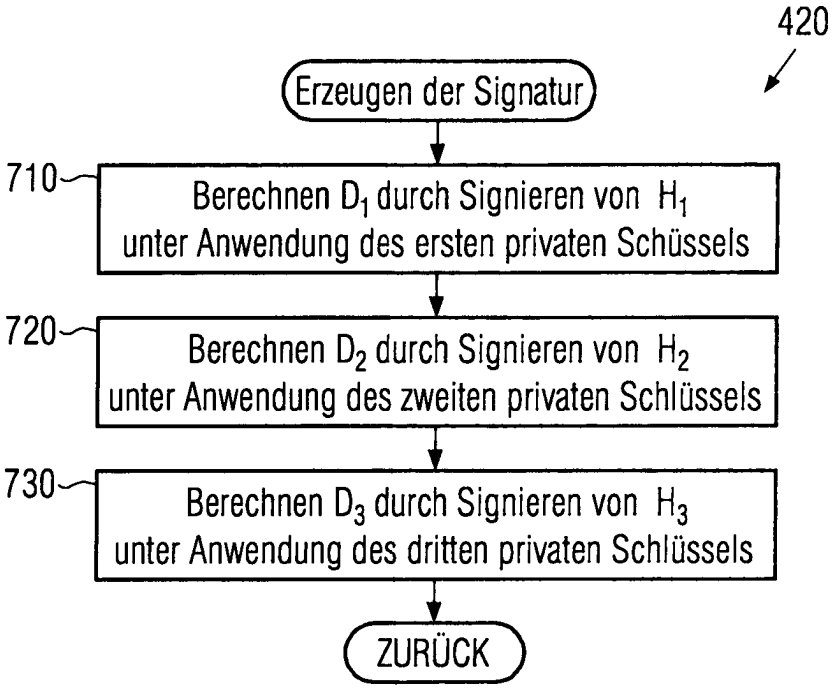

420
Erzeugen der Signatur
710
Berechnen D_1 durch Signieren von H_1 unter Anwendung des ersten privaten Schüssels
720
Berechnen D_2 durch Signieren von H_2 unter Anwendung des zweiten privaten Schlüssels
730
Berechnen D_3 durch Signieren von H_3 unter Anwendung des dritten privaten Schlüssels
ZURÜCK

FIG. 7

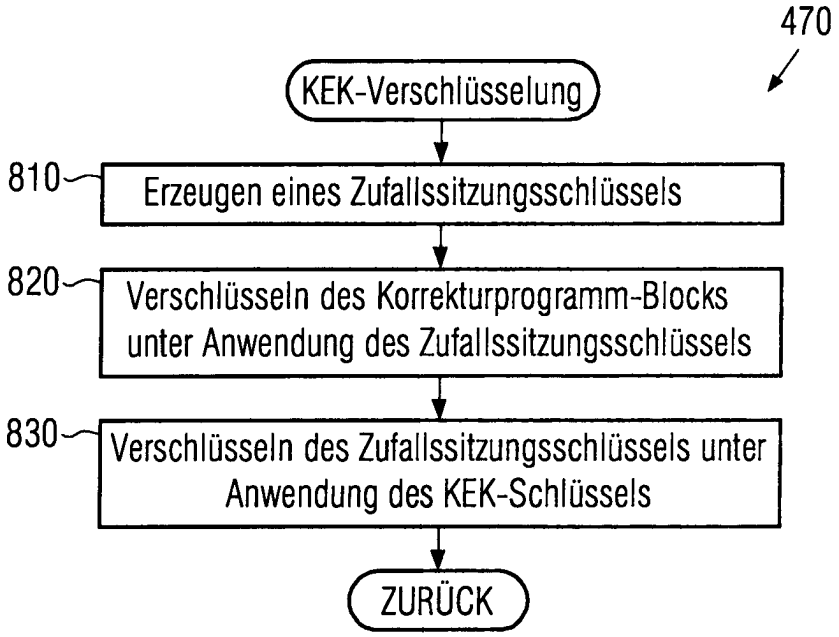

470
KEK-Verschlüsselung
810
Erzeugen eines Zufallssitzungsschlüssels
820
Verschlüsseln des Korrekturprogramm-Blocks unter Anwendung des Zufallssitzungsschlüssels
830
Verschlüsseln des Zufallssitzungsschlüssels unter Anwendung des KEK-Schlüssels
ZURÜCK

FIG. 8

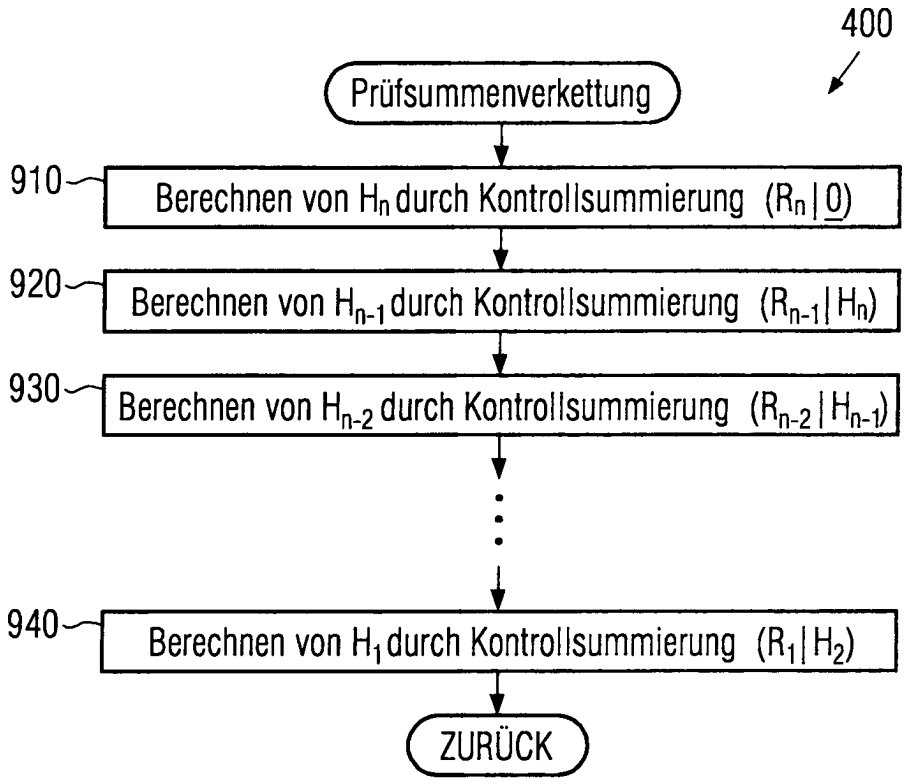

400
Prüfsummenverkettung
910
Berechnen von H_n durch Kontrollsummierung $(R_n | 0)$
920
Berechnen von H_{n-1} durch Kontrollsummierung $(R_{n-1} | H_n)$
930
Berechnen von H_{n-2} durch Kontrollsummierung $(R_{n-2} | H_{n-1})$
940
Berechnen von H_1 durch Kontrollsummierung $(R_1 | H_2)$
ZURÜCK

FIG. 9

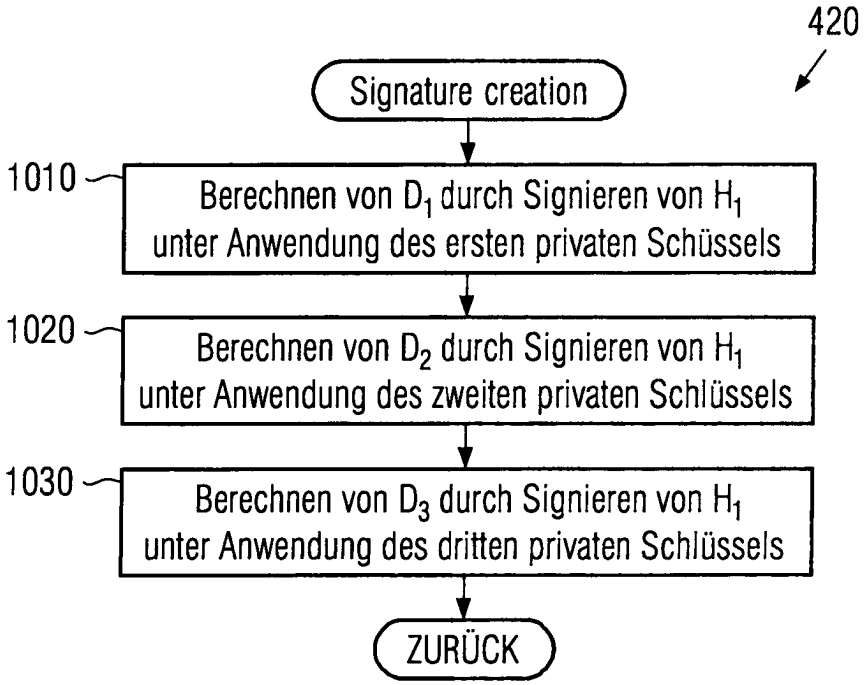

420
Signature creation
1010
Berechnen von D_1 durch Signieren von H_1 unter Anwendung des ersten privaten Schüssels
1020
Berechnen von D_2 durch Signieren von H_1 unter Anwendung des zweiten privaten Schlüssels
1030
Berechnen von D_3 durch Signieren von H_1 unter Anwendung des dritten privaten Schlüssels
ZURÜCK

FIG. 10

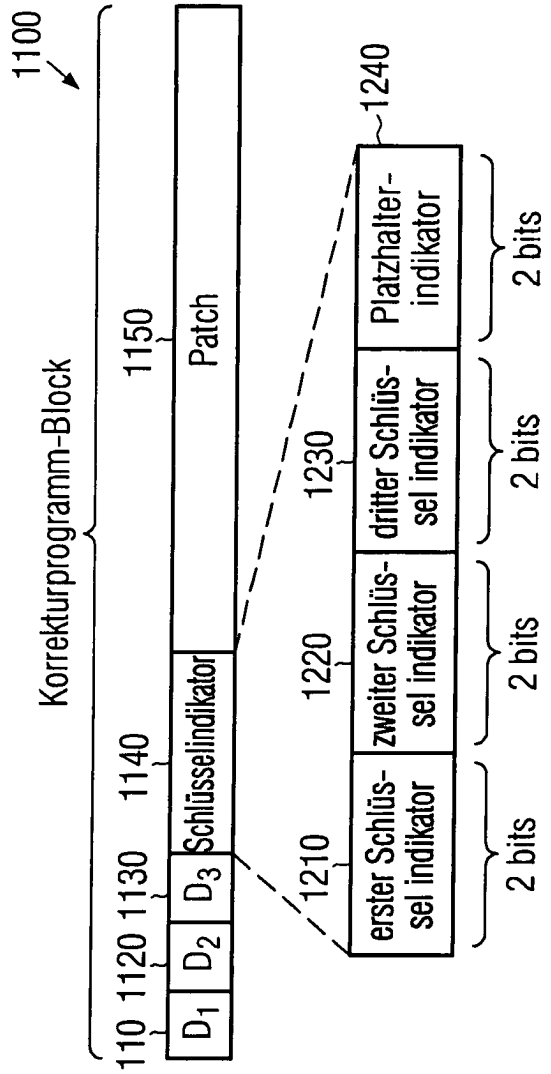


FIG. 11

FIG. 12

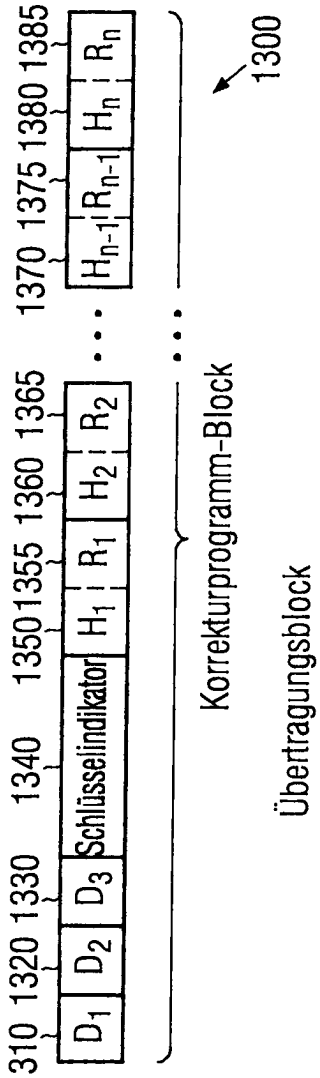


FIG. 13

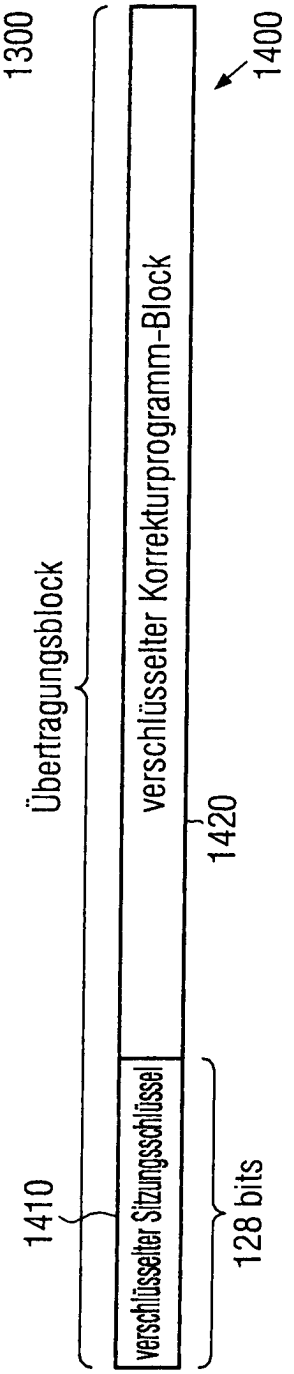


FIG. 14

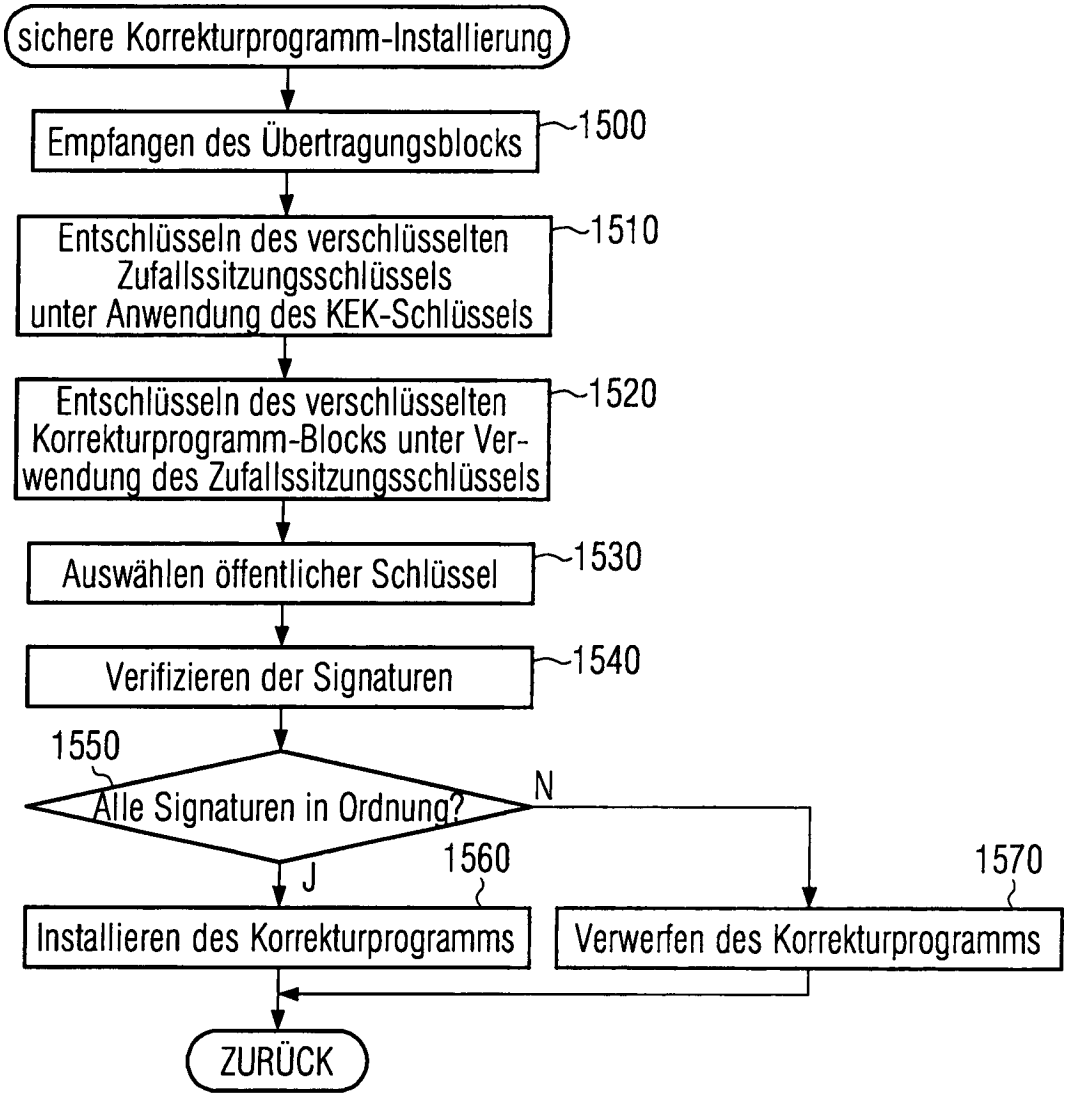

sichere Korrekturprogramm-Installierung
Empfangen des Übertragungsblocks
1500
Entschlüsseln des verschlüsselten Zufallssitzungsschlüssels unter Anwendung des KEK-Schlüssels
1510
Entschlüsseln des verschlüsselten Korrekturprogramm-Blocks unter Verwendung des Zufallssitzungsschlüssels
1520
Auswählen öffentlicher Schlüssel
1530
Verifizieren der Signaturen
1540
1550
Alle Signaturen in Ordnung?
N
J
1560
1570
Installieren des Korrekturprogramms
Verwerfen des Korrekturprogramms
ZURÜCK

FIG. 15

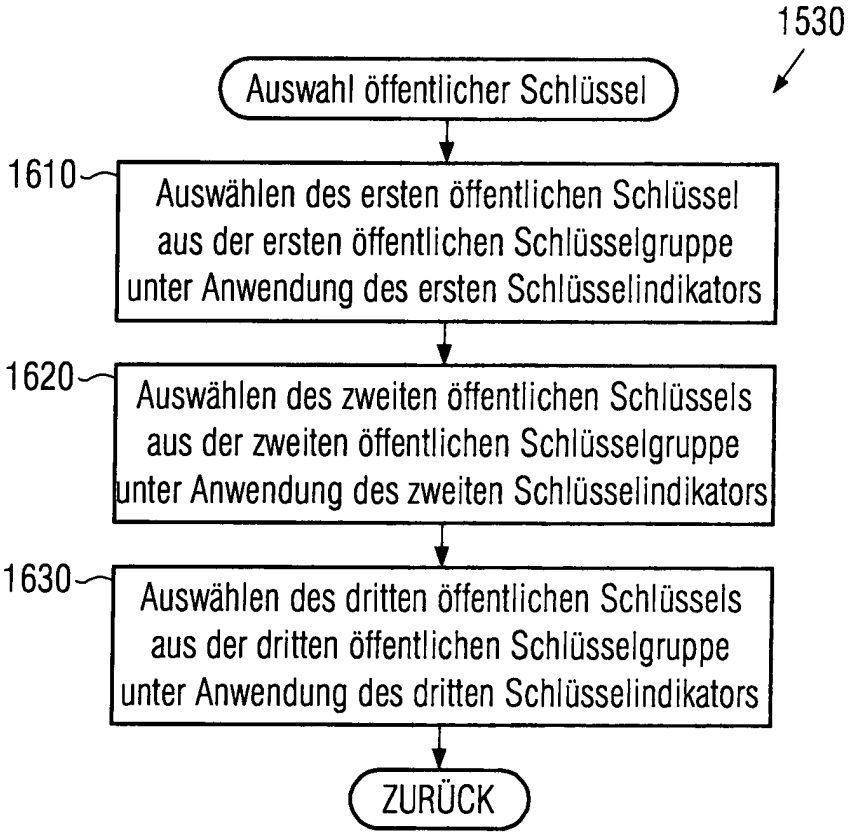

1530
Auswahl öffentlicher Schlüssel
1610
Auswählen des ersten öffentlichen Schlüssel aus der ersten öffentlichen Schlüsselgruppe unter Anwendung des ersten Schlüsselindikators
1620
Auswählen des zweiten öffentlichen Schlüssels aus der zweiten öffentlichen Schlüsselgruppe unter Anwendung des zweiten Schlüsselindikators
1630
Auswählen des dritten öffentlichen Schlüssels aus der dritten öffentlichen Schlüsselgruppe unter Anwendung des dritten Schlüsselindikators
ZURÜCK

FIG. 16

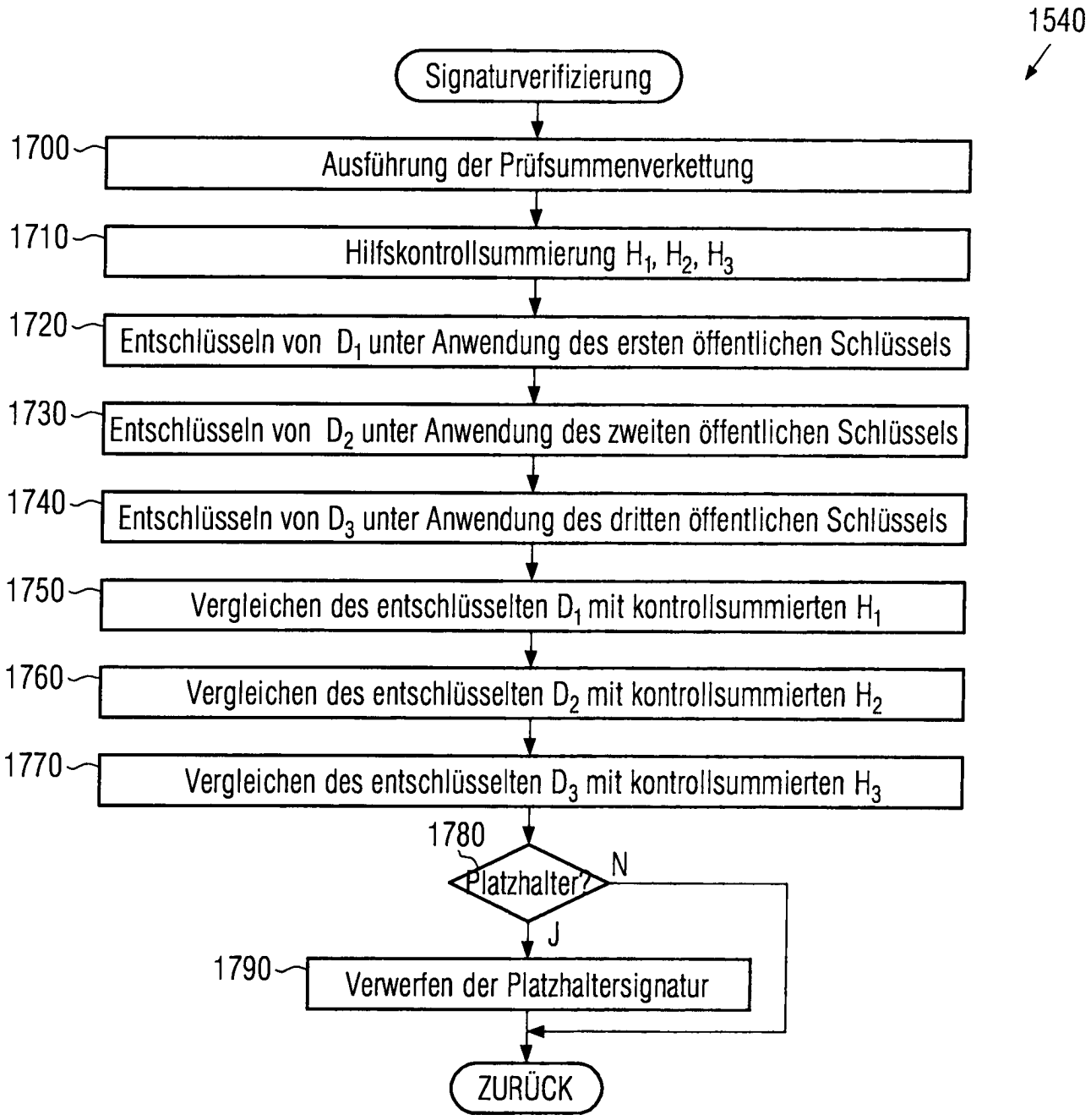

1540
Signaturverifizierung
1700
Ausführung der Prüfsummenverkettung
1710
Hilfskontrollsummierung H_1, H_2, H_3
1720
Entschlüsseln von D_1 unter Anwendung des ersten öffentlichen Schlüssels
1730
Entschlüsseln von D_2 unter Anwendung des zweiten öffentlichen Schlüssels
1740
Entschlüsseln von D_3 unter Anwendung des dritten öffentlichen Schlüssels
1750
Vergleichen des entschlüsselten D_1 mit kontrollsummierten H_1
1760
Vergleichen des entschlüsselten D_2 mit kontrollsummierten H_2
1770
Vergleichen des entschlüsselten D_3 mit kontrollsummierten H_3
1780
Platzhalter?
N
J
1790
Verwerfen der Platzhaltersignatur
ZURÜCK

FIG. 17

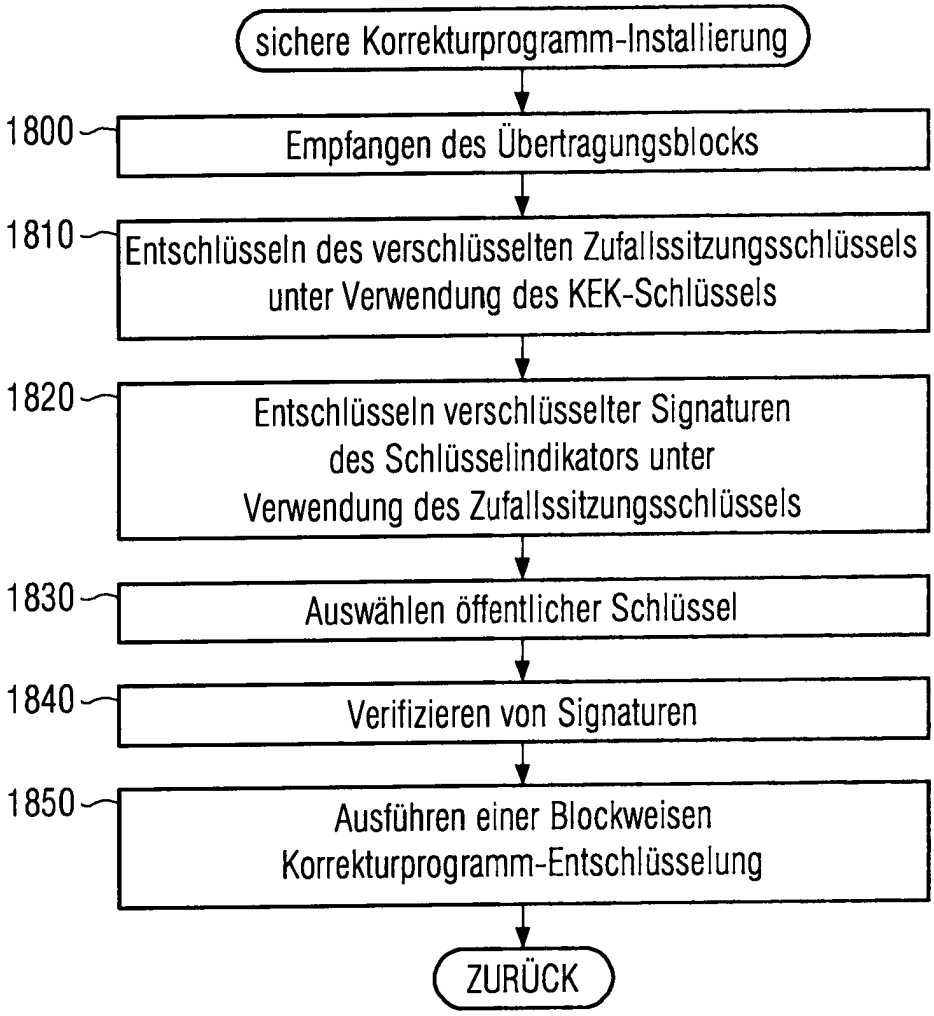

sichere Korrekturprogramm-Installierung
1800
Empfangen des Übertragungsblocks
1810
Entschlüsseln des verschlüsselten Zufallssitzungsschlüssels unter Verwendung des KEK-Schlüssels
1820
Entschlüsseln verschlüsselter Signaturen des Schlüsselindikators unter Verwendung des Zufallssitzungsschlüssels
1830
Auswählen öffentlicher Schlüssel
1840
Verifizieren von Signaturen
1850
Ausführen einer Blockweisen Korrekturprogramm-Entschlüsselung
ZURÜCK

FIG. 18

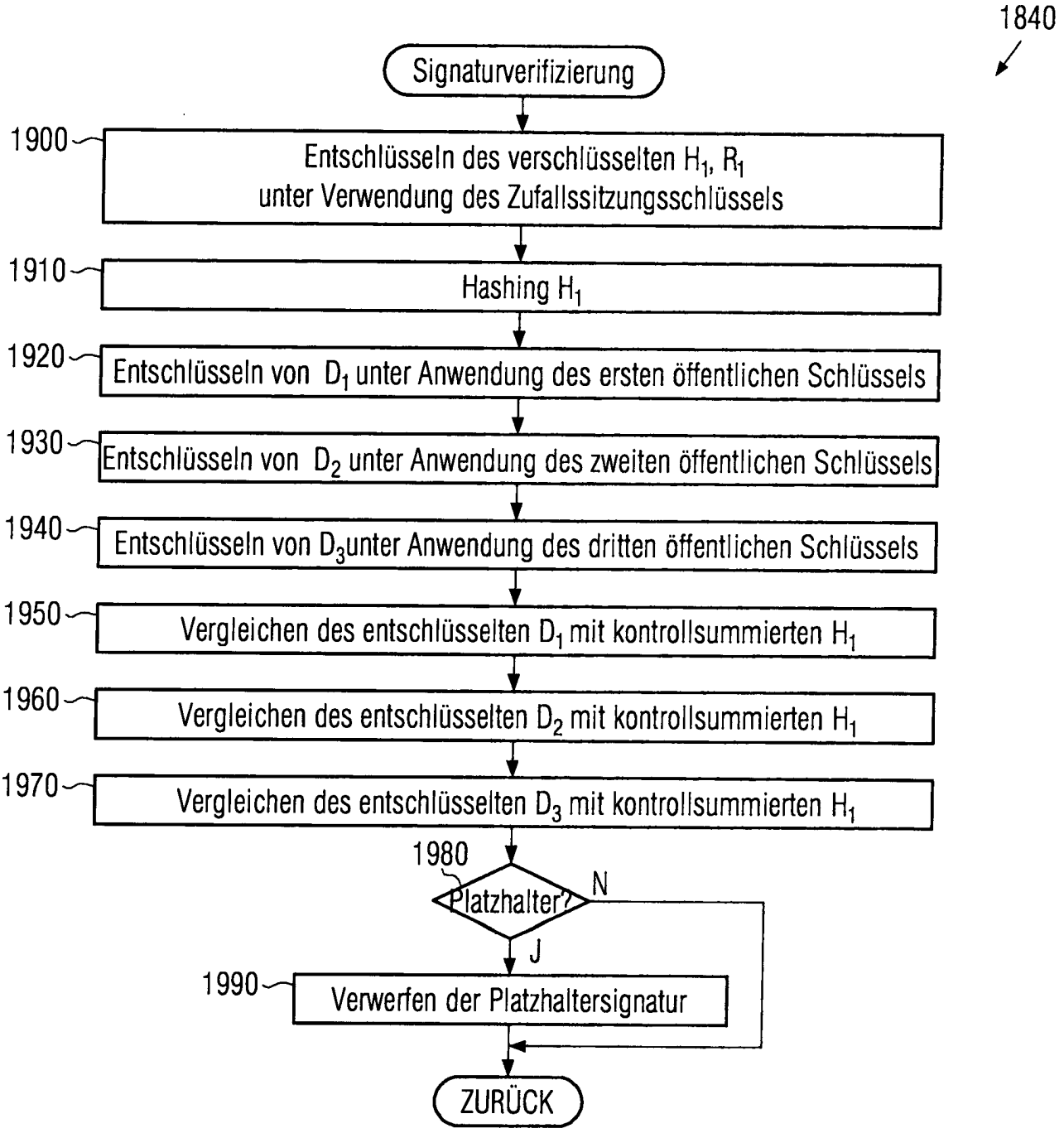

1840
Signaturverifizierung
1900
Entschlüsseln des verschlüsselten H_1, R_1 unter Verwendung des Zufallssitzungsschlüssels
1910
Hashing H_1
1920
Entschlüsseln von D_1 unter Anwendung des ersten öffentlichen Schlüssels
1930
Entschlüsseln von D_2 unter Anwendung des zweiten öffentlichen Schlüssels
1940
Entschlüsseln von D_3unter Anwendung des dritten öffentlichen Schlüssels
1950
Vergleichen des entschlüsselten D_1 mit kontrollsummierten H_1
1960
Vergleichen des entschlüsselten D_2 mit kontrollsummierten H_1
1970
Vergleichen des entschlüsselten D_3 mit kontrollsummierten H_1
1980
Platzhalter?
N
J
1990
Verwerfen der Platzhaltersignatur
ZURÜCK

FIG. 19

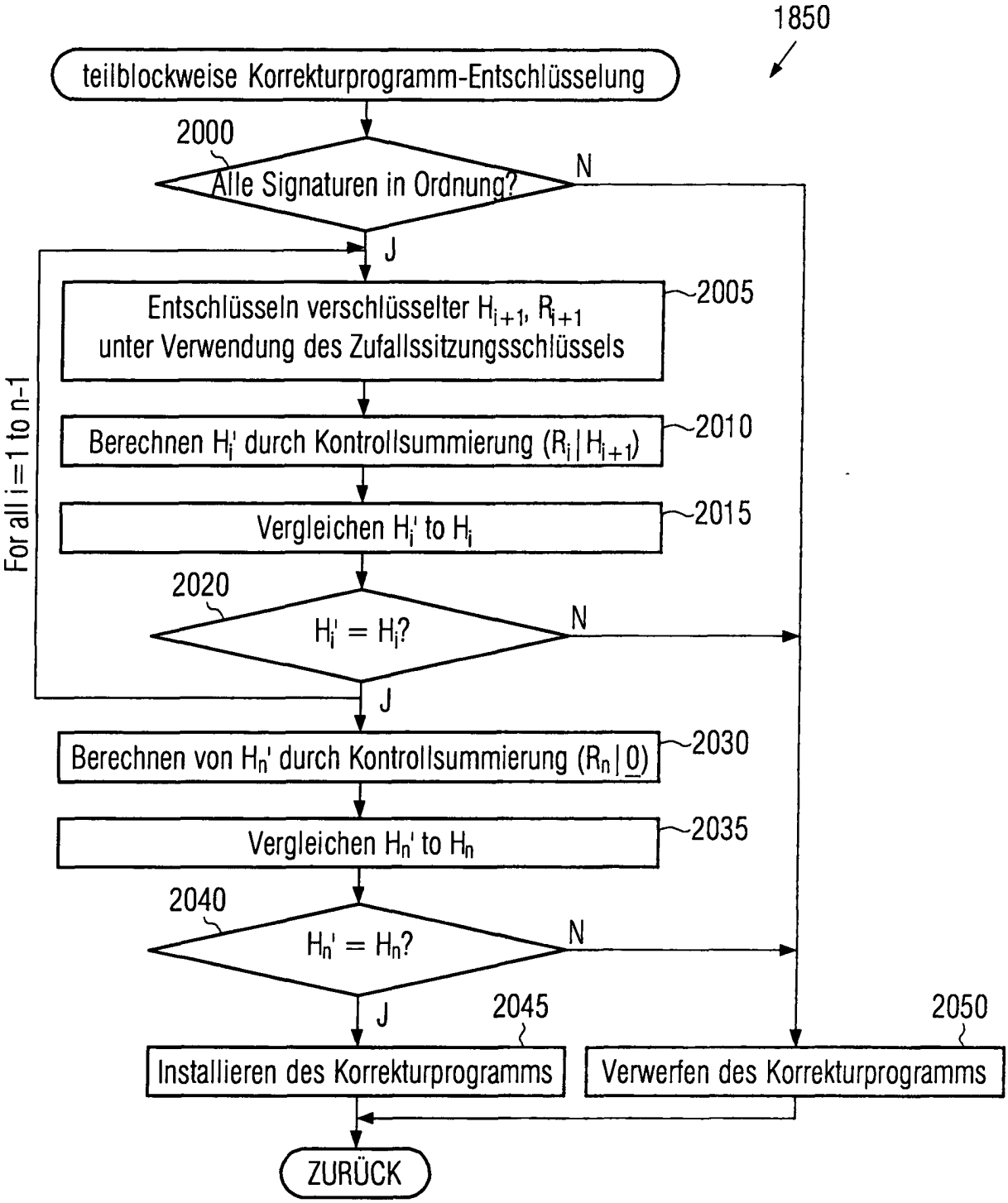

1850
teilblockweise Korrekturprogramm-Entschlüsselung
2000
Alle Signaturen in Ordnung?
N
J
For all i=1 to n-1
Entschlüsseln verschlüsselter H_{i+1}, R_{i+1} unter Verwendung des Zufallssitzungsschlüssels
2005
Berechnen H_i' durch Kontrollsummierung $(R_i | H_{i+1})$
2010
Vergleichen H_i' to H_i
2015
2020
$H_i' = H_i$?
N
J
Berechnen von H_n' durch Kontrollsummierung $(R_n | 0)$
2030
Vergleichen H_n' to H_n
2035
2040
$H_n' = H_n$?
N
J
2045
Installieren des Korrekturprogramms
2050
Verwerfen des Korrekturprogramms
ZURÜCK

FIG. 20