(19) Deutsches Patent- und Markenamt

(10) **DE 11 2020 002 164 B4** 2025.03.13

(12) # Patentschrift

(21) Deutsches Aktenzeichen: **11 2020 002 164.5**
(86) PCT-Aktenzeichen: **PCT/IB2020/052695**
(87) PCT-Veröffentlichungs-Nr.: **WO 2020/222049**
(86) PCT-Anmeldetag: **23.03.2020**
(87) PCT-Veröffentlichungstag: **05.11.2020**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **20.01.2022**
(45) Veröffentlichungstag der Patenterteilung: **13.03.2025**

(51) Int Cl.: ***G06F 21/62*** (2013.01)
G06F 3/06 (2006.01)

Innerhalb von neun Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten (§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 1 Patentkostengesetz).

(30) Unionspriorität:
16/397,392 **29.04.2019** **US**

(73) Patentinhaber:
International Business Machines Corporation, Armonk, US

(74) Vertreter:
LifeTech IP Spies & Behrndt Patentanwälte PartG mbB, 80687 München, DE

(72) Erfinder:
Moll, Georges-Henri, Biot, FR; Oudot, Olivier, Biot, FR; Kaplan, Philippe, Biot, FR; Nodet, Xavier, Biot, FR

(56) Ermittelter Stand der Technik:

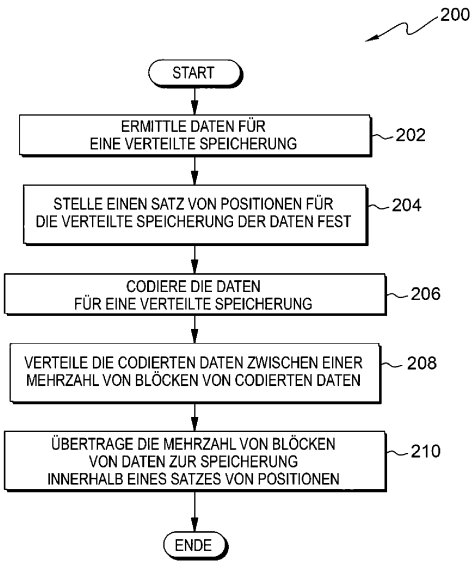
US **2013 / 0 111 166** **A1**

Block cipher mode of operation. In: Wikipedia, the free encyclopedia. Bearbeitungsstand: 27.04.2019. URL: https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=894404646 [abgerufen am 20.09.2023]

Block cipher. In: Wikipedia, the free encyclopedia. Bearbeitungsstand: 26.02.2019. URL: https://en.wikipedia.org/w/index.php?title=Block_cipher&oldid=885152613 [abgerufen am 21.09.2023]

(54) Bezeichnung: **SICHERE DATENSPEICHERUNG AUF DER GRUNDLAGE VON VERSCHLEIERUNG DURCH VERTEILUNG**

(57) Zusammenfassung: Aspekte der vorliegenden Erfindung offenbaren ein Verfahren zum sicheren Speichern von Daten. Das Verfahren umfasst ein oder mehrere Computerprozessoren, die durch einen oder mehrere Computerprozessoren eine Anforderung für das Speichern von Daten empfangen, wobei die Daten eine Mehrzahl von Elementen enthalten. Das Verfahren umfasst des Weiteren ein Erzeugen einer Mehrzahl von Elementen von codierten Daten, indem ein oder mehrere Codierungsalgorithmen auf die Daten angewendet werden, wobei eine Menge der Mehrzahl von Elementen der codierten Daten gleich einer Menge der Mehrzahl von Elementen der Daten ist. Das Verfahren umfasst des Weiteren ein Verteilen der Mehrzahl von Elementen der codierten Daten auf zwei oder mehrere Teilgruppen der codierten Daten ohne Duplizierung. Das Verfahren umfasst des Weiteren ein Übertragen der zwei oder mehreren Teilgruppen der codierten Daten an eine entsprechende Menge von zwei oder mehreren Speichersystemen.



Beschreibung

HINTERGRUND DER ERFINDUNG

[0001] Die vorliegende Erfindung betrifft allgemein das Gebiet der Datensicherheit und insbesondere ein Verschleiern der Daten, indem sie codiert und Teile der codierten Daten auf verschiedene Cloud-Speicherpositionen verteilt werden.

[0002] Cloud-Computing ist ein Informationstechnologie-Paradigma, das uneingeschränkten Zugriff auf gemeinsam genutzte Pools von konfigurierbaren Computersystemressourcen über das Internet ermöglicht. Einige Nutzer (z.B. Firmen, eine Organisation, Behörden) von Cloud-Computing-Diensten sind gegebenenfalls im Besitz von verschiedenen Datenverarbeitungssystemen, verwenden aber Cloud-Computing-Dienste, um zusätzliche Datenverarbeitungsressourcen (z.B. elastische Skalierbarkeit) bereitzustellen, um Kosten zu senken. Weitere Nutzer von Cloud-Computing-Diensten verwenden die Cloud als eine Schnittstelle zwischen Benutzern und einer Firma, um verbesserte Antwortzeiten bereitzustellen, indem sie Anwendungen und Funktionen näher an die Benutzer verteilen, um eine Netzüberlastung zu verringern, die bei einzelnen Datenverarbeitungsstandorten auftreten kann, und um das Kundenerlebnis zu verbessern. Zusätzlich zur Verarbeitung von verschiedenen Geschäftsfunktionen können Cloud-Computing-Anbieter Einzelpersonen, Anbieter von internetbasierten Diensten und Unternehmen Cloud-Speicherlösungen mit hoher Speicherkapazität über ein Infrastructure-as-a-Service-(IaaS)-Modell bereitstellen. In einem Beispiel kann ein Unternehmen eine Webseite intern hosten, Speicher jedoch an einen IaaS-Anbieter per Outsourcing auslagern.

[0003] Ein Cloud-Speicher ist ein Modell eines Computerdatenspeichers, bei dem die digitalen Daten in logischen Pools gespeichert werden. Der physische Speicher kann sich über mehrere Server (mitunter an mehreren Positionen) erstrecken und die physische Umgebung gehört üblicherweise einem Hosting-Unternehmen (d.h. einem Dienstanbieter) und wird von diesem verwaltet. Zusätzlich verwenden Cloud-Computing-Anbieter eine Vielfalt an Sicherheitsmaßnahmen, um gespeicherte Informationen und Daten zu sichern, wie beispielsweise Authentifizierung und Zugriffsüberwachung, Firewalls, Benutzertrennung, Erkennung und Abwehren von unbefugtem Zugriff sowie Sicherheitsinformations- und Ereignisprotokollierungssysteme. Somit bieten Cloud-Speicher-Anbieter große Mengen an Speicherkapazität, Just-in-time-Verwaltung der Kapazität, Sicherheit und Dauerhaftigkeit sowie einen standortunabhängigen, jederzeitigen („anywhere, anytime“) Zugriff auf Daten für Dateneigner, die geografische sowie Anbieterdiversität für die Speicherung von Daten suchen.

[0004] In diesem Kontext gibt es bereits veröffentlichte Dokumente: Das Dokument US 2013 / 0 111 166 A1 beschreibt ein Verfahren zum Datenkopieren in einen verteilten Speichernetzwerk ohne Datenreplikation. Zunächst werden ein oder mehrere Sätze von mindestens einer Dekodierschwellenanzahl von Slice-Namen für einen oder mehrere Sätze von kodierten Daten-Slices des Datenobjekts identifiziert und ein oder mehrere Sätze von mindestens einer Dekodierschwelle von neuen Slice-Namen erzeugt. Schließlich werden kodierte Daten-Slices verknüpft, wodurch eine nicht replizierte Kopie des Datenobjektes erzeugt wird. Weiterhin wird unter https://en.wikipedia.org/w/index.php?title=Block_\cipher&oldid=885152613 [abgerufen am 21.09.2023] ein Block-Cipher-Verfahren beschrieben, welches mit einer festen Anzahl von Bits - d.h. Blocks - arbeitet. Dieses Verfahren verwendet einen symmetrischen Schlüssel. Schließlich beschreibt der Wikipedia-Beitrag - mit Stand vom 27.04.2019 - https://en.wikipedia.org/w/index.php?title=Block_\cipher_mode_of_operation&oldid=894404646 auch ein Block-Cipher-Verfahren. Das Verfahren wird beschrieben als eine wiederholte Anwendung von Einzelblock-Transformationen, um Datenmenge zu verschlüsseln, die größer als ein Block sind.

[0005] Dem hier vorgestellten Konzept liegt die Aufgabe zugrunde, vorhandene Verschlüsselungstechniken insbesondere für einen Einsatz in verteilten Cloud-Computing-Umgebungen zu verbessern.

KURZDARSTELLUNG

[0006] Diese Aufgabe wird durch die Gegenstände der unabhängigen Ansprüche gelöst. Weitere Ausgestaltungen sind durch die jeweils abhängigen Ansprüche angegeben.

[0007] Aspekte der vorliegenden Erfindung offenbaren ein Verfahren, ein Computerprogrammprodukt und ein System zum sicheren Speichern von Daten. Das Verfahren umfasst ein Empfangen einer Anforderung für das Speichern von Daten durch mindestens einen Computerprozessor, wobei die Daten eine Mehrzahl von Elementen enthalten. Das Verfahren umfasst des Weiteren ein Erzeugen einer Mehrzahl von Elementen von codierten Daten durch mindestens einen Computerprozessor, indem ein oder mehrere Codierungsalgorithmen auf die Daten angewendet werden, wobei eine Menge der Mehrzahl von Elementen der codierten Daten gleich einer Menge der Mehrzahl von Elementen der Daten ist. Das Verfahren umfasst des Weiteren ein Verteilen der Mehrzahl von Elementen der codierten Daten durch mindestens einen Computerprozessor auf zwei oder

mehrere Teilgruppen der codierten Daten ohne Duplizierung. Das Verfahren umfasst des Weiteren ein Übertragen der zwei oder mehreren Teilgruppen von codierten Daten durch mindestens einen Computerprozessor an eine entsprechende Menge von zwei oder mehreren Speichersystemen.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

Fig. 1 ist ein Funktionsblockschaltbild eines Teils einer Cloud-Computing-Umgebung gemäß einer Ausführungsform der vorliegenden Erfindung.

Fig. 2 ist ein Ablaufplan, der Prozessschritte eines sicheren Datenverteilungsprogramms gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

Fig. 3 ist ein Ablaufplan, der Prozessschritte eines sicheren Datenwiederherstellungsprogramms gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

Fig. 4 ist ein veranschaulichendes Beispiel für das Codieren von Daten unter Verwendung eines Datenschlüssels, das Verteilen der codierten Daten auf Blöcke, bevor die codierten Daten gespeichert werden, und das Verwenden von Metadaten, um den Blöcken von codierten Daten Informationen zuzuordnen, gemäß mindestens einer einzelnen Ausführungsform der vorliegenden Erfindung.

Fig. 5 ist ein Blockschaltbild von Komponenten eines Computers gemäß einer Ausführungsform der vorliegenden Erfindung.

Fig. 6 stellt eine Cloud-Computing-Umgebung gemäß einer Ausführungsform der vorliegenden Erfindung dar.

Fig. 7 stellt einen Satz von funktionalen Abstraktionsschichten einer Cloud-Computing-Umgebung gemäß einer Ausführungsform der vorliegenden Erfindung dar.

AUSFÜHRLICHE BESCHREIBUNG

[0008] Ausführungsformen der vorliegenden Erfindung erkennen, dass Cloud Computing Unternehmen ermöglicht, Investitionskosten in die Informationstechnologie-(IT-)Infrastruktur zu vermeiden oder zu minimieren. Zusätzlich erkennen Ausführungsformen der vorliegenden Erfindung, dass mehr Unternehmen Anwendungen in die Cloud migrieren und dass verschiedene Unternehmen Multi-Cloud-Plattformen und mehrere Dienstanbieter für Daten und Anwendungen verwenden. Jedoch möchten die Eigner von Daten möglicherweise nicht an einen bestimmten Dienstanbieter gebunden sein. Die Eigner der Daten möchten die Flexibilität, zu einem anderen Anbieter zu wechseln, ohne Daten zu verlieren oder beträchtliche Anstrengungen bei der Datenumstellung zu unternehmen. Als Reaktion auf die zunehmende Kritikalität von Daten, die gespeichert werden, entwickeln sich die Erwartungen von Dateneignern dahingehend, dass zuverlässigere Mechanismen vorhanden sein müssen, um die Verfügbarkeit und Dauerhaftigkeit der gespeicherten Daten zu gewährleisten. Ausführungsformen der vorliegenden Erfindung erkennen, dass Cloud-Dienstanbieter bereits öffentliche Daten in der Cloud speichern, wie beispielsweise veröffentlichte behördliche Statistiken, Kartografieinformationen, sozial geteilte Informationen, Wetterdaten, Nachrichten usw. Ausführungsformen der vorliegenden Erfindung erkennen jedoch auch, dass private Daten aufgrund von Sicherheitsfragen, Vertrauensfragen, Datenschutzanforderungen, Regulierungen usw. im Allgemeinen nicht in öffentlichen Clouds gespeichert werden. Anstatt Daten innerhalb einer einzelnen Cloud zu speichern, versuchen Dateneigner in diesem Zusammenhang, Daten über mehrere Cloud-Plattformen hinweg zu speichern, um mehr Sicherheit, Redundanz zu gewährleisten sowie die mögliche Gefahr einer Beeinträchtigung der Daten zu verringern.

[0009] Ausführungsformen der vorliegenden Erfindung stellen eine Lösung bereit, um Daten in vertrauenswürdiger Form auf der Grundlage eines sicheren verteilten Formats unter mehreren Cloud-Plattformen, wie beispielsweise Cloud-Speichersystemen, öffentlichen Cloud- und/oder hybriden Cloudsystemen, sicher zu speichern. Das hierin beschriebene sichere verteilte Format für die Daten wird als Blöcke, oder genauer gesagt, als Blöcke von codierten Daten bezeichnet. Bei jedem Block von codierten Daten handelt es sich um eine Teilgruppe (z.B. eine kleinere Gruppe) von Elementen von Daten, die aus dem größeren Satz von codierten Elementen der Daten eindeutig verteilt sind. Ausführungsformen der vorliegenden Erfindung codieren die Daten für eine sichere verteilte Speicherung, bevor die Elemente der codierten Daten auf zwei oder mehrere Blöcke von codierten Daten verteilt werden. Ausführungsformen der vorliegenden Erfindung codieren die Elemente der Daten, wobei sie eine oder mehrere dateninterne Abhängigkeiten, rekursive Aktionen und/oder eine inkrementelle (d.h. indexierte) Verwendung von mindestens einem Element der ursprünglichen Daten und/oder mindestens einem vorherigen Element von codierten Daten verwenden. Einige Ausführungsformen der vorliegenden Erfindung verwenden auch einen aus anderen Informationen bestehenden Datenschlüssel, der von einem oder mehreren Codierungsalgorithmen verwendet wird, um die ursprünglichen Daten weiter zu verschleiern. Ausführungsformen der vorliegenden Erfindung verwenden umkehrbare Funktionen, um Daten zu codieren und zu decodieren.

[0010] Ausführungsformen der vorliegenden Erfindung können die Daten des Weiteren sichern, indem sie die codierten Daten auf Blöcke von codierten Daten verteilen, so dass, falls ein einzelner Block von codierten Daten verloren geht, beschädigt ist oder nicht abgerufen wird, das gesamte Original nicht wiederhergestellt (z.B. reproduziert, neu erstellt) und decodiert werden kann. In zusätzlichen Ausführungsformen können einzelne Blöcke von codierten Daten nicht für sich allein decodiert werden. Verschiedene Ausführungsformen der vorliegenden Erfindung verwenden eine oder mehrere dateninterne Elementabhängigkeiten und rekursive Operationen (z.B. Aktionen, Berechnungen) im Gegensatz zu globalen Operationen, um eine Wiederherstellung der gesamten ursprünglichen Daten unmöglich zu machen, sofern nicht alle Blöcke der codierten Daten abgerufen werden und die Verteilungsalgorithmen für die codierten Daten zusätzlich zu den Codierungs- und Decodierungsalgorithmen bekannt sind.

[0011] Somit machen einige Ausführungsformen der vorliegenden Erfindung eine Wiederherstellung der gesamten ursprünglichen Daten als Reaktion auf einen Verlust oder eine Beschädigung von auch nur einem einzigen Byte oder Element von codierten Daten unmöglich. Ein Beispiel für eine dateninterne Elementabhängigkeit ist, dass der Codierungswert für ein aktuelles Element von Daten auf dem aktuellen Wert für ein Element von Daten und mindestens einem vorherigen Wert eines Elements von codierten Daten beruht. Jede zusätzliche Abhängigkeit, wie beispielsweise das Umfassen von mindestens einem vorherigen ursprünglichen Datenwert, um einen aktuellen Datenwert zu codieren, stärkt die Sicherheitsaspekte der vorliegenden Erfindung in Verbindung mit dem Codieren und Verteilen der ursprünglichen Daten auf Blöcke von codierten Daten weiter. Ein Decodieren von codierten Daten umfasst auch dateninterne Elementabhängigkeiten wie beispielsweise den aktuellen Wert für ein Element von codierten Daten und mindestens einen vorherigen Wert eines Elements von codierten Daten. Jede zusätzliche Abhängigkeit wie beispielsweise das Verwenden von mindestens einem vorherigen Wert eines Elements von decodierten Daten, um einen Wert eines aktuellen Elements von codierten Daten zu decodieren, stärkt die Sicherheit eines Decodierungsalgorithmus weiter.

[0012] Verschiedene Ausführungsformen der vorliegenden Erfindung verbessern die Sicherheit in Bezug auf die privaten Daten weiter, indem ein Block von codierten Daten auf jeden der verschiedenen Standorte und/oder Cloud-Speicher-Anbieter verteilt wird, die ausgewählt werden, um die Blöcke von codierten Daten zu speichern, die den ursprünglichen Daten entsprechen. In einem Beispiel können zwei Blöcke von codierten Daten in der gleichen Stadt gespeichert werden, wenn mindestens zwei Cloud-Speicher-Anbieter innerhalb der Stadt über freie Speicherkapazität verfügen.

[0013] Einige Ausführungsformen der vorliegenden Erfindung verwenden eine Duplizierung von Blöcken von codierten Daten für eine Wiederherstellung nach einem Stör-/ Katastrophenfall (Disaster Recovery), wie beispielsweise, wenn ein oder mehrere Cloud-Speicher-Anbieter keine Datensicherung anbieten, oder um sicherzustellen, dass, falls ein Standort von einem Problem (z.B. Naturkatastrophe, Stromausfall, Netzfehler usw.) betroffen ist, die ursprünglichen Daten auf der Grundlage des Zugriffs auf eine sekundäre Speicherposition immer noch zeitnah wiederherstellbar sind. Ausführungsformen der vorliegenden Erfindung verwenden eine oder mehrere in der Technik bekannte Datenprüfungen, wie beispielsweise eine Prüfsumme oder eine Hashfunktion, um die Integrität der abgerufenen Blöcke von codierten Daten und/oder der decodierten Daten sicherzustellen.

[0014] Eine Ausführung von Ausführungsformen der Erfindung kann eine Vielfalt an Formen annehmen und beispielhafte Ausführungsdetails werden anschließend unter Bezugnahme auf die Figuren erörtert.

[0015] Die vorliegende Erfindung wird nun unter Bezugnahme auf die Figuren ausführlich beschrieben. **Fig.** 1 ist ein Funktionsblockschaltbild, das einen Teil einer Cloud-Computing-Umgebung, die allgemein als Umgebung 100 bezeichnet ist, gemäß einer einzelnen Ausführungsform der vorliegenden Erfindung, wie beispielsweise eines Cloud-Computing-Knotens 10 (der unter Bezugnahme auf **Fig.** 6 ausführlicher erörtert wird), veranschaulicht. Eine Ausführungsform der Umgebung 100 enthält: ein System 102 und eine Mehrzahl von Cloud-Speicherpositionen (z.B. einen Speicher 120A bis einschließlich eines Speichers 120N), die alle über ein Netzwerk 130 miteinander verbunden sind. **Fig.** 1 veranschaulicht lediglich eine einzelne Ausführung und bedeutet keine Einschränkungen in Bezug auf die Umgebungen, in denen verschiedene Ausführungsformen ausgeführt werden können. Viele Änderungen an der dargestellten Umgebung können vom Fachmann vorgenommen werden, ohne vom Umfang der Erfindung, wie von den Ansprüchen angegeben, abzuweichen.

[0016] In weiteren Ausführungsformen ist **Fig.** 1 ein Funktionsblockschaltbild, das einen Teil einer vernetzten Datenverarbeitungsumgebung oder eine Speicherarchitektur, die sich von einer Cloud-Computing-Umgebung unterscheidet, veranschaulicht, welche allgemein als Umgebung 100 bezeichnet ist. In einem Beispiel können ein oder mehrere der Speicher 120A bis einschließlich des Speichers 120N für

verschiedene Arten von Speichersystemen stehen, wie beispielsweise an das Netzwerk angeschlossene Speicher-(NAS-)Systeme oder -einheiten, Speicherbereichsnetze (storage area networks, SANs) und/oder andere Speichereinheiten.

[0017] Bei dem System 102 und dem Speicher 120A bis einschließlich des Speichers 120N kann es sich um Laptop-Computer, Tablet-Computer, Netbook-Computer, Personal Computer (PC), Desktop-Computer, elektronische Assistenten (personal digital assistants, PDA), Smartphones, Wearables (z.B. Datenbrillen, eine Smartwatch usw.) oder beliebige programmierbare, in der Technik bekannte Computersysteme handeln. In bestimmten Ausführungsformen stellen das System 102 und der Speicher 120A bis einschließlich des Speichers 120N Computersysteme dar, die Cluster-Computer und Komponenten (z.B. Datenbankserver-Computer, Anwendungsserver-Computer usw.) verwenden, welche die Funktion eines einzelnen Pools an nahtlosen Ressourcen übernehmen, wenn durch das Netzwerk 130 auf sie zugegriffen wird, wie es in Rechenzentren und bei Cloud-Computing-Anwendungen üblich ist. Das System 102 und der Speicher 120A bis einschließlich des Speichers 120N können gemäß Ausführungsformen der vorliegenden Erfindung Komponenten enthalten, wie unter Bezugnahme auf **Fig.** 5 dargestellt und ausführlicher beschrieben ist. Im Allgemeinen stehen der Speicher 120A bis einschließlich des Speichers 120N für ein beliebiges programmierbares elektronisches Gerät oder eine Kombination aus programmierbaren elektronischen Geräten, die maschinenlesbare Programmanweisungen ausführen und über das Netzwerk 130 mit dem System 102 Daten austauschen können.

[0018] Das System 102 enthält: einen Speicher 103, ein sicheres Datenverteilungsprogramm 200 und ein sicheres Datenwiederherstellungsprogramm 300. Das System 102 enthält auch verschiedene Programme wie beispielsweise eine Website-Schnittstelle; einen oder mehrere Web-Dämonprozesse; ein Datenbankverwaltungssystem; ein Netzwerkzuordnungsprogramm; ein Netzwerkübertragungsprogramm; verschiedene Funktionen und/oder Programme zur Durchführung von verschiedenen mathematischen, statistischen Konvertierungen usw. In einer einzelnen Ausführungsform steht das System 102 für ein oder mehrere Datenverarbeitungssysteme innerhalb eines Cloud-Computing-Systems. In einer weiteren Ausführungsform steht das System 102 für ein oder mehrere Datenverarbeitungssysteme einer Organisation, wie beispielsweise einer Firma, einer Universität, eines Unternehmens usw. In einigen Ausführungsformen verwendet das System 102 das Netzwerk 130 und ein oder mehrere Übertragungsprogramme und/oder - protokolle, um den Speicher 120A bis einschließlich des Speichers 120N abzufragen und Informationen in Bezug auf die Verfügbarkeit von Speicherpositionen, wie beispielsweise physische Positionen, Speicheranbieter-Informationen, Speicherkosten, Vereinbarungen zum Dienstumfang (service level agreements, SLAs), Datensicherungsfähigkeiten usw., zu erhalten. Das System 102 kann die Informationen innerhalb von Speicherinformationen 107 für einen nachfolgenden Zugriff durch das sichere Datenverteilungsprogramm 200 speichern.

[0019] In einer einzelnen Ausführungsform enthält der Speicher 103 Daten 104, Datenschlüssel 105, Sicherheitsalgorithmen 106 und Speicherinformationen 107. Der Speicher 103 kann auch verschiedene Tabellen, Listen, assoziative Feldgruppen, Datenbanken und/oder andere Programme (nicht gezeigt) enthalten. Die Daten 104 umfassen eine Mehrzahl von zu einer oder mehreren Entitäten (z.B. Benutzer, Programme, Unternehmen, Organisationen usw.) gehörenden Daten, die das System 102 für eine sichere verteilte Speicherung von Daten auf der Grundlage einer Verschleierung verwenden, indem die Daten für eine verteilte Speicherung codiert sowie die codierten Daten auf Blöcke von codierten Daten verteilt werden. In verschiedenen Ausführungsformen werden Daten innerhalb der Daten 104 durch verschiedene Entitäten und/oder ausführende Programme (nicht gezeigt) über das Netzwerk 130 auf/an das System 102 hochgeladen oder übertragen. In einer weiteren Ausführungsform werden Daten innerhalb der Daten 104 innerhalb des Systems 102 durch ein oder mehrere ausführende Programme (nicht gezeigt) erzeugt.

[0020] Die Datenschlüssel 105 umfassen eine Mehrzahl von Schlüsseln, die von einem Sicherheitsalgorithmus verwendet werden, um verteilt gespeicherte Daten zu codieren und zu decodieren. In einer Ausführungsform umfassen die Datenschlüssel 105 Datenschlüssel unterschiedlicher Strukturen und/oder Formate. In einem Beispiel kann ein Datenschlüssel innerhalb der Datenschlüssel 105 auf zufälligen numerischen Werten, alphabetischen Zeichen, Symbolen oder einer Kombination daraus beruhen. In einem weiteren Beispiel kann ein Datenschlüssel ein Kennwort oder eine Kennphrase sein, das bzw. die von einem Benutzer oder einem Programm, der bzw. das auf das System 102 zugreift, eingegeben wird. In einer bestimmten Ausführungsform beruht ein Datenschlüssel innerhalb der Datenschlüssel 105 auf einer Regel, wie beispielsweise den ersten zehn Ziffern von π, √2 oder einer Bruchzahl. In einigen Ausführungsformen führt das Verteilungsprogramm 200 ein oder mehrere Programme (z.B. Funktionen) aus, um Informationen innerhalb eines Datenschlüssels der Datenschlüssel 105 in indexierte numerische Werte zu konvertieren. In einem Fall wird ein konvertierter Datenschlüssel innerhalb der Datenschlüssel 105 gespeichert und jeweils dem ursprünglichen Datenschlüssel zugeord-

net. In einem weiteren Fall wird ein konvertierter Datenschlüssel innerhalb der Speicherinformationen 107 gespeichert und anderen Informationen in Bezug auf die Daten für eine verteilte Speicherung zugeordnet, wie beispielsweise Speicherpositions-IDs, Block-IDs, Sequenzdaten, die zu einer Mehrzahl von Blöcken von codierten Daten gehören.

[0021] Die Sicherheitsalgorithmen 106 umfassen eine Mehrzahl von Algorithmen, die verwendet werden, um Daten für eine sichere verteilte Speicherung aufzubereiten. Die Sicherheitsalgorithmen 106 umfassen Algorithmen, um: Daten zu codieren, die codierten Daten auf eine Mehrzahl von Blöcken von codierten Daten zu verteilen, die codierten Daten aus einer Mehrzahl von abgerufenen Blöcken von codierten Daten wiederherzustellen (d.h. zu rekonstruieren) und die codierten Daten zu decodieren. Ein Beispiel für einen Satz von Algorithmen, um die aktuelle Erfindung in die Praxis umzusetzen, wird nachstehend aufgezeigt und unter Bezugnahme auf **Fig.** 4 ausführlicher erörtert. Sicherheitsalgorithmen innerhalb der Sicherheitsalgorithmen 106 können von den Administratoren des Systems 102 entwickelt werden und/oder von einem Nutzer des Systems 102 entwickelt und in Verbindung mit den zu dem Benutzer gehörenden Daten verwendet werden.

[0022] In verschiedenen Ausführungsformen handelt es sich bei Datencodierungsalgorithmen innerhalb der Sicherheitsalgorithmen 106 um umkehrbare Funktionen (d.h. Algorithmen), die die Zusammensetzung der Quellendaten zur Durchführung von verschiedenen Transformationen (z.B. Codierungen) verwenden. In einer einzelnen Ausführungsform verwenden ein oder mehrere Algorithmen innerhalb der Sicherheitsalgorithmen 106 einen Datenschlüssel innerhalb der Datenschlüssel 105, um Elemente von Daten während der Codierung der Daten weiter zu verschleiern. In einer Ausführungsform umfassen die Sicherheitsalgorithmen 106 auch ein oder mehrere Programme (z.B. Funktionen), um Informationen innerhalb eines Datenschlüssels in indexierte numerische Werte zu konvertieren und/oder um einen Datenschlüssel zu erzeugen.

[0023] In einigen Ausführungsformen enthalten Algorithmen eine oder mehrere Einschränkungen. In einem Fall enthält ein Datencodierungsalgorithmus eine Einschränkung, wie beispielsweise eine Variable (z.B. max_val), die um mindestens einen Wert von 1 größer als der maximale Wert eines beliebigen Elements von Daten innerhalb der zu speichernden Daten sein muss. In einem Beispiel, unter Bezugnahme auf das Speichern von Binärdaten, ist max_val = 256, wenn dieser Wert eines Elements von Daten 0 bis 255 betragen kann. In einem weiteren Fall, wenn ein Datencodierungsalgorithmus Daten rekursiv codiert und einen aktuellen Codierungswert auf einen Wert basiert, der einem oder mehreren zuvor codierten Elementen von Daten entspricht (z.B. einer dateninternen Elementabhängigkeit), wird ein Datenverteilungsalgorithmus eingeschränkt, um Elemente von codierten Daten zu speichern, die in verschiedenen Blöcken von codierten Daten interagieren. In einer Ausführungsform enthalten die Sicherheitsalgorithmen 106 ein oder mehrere weitere Kriterien (z.B. Regeln), die sich darauf auswirken, ob und wie ein Element von Daten codiert wird. In einem Beispiel kann ein in einem Codierungsalgorithmus enthaltenes Kriterium aussagen, dass, wenn K[i] = 0, dann E[i] = D[i].

[0024] Nach wie vor unter Bezugnahme auf die Sicherheitsalgorithmen 106 wird nachstehend eine beispielhafte Ausführungsform eines Satzes von Sicherheitsalgorithmen aufgezeigt, um: eine Mehrzahl von Elementen von Daten, die in den Daten für eine verteilte Speicherung enthalten sind, zu codieren, die codierten Elemente von Daten auf Blöcke von codierten Daten zu verteilen; und um, als Reaktion darauf, dass auf die verteilt gespeicherten Daten zugegriffen werden muss, die Codierungsdaten aus einer Mehrzahl von gespeicherten Blöcken von codierten Daten abzurufen und wiederherzustellen und die wiederhergestellten codierten Daten anschließend zu decodieren, um die Daten zu reproduzieren.

[0025] In den folgenden Beispielalgorithmen stellt ein Symbol „%“ eine Modulo-Operation (d.h. Modulusoperation) dar. Zum Beispiel enthält ein Paar von Codierungsalgorithmen: E[0] = (D[0] + K[0]) % max_val (hierin als Algorithmus ED1 bezeichnet). Wobei E[0] den Codierungswert für das erste Element (Indexwert [i] = 0) der Daten darstellt. In der Verwendung hierin sind Indexpositionen und Indexwerte austauschbar. K[0] ist der erste Wert (Indexwert [i] = 0) für den Datenschlüssel der den Daten für eine sichere verteilte Speicherung zugewiesenen Datenschlüssel 105. Mit Bezug auf die hierin verwendete Modulo-Operation (d.h. %) ist der Wert, der dem Element auf der linken Seite (d.h. (D[0] + K[0])) des Symbols % entspricht, der Dividend. Der Wert, der dem Element auf der rechten Seite (d.h. max_val) des Symbols % entspricht, ist der Divisor, und E[0] ist gleich dem von der Modulo-Operation für das erste Element der Daten erzeugten Rest. In einer Ausführungsform erzeugen Codierungsalgorithmen eine Menge an Elementen von codierten Daten, die der Menge an Elementen von Daten der Daten für eine sichere verteilte Speicherung entspricht (z.B. gleich dieser ist).

[0026] Für Datenwerte, die Elementen von Daten nach dem ersten Datenwert entsprechen (z.B. Indexwerte von i > 0): E[i] = (E[i-1] + D[i-1] + D[i] +K[i % key_size]) % max_val (hierin als Algorithmus ED2 bezeichnet). Der Algorithmus ED2 enthält Codierungseinschränkungen (z.B. dateninterne Elemen-

tabhängigkeiten) und/oder mindestens eine rekursive Aktion. Ein aktueller Codierungswert eines Elements, E[i] für das entsprechende Element des Datenwerts D[i], wird von einem Element des Datenschlüssels (z.B. K[i % key_size]) geändert; und wird des Weiteren von sowohl dem vorherigen Codierungswert, E[i-1], als auch dem vorherigen Datenwert, D[i-1], geändert. Innerhalb des Algorithmus ED2 erzeugt der Term K[i % key_size] auf der Grundlage des Erzeugens eines Bereichs von Indexwerten von 0 bis (key_size - 1) als Reaktion auf key_size < data_size eine zyklische Auswahl an Elementen aus einem Datenschlüssel, die auf die zu codierenden Elemente von Daten anzuwenden ist. Data_size ist ein Wert, der der Anzahl (d.h. Menge) an Elementen der Daten für eine sichere verteilte Speicherung entspricht.

[0027] Nach wie vor unter Bezugnahme auf die Sicherheitsalgorithmen 106 und auf der Grundlage der zu dem beispielhaften Codierungsalgorithmus gehörenden Einschränkungen wird nachstehend ein zugehöriger Daten-„Chunking“-Algorithmus erörtert. In einer Ausführungsform und auf der Grundlage der Anzahl (d.h. Menge) von für den Empfang der codierten Daten für eine verteilte Speicherung ausgewählten Positionen ist die Menge an erzeugten Blöcken (nb_chunks) > =2 und kleiner oder gleich der Menge an verfügbaren Speicherpositionen für die Daten. In einem Beispiel enthält ein Satz von Daten-Chunking- und Datenelement-Verteilungsalgorithmen einen Größenwert, der einer Menge an Elementen entspricht, die innerhalb eines Blocks („Chunks“, d.h. C) enthalten sind, der aus einem Teil der codierten Daten besteht, welcher dargestellt ist durch: size[c] = (data_size + nb_chunks -1 - c) div nb_chunks (hierin als Algorithmus CD1 bezeichnet). Wobei c = ein einem Block entsprechender Indexwert ist. Innerhalb der erörterten Algorithmen stellt die Funktion „div“ eine ganzzahlige Division des Wertes des Dividenden (d.h. (data_size + nb_chunks -1 - c)) und des Wertes des Divisors (d.h. nb_chunks) dar.

[0028] In einer Ausführungsform beruht das Verteilen von codierten Elementen von Daten auf Blöcke auf einem Algorithmus, der zyklisch Indexwerte erzeugt, die jedem Block und Elementen von Daten innerhalb eines jeweiligen Blocks entsprechen. Als Reaktion darauf werden codierte Elemente von Daten, die den erzeugten Indexwerten entsprechen, auf die Blöcke verteilt oder den Blöcken zugewiesen. In einem Beispiel weist C[c][j] = E[c + j*nb_chunks] (hierin als Algorithmus CD2 bezeichnet) die codierten Informationen unter Blöcken zu, wobei 0 <= c < nb_chunks und 0 <= j < size[c]; und wobei „c“ den Indexwert eines Blocks von codierten Daten und „j“ den Indexwert darstellt, der einem Element von codierten Daten innerhalb eines Blocks von codierten Daten entspricht. In einigen Ausführungsformen wird der Indexwert von „c“ für einen Block jeweils einer Kennung (ID) zugeordnet, die einem Block entspricht. Zudem entspricht eine Block-ID in einigen Fällen einer Sequenz (z.B. 1, 2, 3, ...) in Bezug auf die Verteilung der codierten Daten auf die Blöcke. In weiteren Fällen kann die Sequenz von Blöcken von codierten Daten indirekt den Block-IDs zugeordnet werden, wie beispielsweise über eine assoziative Feldgruppe.

[0029] Den aktuellen Satz von Sicherheitsalgorithmen fortsetzend, ist in einer Ausführungsform ein entsprechender Algorithmus zum Wiederherstellen (z.B. Reproduzieren, Umkehren des Verteilungsalgorithmus) der codierten Daten aus der Mehrzahl von Elementen von Daten, die auf die Blöcke verteilt werden, dargestellt durch: E[i] = C[i % nb_chunks][i div nb_chunks] (hierin als Algorithmus RE1 bezeichnet) für 0 <= i < datasize. Wobei der Term [i % nb_chunks] des Algorithmus 3) einem Blockindex und/oder einer zugehörigen Block-ID entspricht und [i div nb_chunks] einem Satz von Indexwerten von Elementen innerhalb eines Blocks (zuvor unter Bezugnahme auf „j“ erörtert, wobei 0 <= j < size[c]) entspricht.

[0030] Der beispielhafte Satz von Datensicherheitsalgorithmen wird von einem Paar von entsprechenden Decodierungsalgorithmen vervollständigt, die auf die wiederhergestellten codierten Daten angewendet werden. Wobei der Wert für max_val der gleiche Wert von max_val ist, der zur Codierung der Daten verwendet wurde, und wobei die Indexwerte den Elementen der wiederhergestellten codierten Daten entsprechen. Zum Beispiel wird D[0] = (E[0] - K[0] + max_val) % max_val (hierin als Algorithmus DE1 bezeichnet) verwendet, um das Anfangselement (d.h. das erste Element) der wiederhergestellten codierten Daten zu decodieren. Des Weiteren ist, um nachfolgende Elemente der wiederhergestellten codierten Daten zu decodieren, D[i] = (E[i] - E[i -1] - D [i-1] - K[i % key_size] + nb_chunks * max_val) % max_val (hierin als Algorithmus DE2 bezeichnet) für 0 < i < data_size. Wie zuvor unter Bezugnahme auf das Sichern der Daten erörtert, enthält der Algorithmus DE2 mindestens eine rekursive Aktion oder dateninterne Abhängigkeit, wodurch die gesamten ursprünglichen Daten somit nicht wiederherstellbar werden, wenn ein Block von codierten Daten fehlt oder beschädigt ist.

[0031] Die Speicherinformationen 107 enthalten Informationen in Bezug auf eine Mehrzahl von Daten, die für eine verteilte Speicherung aufbereitet und/oder bereits als Blöcke von codierten Daten zwischen dem Speicher 120A bis einschließlich des Speichers 120N gespeichert sind. In einer einzelnen Ausführungsform enthalten die Speicherinformationen 107 Informationen in Bezug auf die Daten für eine verteilte Speicherung, wie beispielsweise eine oder mehrere Entitäten, die die Daten gespeichert

und/oder Zugriff auf die verteilt gespeicherten Daten haben, Speicherpositions-IDs, Block-IDs, eine zu einer Mehrzahl von Blöcken von codierten Daten gehörende Sequenz (z.B. eine auf einem Verteilungsalgorithmus beruhende Block-ID-Sequenz), einen Datenschlüssel, der während der Codierung der Daten verwendet wird, Datenintegritätsüberprüfung/- Informationen usw. In einigen Ausführungsformen werden Informationen innerhalb der Speicherinformationen 107 innerhalb einer Datenstruktur wie beispielsweise einer oder mehreren Tabellen und/oder Datenbanken gespeichert. In verschiedenen Ausführungsformen sind auch Metadaten in Bezug auf Blöcke von codierten Daten (unter Bezugnahme auf **Fig.** 4 ausführlicher beschrieben) innerhalb der Speicherinformationen 107 enthalten und des Weiteren den ursprünglichen Daten oder einer den ursprünglichen Daten für eine verteilte Speicherung zugeordneten Referenz zugeordnet.

[0032] In einer weiteren Ausführungsform werden einige Informationen innerhalb der Speicherinformationen 107 erhalten, indem das System 102 eine Mehrzahl von Speicheranbietern über das Netzwerk 130 abfragt, um Informationen in Bezug auf den Speicher 120A bis einschließlich des Speichers 120N zu erhalten, wie beispielsweise physische Positionen, Speicheranbieter-Informationen, Speicherkosten usw. In einer Ausführungsform werden Speicherpositions-IDs des Weiteren einem Speicheranbieter und/oder einer entsprechenden physischen Position (z.B. Standort) eines Speichersystems zugeordnet. In einigen Fällen werden Speicher-IDs vom System 102 zugewiesen. In weiteren Fällen entsprechen Speicher-IDs Informationen, die dem System 102 von dem Speicheranbieter bereitgestellt werden, der dem Speicher 120A bis einschließlich des Speichers 120N jeweils zugeordnet ist.

[0033] In einigen Ausführungsformen sind das sichere Datenverteilungsprogramm 200 und das sichere Datenwiederherstellungsprogramm 300 Dienste, die ein Eigner und/oder Entwickler einer App abonniert, um eine Unterstützung mehrerer Versionen für die App zu ermöglichen. Das sichere Datenverteilungsprogramm 200 und das sichere Datenwiederherstellungsprogramm 300 können über eine der Funktionen einer Verwaltungsschicht 80 einer Cloud-Computing-Umgebung 50, wie beispielsweise einer Sicherheit 86 (unter Bezugnahme auf **Fig.** 6 und **Fig.** 7 ausführlicher beschrieben), ausgeführt werden. Alternativ oder zusätzlich können das sichere Datenverteilungsprogramm 200 und das sichere Datenwiederherstellungsprogramm 300 über einen der virtualisierten Aspekte einer Arbeitslastschicht 90 der Cloud-Computing-Umgebung 50, wie beispielsweise einer Datensicherheit 96 (unter Bezugnahme auf **Fig.** 6 und **Fig.** 7 ausführlicher beschrieben), ausgeführt werden. In einer weiteren Ausführungsform werden das sichere Datenverteilungsprogramm 200 und das sichere Datenwiederherstellungsprogramm 300 innerhalb einer Speicherumgebung, die sich von einer öffentlichen Cloud-Speicher-Umgebung unterscheidet und/oder zusätzlich zu einer öffentlichen Cloud-Speicher-Umgebung besteht, ausgeführt, welche innerhalb einer anderen Architektur, wie beispielsweise eines Mainframe-Computers, einer vernetzen Datenverarbeitungsumgebung und/oder einer verteilten Datenverarbeitungsumgebung betrieben wird, um Blöcke von codierten Daten unter weiteren, zum System 102 gehörenden internen Speicherpositionen zu speichern.

[0034] Bei dem sicheren Datenverteilungsprogramm 200 handelt es sich um ein Programm, das Daten für eine sichere Speicherung innerhalb der Cloud, wie beispielsweise einer öffentlichen Cloud-Speicher-Umgebung, aufbereitet. Verschiedene Aspekte des sicheren Datenverteilungsprogramms 200 können ausgeführt werden, um innerhalb von weiteren Speicherumgebungen betrieben zu werden. In einer einzelnen Ausführungsform codiert das sichere Datenverteilungsprogramm 200 Elemente von Daten in einem numerischen Format für eine verteilte Speicherung unter Verwendung von einem oder mehreren Codierungsalgorithmen. Das sichere Datenverteilungsprogramm 200 kann einen Datenschlüssel verwenden, um die Codierung der Elemente der Daten weiter zu verschleiern. In verschiedenen Ausführungsformen verwendet das sichere Datenverteilungsprogramm 200 einen oder mehrere Verteilungsalgorithmen, um die codierten Elemente der Daten auf Blöcke von codierten Daten zu verteilen. In einer beispielhaften Ausführungsform kann das sichere Datenverteilungsprogramm 200, indem es die codierten Daten auf Blöcke von codierten Daten verteilt, die gesamten ursprünglichen Daten nicht wiederherstellbar machen, sofern nicht alle zu den ursprünglichen Daten gehörenden Blöcke von codierten Daten abgerufen werden und sofern die Verteilungsreihenfolge der codierten Elemente von Daten auf die Blöcke von codierten Daten nicht ebenfalls bekannt ist.

[0035] In einer weiteren Ausführungsform, wenn das sichere Datenverteilungsprogramm 200 feststellt, dass die Daten für eine verteilte Speicherung zuvor gespeichert wurden und die aktuelle Version der Daten geändert ist, löscht das sichere Datenverteilungsprogramm 200 auf der Grundlage des Nichtvorhandenseins einer Sicherungsvorschrift die Blöcke von codierten Daten innerhalb des verteilten Speichers, bevor es die Blöcke von codierten Daten speichert, die der aktuellen Version der Daten entsprechen. Das sichere Datenverteilungsprogramm 200 kann eine oder mehrere Speicherpositionen wiederverwenden, um die Blöcke von codierten Daten zu speichern, die der aktuellen Version der Daten

entsprechen, oder das sichere Datenverteilungsprogramm 200 kann einen anderen Speichersatz auswählen, um die Blöcke von codierten Daten zu speichern, die der aktuellen Version der Daten entsprechen. In einer Ausführungsform migriert das sichere Datenverteilungsprogramm 200 Daten für eine verteilte Speicherung in den RAM (nicht gezeigt) des Systems 102, um die Leistung von einem oder mehreren Sicherheitsalgorithmen vor der Übertragung der Blöcke von codierten Daten an einen Satz von Speicherpositionen zu verbessern.

[0036] Bei dem sicheren Datenwiederherstellungsprogramm 300 handelt es sich um ein Programm, das Daten ermittelt, die verteilt gespeichert wurden, die Mehrzahl der Blöcke von codierten Daten abruft, die die verteilt gespeicherten Daten aufweisen, und die codierten Daten decodiert, nachdem die codierten Daten aus der Mehrzahl der abgerufenen Blöcke von codierten Daten wiederhergestellt wurden. In einer einzelnen Ausführungsform dient das sichere Datenwiederherstellungsprogramm 300 dazu, Daten aus einer Mehrzahl von Blöcken von codierten Daten wiederherzustellen, die auf zwei oder mehrere Cloud-Speicherpositionen verteilt sind. In einigen Ausführungsformen wird das sichere Datenwiederherstellungsprogramm 300 automatisch von einem Sicherheitsprogramm, einer Funktion und/oder einer API als Reaktion darauf ausgeführt, dass ein Benutzer und/oder ein Programm anzeigt, auf die verteilt gespeicherten Daten zuzugreifen.

[0037] In verschiedenen Ausführungsformen verwendet das sichere Datenwiederherstellungsprogramm 300 replizierte oder gesicherte Blöcke von codierten Daten, um die codierten Daten wiederherzustellen. Das sichere Datenwiederherstellungsprogramm 300 kann ein oder mehrere Datenintegritätsüberprüfungsverfahren verwenden, um sicherzustellen, dass ein replizierter Block von codierten Daten ein exaktes Duplikat eines fehlenden oder beschädigten Blocks von codierten Daten ist. In einer Ausführungsform speichert das sichere Datenwiederherstellungsprogramm 300 die abgerufenen Blöcke von codierten Daten und/oder die wiederhergestellten codierten Daten innerhalb des RAM (nicht gezeigt) des Systems 102, um die Leistung von einem oder mehreren Sicherheitsalgorithmen zu verbessern.

[0038] In einer einzelnen Ausführungsform tauscht das System 102 durch das Netzwerk 130 mit dem Speicher 120A bis einschließlich des Speichers 120N Daten aus. In einer weiteren Ausführungsform kann das Netzwerk 130 das System 102 auch mit einem oder mehreren Knoten der Cloud-Computing-Umgebung 50 (unter Bezugnahme auf **Fig.** 6 ausführlicher erörtert) verbinden, der bzw. die einen oder mehrere Dienste wie beispielsweise eine Überwachung von Softwarevereinbarungen (unter Bezugnahme auf **Fig.** 7 ausführlicher erörtert) enthält/enthalten. In einem Beispiel kann das Netzwerk 130 einen Netzagenten/Dämonprozess enthalten, der zu Apps gehörenden Netzwerkverkehr, welche zur Verwendung von einer oder mehreren Ausführungsformen der vorliegenden Erfindung lizenziert sind, an das System 102 weiterleitet. Alternativ kann das Netzwerk 130 zu Apps gehörenden Netzwerkverkehr, welche nicht zur Verwendung von Ausführungsformen der vorliegenden Erfindung lizenziert sind, an einen anderen Knoten innerhalb der Cloud-Computing-Umgebung 50 weiterleiten.

[0039] Das Netzwerk 130 kann zum Beispiel ein lokales Netz (LAN, local area network), ein Telekommunikationsnetz, ein drahtloses lokales Netzwerk (WLAN, wireless local area network), wie beispielsweise ein Intranet, ein Weitverkehrsnetz (WAN, wide area network), wie beispielsweise das Internet, oder eine beliebige Kombination der Vorstehenden sein und kann kabelgebundene, kabellose oder Lichtwellenleiterverbindungen enthalten. Im Allgemeinen kann es sich bei dem Netzwerk 130 um eine beliebige Kombination aus Verbindungen und Protokollen handeln, die Übertragungen zwischen dem System 102 und dem Speicher 120A bis einschließlich des Speichers 120N gemäß Ausführungsformen der vorliegenden Erfindung unterstützen. In verschiedenen Ausführungsformen arbeitet das Netzwerk 130 lokal über kabelgebundene, kabellose oder optische Verbindungen und kann eine beliebige Kombination aus Verbindungen und Protokollen (z.B. Personal Area Network (PAN), Near Field Communication (NFC), Laser, Infrarot, Ultraschall usw.) sein. In einigen Ausführungsformen ist eine Instanz des Netzwerks 130 ein virtuelles LAN (VLAN), das zwei oder mehrere virtualisierte Entitäten (nicht gezeigt) der Cloud-Computing-Umgebung 50 verbindet. In weiteren Ausführungsformen steht das Netzwerk 130 für eine oder mehrere LANs und/oder VLANs innerhalb einer Instanz der Umgebung 100, bei der es sich um eine vernetzte Datenverarbeitungsumgebung handelt, die Übertragungen zwischen dem System 102 und dem Speicher 120A bis einschließlich des Speichers 120N ermöglicht.

[0040] In einer einzelnen Ausführungsform stehen der Speicher 120A bis einschließlich des Speichers 120N für eine Mehrzahl von Speichersystemen von jeweiligen unterschiedlichen Standorten. In verschiedenen Ausführungsformen sind ein oder mehrere Speicher des Speichers 120A bis einschließlich des Speichers 120N innerhalb der Cloud-Speichersysteme und/oder Cloud-Computing-Umgebungen enthalten. In einigen Ausführungsformen entsprechen die Cloud-Speichersysteme und/oder Cloud-Computing-Umgebungen, die jeweiligen Instanzen eines Speichers 120 hosten, zwei oder mehreren unterschiedlichen Anbietern oder Cloud-Dienstanbietern. In einer weiteren Ausführungsform stehen der Spei-

cher 120A bis einschließlich des Speichers 120N für eine Mehrzahl von Speichersystemen und/oder Speichereinheiten einer vernetzten Datenverarbeitungsumgebung, einer verteilten Datenverarbeitungsumgebung usw., die zu dem System 102 gehört.

[0041] Fig. 2 ist ein Ablaufplan, der die Prozessschritte des sicheren Datenverteilungsprogramms 200 darstellt, eines Programms, das gemäß Ausführungsformen der vorliegenden Erfindung Daten sicher speichert, indem es Daten codiert, die codierten Daten auf kleinere Sätze von Daten verteilt und die kleineren Sätze von Daten an eine Mehrzahl von Speicherpositionen überträgt. In verschiedenen Ausführungsformen werden einige der von dem sicheren Datenverteilungsprogramm 200 ermittelten und/oder aktualisierten Informationen anschließend von Aspekten des sicheren Datenwiederherstellungsprogramms 300 verwendet, um die verteilt gespeicherten Daten abzurufen und wiederherzustellen.

[0042] Im Schritt 202 ermittelt das sichere Datenverteilungsprogramm 200 Daten für eine verteilte Speicherung. In einer einzelnen Ausführungsform ermittelt das sichere Datenverteilungsprogramm 200 Daten innerhalb der Daten 104 für eine verteilte Speicherung auf der Grundlage von von einem Benutzer, einem Programm, einer Sicherheitsfunktion erhaltenen Informationen und/oder auf der Grundlage von zu den Daten gehörenden Sicherheitsregeln. In einigen Ausführungsformen stellt das sichere Datenverteilungsprogramm 200 weitere zu den Daten für eine sichere verteilte Speicherung gehörende Informationen fest, wie beispielsweise einen Wert oder einen Bereich von Werten, der bzw. die der Menge an Speicherpositionen entspricht/entsprechen, auf die die Daten verteilt werden sollen, eine zu einem Datenschlüssel gehörende Vorschrift (z.B. Schlüssellänge, Schlüssel-ID, Regeln in Bezug auf Datenschlüssel usw.), einen unter den Elementen, die die zu speichernden Daten aufweisen, ermittelten Maximalwert usw. In einer weiteren Ausführungsform stellt das sichere Datenverteilungsprogramm 200 Informationen, die zu den Daten für eine sichere verteilte Speicherung innerhalb der Datenschlüssel 105 gehören, Sicherheitsalgorithmen 106 und/oder Speicherinformationen 107 fest. Das sichere Datenverteilungsprogramm 200 kann zu den Daten für eine sichere verteilte Speicherung gehörende Einschränkungen wie beispielsweise Kostenvorgaben oder behördliche Bestimmungen ermitteln.

[0043] In einer Ausführungsform kann das sichere Datenverteilungsprogramm 200 verschiedene, auf dem System 102 installierte Funktionen und/oder Programme (nicht gezeigt) verwenden, um verschiedene mathematische, statistische Konvertierungen usw. durchzuführen, um Informationen in Bezug auf die Daten für eine verteilte Speicherung zu erhalten, wie beispielsweise eine den Daten entsprechende Größe, max_val in Bezug auf die Daten, Indexwerte für Elemente von Schlüsseln und den Daten sowie Erzeugungstabellen/assoziative Feldgruppen, die zu den Daten für eine verteilte Speicherung gehörende Informationen verknüpfen. In verschiedenen Ausführungsformen ermittelt das sichere Datenverteilungsprogramm 200 auch einen Satz von Algorithmen, die in Verbindung mit den Daten für eine verteilte Speicherung verwendet werden, wie beispielsweise Datencodierungs-, Daten-Chunking- und Datenverteilungsalgorithmen.

[0044] Im Schritt 204 stellt das sichere Datenverteilungsprogramm 200 einen Satz von Positionen für die verteilte Speicherung der Daten fest. In einer Ausführungsform stellt das sichere Datenverteilungsprogramm 200 einen Satz von Positionen und/oder Speicher-IDs für die verteilte Speicherung von Daten auf der Grundlage von Informationen fest, die innerhalb der Speicherinformationen 107 zur Verfügung stehen. Anschließend kann das sichere Datenverteilungsprogramm 200 weitere Informationen in Bezug auf die verteilte Speicherung der Daten innerhalb der Speicherinformationen 107 enthalten und/oder aktualisieren. In verschiedenen Ausführungsformen verwendet das sichere Datenverteilungsprogramm 200 ein Netzwerk 130, um einen oder mehrere Cloud-Speicher-Anbieter abzufragen, um Positionen zu ermitteln, wie beispielsweise den Speicher 120N bis einschließlich des Speichers 120N, die zur Speicherung der Blöcke von codierten Daten zur Verfügung stehen, welche die Daten für eine verteilte Speicherung aufweisen. In einer einzelnen Ausführungsform stellt das sichere Datenverteilungsprogramm 200 auf der Grundlage von Informationen in Bezug auf die Menge der Speicherpositionen, die zu den Daten für eine verteile Speicherung gehören, einen Satz von Positionen fest (im Schritt 202).

[0045] In einer weiteren Ausführungsform stellt das sichere Datenverteilungsprogramm 200 einen Satz von Positionen für eine verteilte Speicherung der Daten auf der Grundlage der Menge an verfügbaren Positionen fest, die als Reaktion auf das Abfragen des einen oder der mehreren Cloud-Speicher-Anbieter ermittelt wurden. In einigen Ausführungsformen ermittelt das sichere Datenverteilungsprogramm 200 eine oder mehrere Speicherpositionen zusätzlich zu der Menge an Speicherpositionen, die durch Informationen vorgeschrieben werden, welche zu den Daten für eine verteilte Speicherung gehören. Anschließend verwendet das sichere Datenverteilungsprogramm 200 eine oder mehrere Einschränkungen oder Regeln, um den Satz von Positionen für die verteilte Speicherung der Daten festzustellen. In weiteren Ausführungsformen stellt das sichere Datenverteilungsprogramm 200 einen Satz von Speicherpositionen für die verteilte Speicherung der

Daten auf der Grundlage von einer oder mehreren Speicherarchitekturen fest, die sich von der Cloud-Speicher-Umgebung unterscheiden.

[0046] Im Schritt 206 codiert das sichere Datenverteilungsprogramm 200 die Daten für eine verteilte Speicherung. Das sichere Datenverteilungsprogramm 200 verwendet einen oder mehrere Codierungsalgorithmen, um eine Mehrzahl von Elementen von codierten Daten zu erzeugen, die jeweils der Mehrzahl von Elementen der ursprünglichen Daten für eine verteilte Speicherung zugeordnet sind. In einer einzelnen Ausführungsform codiert das sichere Datenverteilungsprogramm 200 die Daten für eine verteilte Speicherung unter Verwendung von einem oder mehreren Codierungsalgorithmen innerhalb der Sicherheitsalgorithmen 106, die von Informationen vorgeschrieben werden, welche zu den Daten für eine sichere verteilte Speicherung gehören. In einer weiteren Ausführungsform codiert das sichere Datenverteilungsprogramm 200 die Daten für eine verteilte Speicherung unter Verwendung von einem oder mehreren, aus den Sicherheitsalgorithmen 106 ausgewählten Codierungsalgorithmen.

[0047] In einigen Ausführungsformen verwendet das sichere Datenverteilungsprogramm 200 einen oder mehrere Codierungsalgorithmen, die die codierten Daten weiter verschleiern, indem sie einen oder mehrere numerische Werte aus einem Datenschlüssel anwenden (unter Bezugnahme auf **Fig.** 4 ausführlicher erörtert). Wenn das sichere Datenverteilungsprogramm 200 feststellt, dass ein Datenschlüssel auf einer oder mehreren Regeln beruht und/oder nicht numerische Zeichen enthält, verwendet das sichere Datenverteilungsprogramm 200 eine oder mehrere Konvertierungen und/oder Regeln, um eine Version eines Datenschlüssels zu erzeugen, der numerische Werte enthält. In verschiedenen Ausführungsformen aktualisiert das sichere Datenverteilungsprogramm 200 zu den Daten für eine sichere verteilte Speicherung gehörende Informationen auf der Grundlage der ermittelten Informationen in Bezug auf das Codieren der Daten für eine verteilte Speicherung, wie beispielsweise eine Variable max_val, einen Datenschlüssel aus den Datenschlüsseln 105, Informationen, die den einen oder die mehreren verwendeten Codierungsalgorithmen ermitteln, eine numerische Version eines Datenschlüssels, eine Prüfsumme oder einen Hashwert, der den codierten Daten entspricht, usw.

[0048] Im Schritt 208 verteilt das sichere Datenverteilungsprogramm 200 die codierten Daten auf eine Mehrzahl von Blöcken von codierten Daten. In einer Ausführungsform verteilt das sichere Datenverteilungsprogramm 200 die codierten Daten auf eine Mehrzahl von Blöcken (z.B. nb_chunks) von codierten Daten, die der Menge an Speicherpositionen innerhalb des festgestellten Satzes von Speicherpositionen entsprechen. Das sichere Datenverteilungsprogramm 200 kann des Weiteren Informationen feststellen und/oder zuordnen, wie beispielsweise eine ID, Sequenzinformationen und einen Größenwert, der den Blöcken von codierten Daten entspricht (unter Bezugnahme auf **Fig.** 4 ausführlicher erörtert), eine Prüfsumme oder einen Hashwert für jeden Block codierter Daten usw.

[0049] In einigen Ausführungsformen verteilt das sichere Datenverteilungsprogramm 200 die codierten Daten unter Verwendung von einem oder mehreren Daten-Chunking- und Datenverteilungsalgorithmen innerhalb der Sicherheitsalgorithmen 106 und wie von zu den Daten für eine verteilte Speicherung gehörenden Informationen vorgeschrieben (zuvor unter Bezugnahme auf die Sicherheitsalgorithmen 106 erörtert). In einer weiteren Ausführungsform verteilt das sichere Datenverteilungsprogramm 200 die codierten Daten unter Verwendung von einem oder mehreren aus den Sicherheitsalgorithmen 106 ausgewählten Daten-Chunking- und Datenverteilungsalgorithmen. Wenn das sichere Datenverteilungsprogramm 200 feststellt, dass ein verwendeter Codierungsalgorithmus eine dateninterne Elementabhängigkeit enthält, verwendet das sichere Datenverteilungsprogramm 200 in verschiedenen Ausführungsformen einen mit der dateninternen Elementabhängigkeit kompatiblen Datenverteilungsalgorithmus (unter Bezugnahme auf **Fig.** 4 ausführlicher erörtert).

[0050] Im Schritt 210 überträgt das sichere Datenverteilungsprogramm 200 die Mehrzahl der Blöcke von codierten Daten zur Speicherung innerhalb eines Satzes von Positionen. In verschiedenen Ausführungsformen führt das sichere Datenverteilungsprogramm 200 ein oder mehrere Übertragungsprotokolle aus, um über das Netzwerk 130 zu dem festgestellten Satz von cloudbasierten Speicherpositionen (z.B. zwischen dem Speicher 120A bis einschließlich des Speichers 120N), die die Blöcke von codierten Daten speichern, eine Verbindung aufzubauen und sich zu authentifizieren (z.B. einen Zugriff herzustellen). In einer einzelnen Ausführungsform löscht das sichere Datenverteilungsprogramm 200 als Reaktion darauf, dass das sichere Datenverteilungsprogramm 200 Bestätigungen empfängt, dass jeder der übertragenen Blöcke von codierten Daten erfolgreich gespeichert ist, die ursprünglichen Daten aus den Daten 104. In einigen Ausführungsformen, wenn das sichere Datenverteilungsprogramm 200 einen Block von codierten Daten nicht innerhalb einer Speicherposition aus dem festgestellten Satz von Speicherpositionen speichern kann (z.B. ein Authentifizierungsfehler, eine Zeitlimitüberschreitung, Konnektivitätsprobleme, ein Problem, ein weiteres Verfügbarkeitsproblem usw.), wählt das sichere Datenverteilungsprogramm 200 eine andere Speicherposition für den Block von codierten Daten aus

den festgestellten ein oder mehreren zusätzlichen (z.B. alternativen) Speicherpositionen aus.

[0051] In einer Ausführungsform aktualisiert das sichere Datenverteilungsprogramm 200 als Reaktion darauf, dass das sichere Datenverteilungsprogramm 200 Bestätigungen empfängt, dass jeder der übertragenen Blöcke von codierten Daten erfolgreich gespeichert ist, die Speicherinformationen 107, die die Positionen ermitteln, welche einem Block von codierten Daten entsprechen. Das sichere Datenverteilungsprogramm 200 aktualisiert die Speicherinformationen 107 auf der Grundlage der Informationen in Bezug auf die verteilte Speicherung der Daten, wie beispielsweise des Aktualisierens von Metadaten-Informationen und/oder anderen Datenstrukturen (z.B. einer Datenbank, einer Tabelle, einer assoziativen Feldgruppe usw.), die Informationen in Bezug auf die verteilte Speicherung der Daten speichern.

[0052] Fig. 3 ist ein Ablaufplan, der Prozessschritte des sicheren Datenwiederherstellungsprogramms 300 gemäß Ausführungsformen der vorliegenden Erfindung darstellt, eines Programms, das die für eine sichere Speicherung verteilten Daten wiederherstellt, indem die Mehrzahl der Blöcke von codierten Daten, die für eine sichere Speicherung auf eine Mehrzahl von Positionen verteilt wurden, abgerufen, wieder zusammengesetzt und decodiert werden.

[0053] Im Schritt 302 ermittelt das sichere Datenwiederherstellungsprogramm 300 Informationen in Bezug auf verteilt gespeicherte Daten. Das sichere Datenwiederherstellungsprogramm 300 kann auf eine Anforderung an das System 102, verteilt gespeicherte Daten abzurufen und wiederherzustellen, auf der Grundlage des Empfangens der Anforderung von einer oder mehreren Entitäten, die das System 102 verwenden, wie beispielsweise einem Benutzer, Softwareanwendungen und/oder Systemfunktionen, reagieren. In einer Ausführungsform ermittelt das sichere Datenwiederherstellungsprogramm 300 Informationen in Bezug auf verteilt gespeicherte Daten auf der Grundlage von Informationen innerhalb der Speicherinformationen 107. In einem Beispiel ermittelt das sichere Datenwiederherstellungsprogramm 300 Informationen innerhalb der Speicherinformationen 107 in Bezug auf die verteilt gespeicherten Daten, wie beispielsweise Block-IDs, eine Sequenz in Bezug auf die Blöcke von codierten Daten (z.B. eine Block-ID-Sequenz), Blöcken von codierten Daten entsprechende Speicherpositionen, einen Satz von Sicherheitsalgorithmen, der verwendet wird, um die Daten für eine verteilte Speicherung aufzubereiten, einen während einer Codierung der Daten verwendeten Datenschlüssel usw. Das sichere Datenwiederherstellungsprogramm 300 kann auch eine angegebene Position zur Speicherung der wiederhergestellten Daten ermitteln. In einigen Ausführungsformen ermittelt das sichere Datenwiederherstellungsprogramm 300 auch weitere Informationen über die verteilten Daten, wie beispielsweise, ob das sichere Datenverteilungsprogramm 200 eine Sicherungs-(z.B. Replikations-) Strategie, um extra Kopien der Blöcke der codierten Daten zu verteilen, Datenintegritätsüberprüfungsinformationen sowie Informationen in Bezug auf die Speicherung der extra Kopien der Blöcke von codierten Daten verwendet hat.

[0054] Im Schritt 304 ruft das sichere Datenwiederherstellungsprogramm 300 die Blöcke von codierten Daten ab, die den verteilt gespeicherten Daten entsprechen. Das sichere Datenwiederherstellungsprogramm 300 verwendet die ermittelten Informationen in Bezug auf verteilt gespeicherte Daten innerhalb der Speicherinformationen 107, um die Speicherpositionen zu ermitteln, die Blöcken von codierten Daten entsprechen, welche den verteilt gespeicherten Daten entsprechen. In einer bestimmten Ausführungsform führt das sichere Datenverteilungsprogramm 300 ein oder mehrere Übertragungsprotokolle aus, um über das Netzwerk 130 zu den Instanzen des Speichers 120, die die abzurufenden Blöcke von codierten Daten speichern, eine Verbindung aufzubauen und sich zu authentifizieren (z.B. einen Zugriff herzustellen). In verschiedenen Ausführungsformen ruft das sichere Datenwiederherstellungsprogramm 300 die Blöcke von codierten Daten aus den ermittelten Cloud-Speicherpositionen aus dem Speicher 120A bis einschließlich des Speichers 120N (z.B. unterschiedliche Standorte) ab.

[0055] In einer weiteren Ausführungsform löscht das sichere Datenwiederherstellungsprogramm 300 als Reaktion darauf, dass das sichere Datenwiederherstellungsprogramm 300 die Blöcke von codierten Daten, die den verteilt gespeicherten Daten entsprechen, erfolgreich abruft, die Blöcke von codierten Daten aus jeweiligen Speicherpositionen. Alternativ kann das sichere Datenwiederherstellungsprogramm 300 auf der Grundlage von Informationen in Bezug auf die verteilt gespeicherten Daten die Blöcke von codierten Daten innerhalb jeweiliger Speicherpositionen belassen.

[0056] In einer Ausführungsform, wenn das sichere Datenwiederherstellungsprogramm 300 feststellt, dass auf eine oder mehrere Positionen, die die Blöcke von codierten Daten speichern, welche den verteilt gespeicherten Daten entsprechen, nicht zugegriffen werden kann (z.B. Netzwerkprobleme, Serverprobleme usw.), stellt das sichere Datenwiederherstellungsprogramm 300 fest, ob das sichere Datenverteilungsprogramm 200 eine Replikationsstrategie verwendet hat, um die Blöcke von codierten Daten zu verteilen, die den verteilt gespeicherten Daten entsprechen. Als Reaktion auf die Feststellung, dass das sichere Datenverteilungsprogramm

200 eine Replikationsstrategie verwendet hat, ermittelt das sichere Datenwiederherstellungsprogramm 300 eine oder mehrere andere (z.B. sekundäre, alternative) Positionen aus dem Speicher 120A bis einschließlich des Speichers 120N, die einen Block von codierten Daten speichern, der nicht aus einer primären Speicherposition abgerufen werden kann.

[0057] Weiterhin unter Bezugnahme auf den Schritt 304 verwendet das sichere Datenwiederherstellungsprogramm 300 in weiteren Ausführungsformen verschiedene Übertragungsverfahren zur Herstellung eines Zugriffs auf eine oder mehrere Instanzen des Speichers 120A bis einschließlich des Speichers 120N, die die Blöcke von codierten Daten an anderen oder zusätzlich zu Cloud-Speicherpositionen speichern, wie beispielsweise verschiedene Speicherknoten, Server, SANs-Systeme und/oder NASs-Systeme. In einer Ausführungsform können sich eine oder mehrere der weiteren Instanzen des Speichers 120A bis einschließlich des Speichers 120N am gleichen Standort und/der der gleichen physischen Position befinden.

[0058] Im Schritt 306 stellt das sichere Datenwiederherstellungsprogramm 300 die codierten Daten aus den abgerufenen Blöcken von codierten Daten wieder her. In einer Ausführungsform verwendet das sichere Datenwiederherstellungsprogramm 300 einen oder mehrere Algorithmen zum Wiederherstellen von Daten, die innerhalb der Sicherheitsalgorithmen 106 enthalten sind, sowie von Informationen in Bezug auf die verteilt gespeicherten Daten, um die codierten Daten aus der Mehrzahl von Elementen von codierten Daten wiederherzustellen, die auf die Blöcke von codierten Daten verteilt sind. In einer einzelnen Ausführungsform verwendet das sichere Datenwiederherstellungsprogramm 300 die Block-IDs und/oder Blocksequenzinformationen, um die codierten Daten, die den verteilt gespeicherten Daten entsprechen, wiederherzustellen (z.B. wieder zusammenzusetzen, neu zu erstellen, zu reproduzieren). In einer weiteren Ausführungsform verwendet das sichere Datenwiederherstellungsprogramm 300 zu den Blöcken von codierten Daten gehörende Metadaten, um codierte Daten wiederherzustellen, die den verteilt gespeicherten Daten entsprechen. In verschiedenen Ausführungsformen verwendet das sichere Datenwiederherstellungsprogramm 300 den einen oder die mehreren Wiederherstellungsalgorithmen, die die Prozesse der Daten-Chunking- und Datenverteilungsalgorithmen umkehren, welche auf die ursprünglichen codierten Daten angewendet wurden, die den verteilt gespeicherten Daten entsprechen.

[0059] Im Schritt 308 decodiert das sichere Datenwiederherstellungsprogramm 300 die wiederhergestellten codierten Daten. Das sichere Datenwiederherstellungsprogramm 300 decodiert die wiederhergestellten codierten Daten, die den verteilt gespeicherten Daten entsprechen. In verschiedenen Ausführungsformen ermittelt das sichere Datenwiederherstellungsprogramm 300 weitere im Schritt 302 erhaltene Informationen in Bezug auf das Decodieren der codierten Daten, die den verteilt gespeicherten Daten entsprechen. In einem Beispiel ermittelt das sichere Datenwiederherstellungsprogramm 300 den zum Codieren der Daten für eine verteilte Speicherung verwendeten Datenschlüssel, Informationen in Bezug auf den Datenschlüssel (z.B. Schlüsselgröße, Datenschlüssel-Erzeugungsregeln usw.) und einen Decodierungsalgorithmus, der dem zum Codieren der Daten für eine verteilte Speicherung verwendeten Codierungsalgorithmus entspricht. In einer einzelnen Ausführungsform decodiert das sichere Datenwiederherstellungsprogramm 300 die wiederhergestellten codierten Daten, indem es den ermittelten, innerhalb der Datenschlüssel 105 gespeicherten Datenschlüssel und den entsprechenden, innerhalb der Sicherheitsalgorithmen 106 enthaltenen Decodierungsalgorithmus verwendet. In einer weiteren Ausführungsform decodiert das sichere Datenwiederherstellungsprogramm 300 die wiederhergestellten codierten Daten, indem es den ermittelten Datenschlüssel und/oder eine in den Speicherinformationen 107 enthaltene und zu den verteilt gespeicherten Daten gehörende Datenschlüssel-Erzeugungsregel verwendet.

[0060] **Fig.** 4 ist ein veranschaulichendes Beispiel für eine Aufbereitung von Daten, die eine Mehrzahl von Elementen von Daten für eine sichere verteilte Speicherung enthalten, die auf dem Verwenden von einem Satz von Sicherheitsalgorithmen (z.B. der Algorithmen ED1, ED2, CD1, CD2, RE1, DE1 und DE2) beruht, um gemäß Ausführungsformen der vorliegenden Erfindung die Mehrzahl von Elementen von Daten zu codieren und die codierten Elemente von Blöcken von codierten Daten anschließend zu verteilen. **Fig.** 4 stellt auch die Ausgabe von verschiedenen Aspekten von einigen der Sicherheitsalgorithmen dar, die auf die Daten für eine sichere verteilte Speicherung angewendet werden.

[0061] Bei den Daten 401 handelt es sich um ein Beispiel von Daten für eine sichere verteilte Speicherung. In einer Ausführungsform verfügen die Daten 401 über zugehörige Metadaten (nicht gezeigt) in Bezug auf einen Namen oder eine andere Kennung (z.B. Daten-ID), wie beispielsweise „A“. Zu den Daten 401 gehörende Metadaten können in den Speicherinformationen 107 enthalten sein. In dem veranschaulichenden Beispiel bestehen die Daten 401 aus fünfzehn Elementen von Daten. In dem veranschaulichenden Beispiel stellt die Zeile 402 die Indexpositionen (z.B. Indexwerte) dar, die von 0 bis 14 reichen und den Elementen von Daten der Daten 401 entsprechen. In dem dargestellten Beispiel beträgt der Wert des entsprechenden Elements von

Daten innerhalb der Daten 401 an der Indexposition i = 0 „14“, und bei i = 10 beträgt der Wert des entsprechenden Elements von Daten „1“. In einer Ausführungsform stellt das sichere Datenverteilungsprogramm 200 (im Schritt 202) fest, dass max_val = 24 für das Codieren von Daten für die Daten 401 (z.B. 23+1) ist. Ferner stellt das sichere Datenverteilungsprogramm 200 einen Wert fest, der der Menge an Elementen von Daten innerhalb der Daten 401 entspricht (d.h. data_size = 15).

[0062] Ferner stellt das sichere Datenverteilungsprogramm 200 auf der Grundlage von Informationen, die zu den Daten 401 gehören, wie beispielsweise einer zu einem Datenschlüssel gehörenden Vorschrift, auch fest, dass der den Daten 401 zugewiesene Codierungsalgorithmus einen Schlüssel von key_size = 3 vorschreibt. In dem dargestellten Beispiel besteht der Schlüssel 403 aus drei Elementen: 3, 1 und 9 mit jeweiligen Indexpositionen von 0, 1 und 2. In einer Ausführungsform stellt das sichere Datenverteilungsprogramm 200 fest, dass keine zusätzlichen Konvertierungen oder Bearbeitungen von Informationen innerhalb des Schlüssels 403 notwendig sind, da es sich bei den drei Elementen des Schlüssels 403 um numerische Werte handelt.

[0063] In einer Ausführungsform legt das sichere Datenverteilungsprogramm 200 (im Schritt 202) fest, den Algorithmus ED1 und den Algorithmus ED2 (die zuvor unter Bezugnahme auf die Sicherheitsalgorithmen 106 beschrieben wurden), E[0] = (D[0] + K[0]) % max_val und E[i] = (E[i-1] + D[i-1] + D[i] +K[i % key_size]) % max_val, zu verwenden, um die Daten 401 für eine sichere verteilte Speicherung zu codieren.

[0064] In einem Beispiel codiert das sichere Datenverteilungsprogramm 200 (im Schritt 206) die Daten 401. In einer Ausführungsform wendet der Term K[i % key_size] der beispielhaften Codierungsfunktion einen Wert aus dem Datenschlüssel 403 unter den Elementen der zu codierenden Daten 401 zyklisch an. In dem veranschaulichenden Beispiel handelt es sich bei der Zeichenfolge 404 um die Ausgabe des auf den Schlüssel 403 angewendeten Terms K[i % key_size]. Auf der Grundlage der Terme K[0] und K[i] innerhalb des Paares beispielhafter Codierungsalgorithmen wird jedes Element der Zeichenfolge 404 jeweils auf einen Datenwert eines Elements der Daten 401 mit einem entsprechenden Indexwert angewendet. Zum Beispiel ist bei i = 6 K[6] = 3 und wird als Reaktion auf das Codieren eines Elements von Daten D[6] der Daten 401 (d.h. 17) verwendet.

[0065] In dem veranschaulichenden Beispiel stellen die codierten Daten 410 das ausgegebene sichere Datenverteilungsprogramm 200 dar, das die Daten 401 unter Verwendung des Schlüssels 403 und des Algorithmus ED1 und des Algorithmus ED2 codiert. In einer Ausführungsform entsprechen die Indexwerte von Elementen von Daten der Daten 401 auch den Indexwerten der Elemente von Daten der codierten Daten 410. In einigen Ausführungsformen stellt das sichere Datenverteilungsprogramm 200 (im Schritt 206) auch fest, dass mindestens einer der vorgeschriebenen Codierungsalgorithmen eine dateninterne Elementabhängigkeit enthält. Zum Beispiel enthalten die vorstehenden Codierungsalgorithmen dateninterne Elementabhängigkeiten, die zu einem vorherigen Codierungswert (d.h. E[i-1]) und einem entsprechenden vorherigen Wert für ein Element von Daten (d.h. D[i-1]) gehören.

[0066] In einer Ausführungsform stellt das sichere Datenverteilungsprogramm 200 (im Schritt 204) fest, dass der Wert, der der Menge an Standorten entspricht, die die zu speichernden verteilten Daten (d.h. die Daten 401) empfangen sollen, vier beträgt; daher beträgt die Menge der Teilgruppen von Daten oder Blöcken, die erzeugt werden sollen, vier (z.B. nb_chunks = 4). Des Weiteren stellt das sichere Datenverteilungsprogramm 200 als Reaktion darauf, dass das sichere Datenverteilungsprogramm 200 feststellt, dass ein zum Codieren der Daten 401 verwendeter Algorithmus (d.h. der Algorithmus ED2) eine oder mehrere dateninterne Elementabhängigkeiten enthält, fest, dass die codierten Daten 410 auf vier Blöcke verteilt sind und dass jeder Block von Daten daran gehindert wird, Datenelemente mit entsprechenden Indexwerten zu enthalten, die sequenziell sind, wie beispielsweise E[5] und E[6] der codierten Daten 410. In einigen Ausführungsformen ist der Indexwert „c“ eines Blocks jeweils einer Kennung (ID) zugeordnet, die dem Block von Daten für eine verteilte Speicherung entspricht, wie beispielsweise den Block-IDs des Elements 420.

[0067] In dem veranschaulichenden Beispiel ist die Block-ID 1 dem Block 430A zugeordnet, die Block-ID 2 ist dem Block 430B zugeordnet, die Block-ID 3 ist dem Block 430C zugeordnet, die Block-ID 4 ist dem Block 430D zugeordnet. In einigen Ausführungsformen stellt das sichere Datenverteilungsprogramm 200 fest, dass ein Satz von Sicherheitsalgorithmen, der zu der verteilten Speicherung von Daten 401 gehört, mindestens einen oder mehrere Daten-Chunking-Algorithmen enthält. Ein Aspekt eines Daten-Chunking-Algorithmus stellt auf der Grundlage eines Indexwerts, der dem Block entspricht, einen Größenwert fest, der einer Menge an Elementen entspricht, die innerhalb einer „Block-ID“ enthalten sind. In einer beispielhaften Ausführungsform wird durch den Algorithmus CD1 eine einem Block entsprechende Größe festgestellt. In einem Beispiel hat die zum Block 430B gehörende Block-ID 2 einen Block-Indexwert = 1 und einen entsprechenden Größenwert = 4, der anzeigt, dass vier Elemente der codierten Daten 410 dem Block 430B zugewiesen werden.

[0068] In einer Ausführungsform verteilt das sichere Datenverteilungsprogramm 200 codierte Daten 410 zyklisch auf vier Blöcke von codierten Daten. In dem dargestellten Beispiel verwendet das sichere Datenverteilungsprogramm 200 einen Daten-Chunking-Algorithmus, um Indexwerte zyklisch festzustellen, die den Elementen von codierten Daten innerhalb der codierten Daten 410 zur Verteilung auf die vier Blöcke von Daten entsprechen. Als Reaktion darauf verteilt das sichere Datenverteilungsprogramm 200 (im Schritt 208) die Elemente von codierten Daten, die den festgestellten Indexwerten entsprechen, auf den Block, der den Block-IDs entspricht. In dem veranschaulichenden Beispiel verwendet das sichere Datenverteilungsprogramm 200 den Algorithmus CD2, um die codierten Daten 410 auf Blöcke mit entsprechenden IDs von 1, 2, 3 und 4 zu verteilen; wobei „c“ dem Indexwert eines Blocks entspricht und „j“ den Indexwert darstellt, der einem Element innerhalb eines Blocks entspricht. Zum Beispiel ergibt das Eingeben der Indexwerte für das letzte Element des dritten Blocks von Daten (d.h. des Blocks 430C) in den Algorithmus CD2 C[2][3] = E[2+3*4] = E[14], was dem Wert von 21 innerhalb der codierten Daten 410 entspricht.

[0069] In dem veranschaulichenden Beispiel ist die zyklische Verteilung der codierten Daten 410 durch eine Schattierung dargestellt, die zu dem Element 420 gehörenden Block-IDs entspricht. In einigen Ausführungsformen verwendet das sichere Datenverteilungsprogramm 200 Block-IDs, die die Reihenfolge der Blöcke von codierten Daten angeben, welche zu den Daten 401 gehören. In einem Beispiel entsprechen Elemente von codierten Daten 410, die ohne Schattierung dargestellt sind, der Block-ID 1 (d.h. den Codierungswerten 17, 15, 20 und 2), mit gesprenkelter Schattierung dargestellte entsprechen der Block-ID 2 (d.h. den Codierungswerten 10, 7, 22 und 19), mit schraffierter Schattierung dargestellte entsprechen der Block-ID 3 (d.h. den Codierungswerten 22, 12, 0 und 21) und mit gesprenkelter Schattierung hoher Dichte dargestellte entsprechen der Block-ID 4 (d.h. den Codierungswerten 9, 9 und 14).

[0070] In dem veranschaulichenden Beispiel stellt der Block 430A den Satz von Elementen von codierten Daten und eine entsprechende Reihenfolge von Elementen von codierten Daten 410 der Block-ID 1 dar. Ebenso stellt der Block 430B den Satz von Elementen von codierten Daten und eine entsprechende Reihenfolge der Elemente von codierten Daten 410 der Block-ID 2 dar, der Block 430C stellt den Satz von Elementen von codierten Daten und eine entsprechende Reihenfolge der Elemente von codierten Daten 410 der Block-ID 3 dar und der Block 430D stellt den Satz von Elementen von codierten Daten und eine entsprechende Reihenfolge der Elemente von codierten Daten 410 der Block-ID 4 dar.

[0071] In verschiedenen Ausführungsformen erzeugt das sichere Datenverteilungsprogramm 200 zu den Daten 401 gehörende Metadaten in Bezug auf die verteilte Speicherung von codierten Daten 410. In dem veranschaulichenden Beispiel stellen Metadaten 431A Metadaten in Bezug auf den Block 430A dar. Ebenso stellen Metadaten 431B Metadaten in Bezug auf den Block 430B dar, Metadaten 431C stellen Metadaten in Bezug auf den Block 430C dar und Metadaten 431D stellen Metadaten in Bezug auf den Block 430D dar. In einer Ausführungsform erzeugt das sichere Datenverteilungsprogramm 200 Metadaten, die eine den ursprünglichen Daten (z.B. den Daten 401) entsprechende Daten-ID, eine Block-ID-Kennung und eine Positionskennung angeben. Die Metadaten 431 stellen ein Beispiel eines Metadaten-Schemas dar. In dem dargestellten Beispiel sind die Metadaten 431B „A2+25“, was anzeigt, dass der Block 430B der Daten-ID „A“ und der Block-ID „2“ zugeordnet ist; und „+“ ist ein Beispiel eines Datenbeschreibers, wobei die Informationen rechts des Beschreibers die Speicherkennung „25“ angeben, die zur Speicherung des entsprechenden Blocks von codierten Daten gehört. In verschiedenen Ausführungsformen entspricht eine Speicherkennung einer Speicher-ID und/oder einer Speicherposition innerhalb der Speicherinformationen 107, die zur Speicherung eines Blocks von codierten Daten ermittelt wurde/n (in **Fig.** 2 Schritt 204 und/oder Schritt 208). In einer Ausführungsform sind Metadaten in Bezug auf Blöcke von codierten Daten innerhalb der Speicherinformationen 107 enthalten und des Weiteren den ursprünglichen Daten für eine sichere verteilte Speicherung zugeordnet.

[0072] Es sei von vornherein klargestellt, dass das Umsetzen der hierin angeführten Lehren nicht auf eine Cloud-Computing-Umgebung beschränkt ist, obwohl diese Offenbarung eine ausführliche Beschreibung von Cloud-Computing umfasst. Stattdessen können Ausführungsformen der vorliegenden Erfindung gemeinsam mit jeder beliebigen weiteren Art von jetzt bekannter oder später erfundener Datenverarbeitungsumgebung umgesetzt werden.

[0073] Cloud-Computing ist ein Servicebereitstellungsmodell zum Ermöglichen eines problemlosen bedarfsgesteuerten Netzwerkzugriffs auf einen gemeinsam genutzten Pool von konfigurierbaren Datenverarbeitungsressourcen (z.B. Netzwerke, Netzwerkbandbreite, Server, Verarbeitung, Hauptspeicher, Speicher, Anwendungen, virtuelle Maschinen und Dienste), die mit minimalem Verwaltungsaufwand bzw. minimaler Interaktion mit einem Anbieter des Service schnell bereitgestellt und freigegeben werden können. Dieses Cloud-Modell kann mindestens fünf Eigenschaften umfassen, min-

destens drei Dienstmodelle und mindestens vier Implementierungsmodelle.

[0074] Bei den Eigenschaften handelt es sich um die folgenden:

On-Demand Self-Service: Ein Cloud-Nutzer kann einseitig automatisch nach Bedarf für Datenverarbeitungsfunktionen wie Serverzeit und Netzwerkspeicher sorgen, ohne dass eine menschliche Interaktion mit dem Anbieter der Dienste erforderlich ist.

Broad Network Access: Es sind Funktionen über ein Netzwerk verfügbar, auf die durch Standardmechanismen zugegriffen wird, welche die Verwendung durch heterogene Thin- oder Thick-Client-Plattformen (z.B. Mobiltelefone, Laptops und PDAs) unterstützen.

Resource-Pooling: Die Datenverarbeitungsressourcen des Anbieters werden zusammengeschlossen, um mehreren Nutzern unter Verwendung eines Multi-Tenant-Modells zu dienen, wobei verschiedene physische und virtuelle Ressourcen dynamisch nach Bedarf zugewiesen und neu zugewiesen werden. Es gibt eine gefühlte Standortunabhängigkeit, da der Nutzer allgemein keine Kontrolle bzw. Kenntnis über den genauen Standort der bereitgestellten Ressourcen hat, aber in der Lage sein kann, einen Standort auf einer höheren Abstraktionsebene festzulegen (z.B. Land, Staat oder Rechenzentrum).

Rapid Elasticity: Funktionen können für eine schnelle horizontale Skalierung (scale out) schnell und elastisch bereitgestellt werden, in einigen Fällen auch automatisch, und für ein schnelles Scale-in schnell freigegeben werden. Für den Nutzer erscheinen die für das Bereitstellen verfügbaren Funktionen häufig unbegrenzt und sie können jederzeit in jeder beliebigen Menge gekauft werden.

Measured Service: Cloud-Systeme steuern und optimieren die Verwendung von Ressourcen automatisch, indem sie eine Messfunktion auf einer gewissen Abstraktionsebene nutzen, die für die Art von Dienst geeignet ist (z.B. Speicher, Verarbeitung, Bandbreite sowie aktive Benutzerkonten). Die Nutzung von Ressourcen kann überwacht, gesteuert und gemeldet werden, wodurch sowohl für den Anbieter als auch für den Nutzer des verwendeten Dienstes Transparenz geschaffen wird.

[0075] Bei den Dienstmodellen handelt es sich um die folgenden:

Software as a Service (SaaS): Die dem Nutzer bereitgestellte Funktion besteht darin, die in einer Cloud-Infrastruktur laufenden Anwendungen des Anbieters zu verwenden. Die Anwendungen sind über eine Thin-Client-Schnittstelle wie einen Web-Browser (z.B. auf dem Web beruhende E-Mail) von verschiedenen Client-Einheiten her zugänglich. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, darunter das Netzwerk, Server, Betriebssysteme, Speicher bzw. sogar einzelne Anwendungsfunktionen, mit der möglichen Ausnahme von eingeschränkten benutzerspezifischen Anwendungskonfigurationseinstellungen.

Platform as a Service (PaaS): Die dem Nutzer bereitgestellte Funktion besteht darin, durch einen Nutzer erstellte bzw. erhaltene Anwendungen, die unter Verwendung von durch den Anbieter unterstützten Programmiersprachen und Tools erstellt wurden, in der Cloud-Infrastruktur einzusetzen. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, darunter Netzwerke, Server, Betriebssysteme bzw. Speicher, hat aber die Kontrolle über die eingesetzten Anwendungen und möglicherweise über Konfigurationen des Application Hosting Environment.

Infrastructure as a Service (IaaS): Die dem Nutzer bereitgestellte Funktion besteht darin, das Verarbeiten, Speicher, Netzwerke und andere grundlegende Datenverarbeitungsressourcen bereitzustellen, wobei der Nutzer in der Lage ist, beliebige Software einzusetzen und auszuführen, zu der Betriebssysteme und Anwendungen gehören können. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, hat aber die Kontrolle über Betriebssysteme, Speicher, eingesetzte Anwendungen und möglicherweise eine eingeschränkte Kontrolle über ausgewählte Netzwerkkomponenten (z.B. Host-Firewalls).

[0076] Bei den Einsatzmodellen handelt es sich um die folgenden:

Private Cloud: Die Cloud-Infrastruktur wird einzig und allein für eine Organisation betrieben. Sie kann durch die Organisation oder einen Dritten verwaltet werden und kann sich in den eigenen Räumen oder in fremden Räumen befinden.

Community Cloud: Die Cloud-Infrastruktur wird von mehreren Organisationen gemeinsam genutzt und unterstützt eine spezielle Benutzergemeinschaft, die gemeinsame Angelegenheiten hat (z.B. Mission, Sicherheitsanforderungen, Richtlinien sowie Überlegungen bezüglich der Einhaltung von Vorschriften). Sie kann durch die Organisationen oder einen Dritten verwaltet werden und kann sich in den eigenen Räumen oder in fremden Räumen befinden.

Public Cloud: Die Cloud-Infrastruktur wird der allgemeinen Öffentlichkeit oder einer großen Industriegruppe zur Verfügung gestellt und sie gehört einer Cloud-Dienste verkaufenden Organisation.

Hybrid Cloud: Die Cloud-Infrastruktur ist eine Zusammensetzung aus zwei oder mehreren Clouds (privat, Benutzergemeinschaft oder öffentlich), die zwar einzelne Einheiten bleiben, aber durch eine standardisierte oder proprietäre Technologie miteinander verbunden sind, die Daten- und Anwendungsportierbarkeit ermöglicht (z.B. Cloud-Zielgruppenverteilung für den Lastenausgleich zwischen Clouds).

[0077] Eine Cloud-Computing-Umgebung ist dienstorientiert mit Fokus auf Statusunabhängigkeit, geringer Kopplung, Modularität und semantischer Interoperabilität. Im Herzen von Cloud-Computing liegt eine Infrastruktur, die ein Netzwerk aus zusammengeschalteten Knoten aufweist.

[0078] **Fig.** 5 steht für verschiedene Datenverarbeitungseinheiten des Cloud-Computing-Knotens 10 (erörtert wird lediglich ein Beispiel eines geeigneten Cloud-Computing-Knotens und es soll keine Einschränkung hinsichtlich des Nutzungsumfangs oder der Funktionalität von hierin beschriebenen Ausführungsformen der Erfindung bedeuten). Ungeachtet dessen kann der Cloud-Computing-Knoten 10 ausgeführt werden und/oder jedwede Funktionalität von hierin beschriebenen Ausführungsformen der Erfindung durchführen.

[0079] Ungeachtet dessen kann der Cloud-Computing-Knoten 10 ausgeführt werden und/oder jedwede der vorstehend dargelegten Funktionalität durchführen. In beispielhaften Ausführungsformen steht der Cloud-Computing-Knoten 10 für das System 102. In verschiedenen Ausführungsformen kann der Cloud-Computing-Knoten 10 für physische Hardware-Instanzen von Hardware-Elementen und/oder Datenverarbeitungseinheiten (z.B. RISC-basierte Server 62, Server 63 usw.) und/oder eine virtualisierte Instanz von Hardware-Elementen, Datenverarbeitungseinheiten (z.B. virtuelle Server 71, virtuellen Speicher 72, virtuelle Netzwerke 73 usw.) stehen, die unter Bezugnahme auf **Fig.** 7 näher erörtert werden.

[0080] Wie in **Fig.** 5 gezeigt ist, ist das Computersystem 500 im Cloud-Computing-Knoten 10 in einigen Ausführungsformen in Form einer Universal-Datenverarbeitungseinheit wie beispielsweise des Systems 102 gezeigt. In weiteren Ausführungsformen steht das Computersystem 500 für eine oder mehrere mit der Cloud-Computing-Umgebung 50 verbundene Datenverarbeitungseinheiten, wie beispielsweise den Speicher 120A bis einschließlich des Speichers 120N. Das Computersystem 500 ist ein Beispiel eines Systems, das Software und Daten 510 enthält. Das Computersystem 500 enthält (einen) Prozessor(en) 501, einen Hauptspeicher 502, einen Cache 503, einen persistenten Speicher 505, eine Übertragungseinheit 507, (eine) Eingabe-/Ausgabe-(E/A-)Schnittstelle(n) 506 und eine Übertragungsstruktur 504.

[0081] Die Übertragungsstruktur 504 stellt Übertragungen zwischen dem Hauptspeicher 502, dem Cache 503, dem persistenten Speicher 505, der Übertragungseinheit 507 und (einer) Eingabe-/Ausgabe-(E/A-)Schnittstelle(n) 506 bereit. Die Übertragungsstruktur 504 kann mit einer beliebigen Architektur ausgeführt sein, die dafür ausgelegt ist, Daten und/oder Steuerinformationen zwischen Prozessoren (wie beispielsweise Mikroprozessoren, Übertragungs- und Netzwerkprozessoren usw.), dem Systemspeicher, peripheren Einheiten und beliebigen weiteren Hardware-Komponenten innerhalb eines Systems zu übergeben. Zum Beispiel kann die Übertragungsstruktur 504 mit einem oder mehreren Bussen ausgeführt sein, wie beispielsweise einem Peripheral-Component-Interconnect-(PCI-)Bus oder einem Kreuzschienenschalter.

[0082] Bei dem Hauptspeicher 502 und dem persistenten Speicher 505 handelt es sich um durch einen Computer lesbare Speichermedien. In dieser Ausführungsform enthält der Hauptspeicher 502 einen Direktzugriffsspeicher (RAM). Im Allgemeinen kann der Hauptspeicher 502 beliebige geeignete flüchtige oder nicht flüchtige, durch einen Computer lesbare Speichermedien enthalten. Der Cache 503 ist ein Schnellspeicher, der die Leistung des Prozessors/-der Prozessoren 501 verbessert, indem er Daten, auf die vor Kurzem zugegriffen wurde, sowie Daten, auf die nahezu vor Kurzem zugegriffen wurde, aus dem Hauptspeicher 502 hält.

[0083] Programmanweisungen und Daten, die verwendet werden, um Ausführungsformen der vorliegenden Erfindung in die Praxis umzusetzen, können im persistenten Speicher 505 und im Hauptspeicher 502 zur Ausführung durch einen oder mehrere der jeweiligen Prozessoren 501 über den Cache 503 gespeichert werden. In einer Ausführungsform enthält der persistente Speicher 505 ein Magnet-Festplattenlaufwerk. Alternativ oder zusätzlich zu einem Magnet-Festplattenlaufwerk kann der persistente Speicher 505 ein Solid-State-Festplattenlaufwerk, eine Halbleiterspeichereinheit, einen Nur-Lese-Speicher (ROM), einen löschbaren programmierbaren Nur-Lese-Speicher (EPROM), einen Flashspeicher oder ein beliebiges anderes durch einen Computer lesbares Speichermedium enthalten, das Programmanweisungen oder digitale Informationen speichern kann. In einer Ausführungsform ist der Speicher 103 in dem persistenten Speicher 505 enthalten.

[0084] Die durch den persistenten Speicher 505 verwendeten Datenträger können auch austauschbar sein. Zum Beispiel kann ein austauschbares Festplattenlaufwerk für den persistenten Speicher 505 verwendet werden. Zu weiteren Beispielen gehören optische und magnetische Platten, Thumb-Drives und Smartcards, die in ein Laufwerk zur Übertragung auf ein anderes durch einen Computer lesbares Speichermedium, das ebenfalls Teil des persistenten Speichers 505 ist, eingelegt werden. Software und Daten 510 werden im persistenten Speicher 505 zum Zugriff und/oder zur Ausführung durch einen oder mehrere der jeweiligen Prozessoren 501 über den Cache 503 und einen oder mehrere Hauptspeicher des Hauptspeichers 502 gespeichert. Unter Bezugnahme auf das System 102 enthalten die Software und Daten 510 die Daten 104, Datenschlüssel 105, Sicherheitsalgorithmen 106, Speicherinformationen 107, das sichere Datenverteilungsprogramm 200, das sichere Datenwiederherstellungsprogramm 300 und verschiedene Programme (nicht gezeigt). Unter Bezugnahme auf den Speicher 120A bis einschließlich des Speichers 120N enthalten die Software und Daten 510 jeweils verschiedene Programme und Daten (nicht gezeigt).

[0085] Die Übertragungseinheit 507 in diesen Beispielen ermöglicht Übertragungen mit anderen Datenverarbeitungssystemen oder -einheiten, darunter Ressourcen des Systems 102 und des Speichers 120A bis einschließlich des Speichers 120N. In diesen Beispielen enthält die Übertragungseinheit 507 eine oder mehrere Netzschnittstellenkarten. Die Übertragungseinheit 507 kann durch die Verwendung von physischen oder drahtlosen oder aber durch die Verwendung von sowohl physischen als auch drahtlosen Datenübertragungsverbindungen Übertragungen bereitstellen. Programmanweisungen und Daten, die verwendet werden, um Ausführungsformen der vorliegenden Erfindung in die Praxis umzusetzen, können durch die Übertragungseinheit 507 in den persistenten Speicher 505 heruntergeladen werden.

[0086] Die E/A-Schnittstelle(n) 506 ermöglicht/ermöglichen die Ein- und Ausgabe von Daten mit anderen Einheiten, die mit jedem Datenverarbeitungssystem wie beispielsweise einer Instanz des Knotens 10 verbunden sein können. Zum Beispiel kann/können die E/A-Schnittstelle(n) 506 eine Verbindung zu (einer) externen Einheit(en) 508 wie beispielsweise einer Tastatur, einem Tastenblock, einem berührungsempfindlichen Bildschirm und/oder einer anderen geeigneten Eingabeeinheit bereitstellen. Zu der/den externen Einheit(en) 508 können auch auswechselbare, durch einen Computer lesbare Speichermedien wie zum Beispiel Thumb-Drives, auswechselbare optische oder Magnetplatten und Speicherkarten gehören. Die Software und Daten 510, die verwendet werden, um Ausführungsformen der vorliegenden Erfindung in die Praxis umzusetzen, können auf solchen auswechselbaren, durch einen Computer lesbaren Speichermedien gespeichert und über (eine) E/A-Schnittstelle(n) 506 in den persistenten Speicher 505 geladen werden. Die E/A-Schnittstelle(n) 506 ist/sind auch mit dem Bildschirm 509 verbunden.

[0087] Die E/A-Schnittstelle(n) 506 ermöglicht/ermöglichen es auch einer oder mehreren Einheiten, mit dem Computersystem/Server 500 in Dialogverkehr zu treten; und/oder beliebigen Einheiten (z.B. Netzkarte, Modem usw.), die es dem Computersystem/Server 500 ermöglichen, mit einer oder mehreren anderen Datenverarbeitungseinheiten Daten auszutauschen. Ein solcher Datenaustausch kann über die E/A-Schnittstelle(n) 506 stattfinden. Dennoch kann das/der Computersystem/Server 500 mit einem oder mehreren Netzwerken wie beispielsweise einem lokalen Netz (LAN), einem allgemeinen Weitverkehrsnetz (WAN) und/oder einem öffentlichen Netz (z.B. dem Internet) über die Übertragungseinheit 507 Daten austauschen. Es sollte klar sein, dass auch andere Hardware- und/oder Software-Komponenten in Verbindung mit dem Computersystem/Server 500 verwendet werden könnten, obgleich diese nicht gezeigt sind. Zu Beispielen gehören, ohne darauf beschränkt zu sein: Mikrocode, Einheitentreiber, redundante Verarbeitungseinheiten, externe Anordnungen von Festplattenlaufwerken, RAID-Systeme, Bandlaufwerke und Speichersysteme zur Datenarchivierung usw.

[0088] Der Bildschirm 509 stellt einen Mechanismus bereit, um einem Benutzer Daten anzuzeigen, und kann zum Beispiel ein Computer-Monitor sein. Der Bildschirm 509 kann auch die Funktion eines berührungsempfindlichen Bildschirms wie beispielsweise des Bildschirms eines Tablet-Computers oder eines Smartphones übernehmen.

[0089] **Fig.** 6 stellt ein veranschaulichendes Beispiel der Cloud-Computing-Umgebung 50 dar. Wie gezeigt ist, enthält die Cloud-Computing-Umgebung 50 einen oder mehrere Cloud-Computing-Knoten 10, mit denen von Cloud-Nutzern verwendete lokale Datenverarbeitungseinheiten wie zum Beispiel ein elektronischer Assistent (PDA, personal digital assistant) oder ein Mobiltelefon 54A, ein Desktop-Computer 54B, ein Laptop-Computer 54C und/oder ein Automobil-Computer-System 54N Daten austauschen können. Die Knoten 10 können miteinander Daten austauschen. Sie können physisch oder virtuell in ein oder mehrere Netzwerke wie private, Benutzergemeinschafts-, öffentliche oder hybride Clouds gruppiert werden (nicht gezeigt), wie vorstehend beschrieben wurde, oder in eine Kombination daraus. Dies ermöglicht es der Cloud-Computing-Umgebung 50, Infrastruktur, Plattformen und/oder Software als Dienst anzubieten, für die ein Cloud-

Nutzer keine Ressourcen auf einer lokalen Datenverarbeitungseinheit vorhalten muss. Es sei darauf hingewiesen, dass die Arten von in **Fig.** 6 gezeigten Datenverarbeitungseinheiten 54A bis N lediglich veranschaulichend sein sollen und dass die Datenverarbeitungsknoten 10 und die Cloud-Computing-Umgebung 50 über eine beliebige Art Netzwerk und/oder über eine beliebige Art von über ein Netzwerk aufrufbarer Verbindung (z.B. unter Verwendung eines Web-Browsers) mit einer beliebigen Art von computergestützter Einheit Daten austauschen können. Zudem können gemäß Ausführungsformen der vorliegenden Erfindung eine Instanz des Cloud-Computing-Knotens 10 und der Datenverarbeitungseinheiten 54A bis N Komponenten enthalten, die unter Bezugnahme auf **Fig.** 5 zuvor dargestellt und ausführlicher beschrieben wurden.

[0090] Fig. 7 stellt veranschaulichend einen Satz von funktionalen Abstraktionsschichten dar, die durch die Cloud-Computing-Umgebung 50 (**Fig.** 6) bereitgestellt werden. Es sollte von vornherein klar sein, dass die in **Fig.** 7 gezeigten Komponenten, Schichten und Funktionen lediglich veranschaulichend sein sollen und Ausführungsformen der Erfindung nicht darauf beschränkt sind. Wie dargestellt ist, werden die folgenden Schichten und entsprechenden Funktionen bereitgestellt:

[0091] Eine Hardware- und Software-Schicht 60 enthält Hardware- und Software-Komponenten. Zu Beispielen für Hardware-Komponenten gehören: Mainframe-Computer 61; auf der RISC-(Reduced-Instruction-Set-Computer-)Architektur beruhende Server 62; Server 63; Blade-Server 64; Speichereinheiten 65; und Netzwerke sowie Netzwerkkomponenten 66. In einigen Ausführungsformen umfassen Software-Komponenten eine Netzwerk-Anwendungsserver-Software 67 und Datenbank-Software 68.

[0092] Eine Virtualisierungsschicht 70 stellt eine Abstraktionsschicht bereit, aus der die folgenden Beispiele für virtuelle Einheiten bereitgestellt werden können: virtuelle Server 71; virtueller Speicher 72; virtuelle Netzwerke 73, darunter virtuelle private Netzwerke; virtuelle Anwendungen und Betriebssysteme 74; und virtuelle Clients 75.

[0093] In einem Beispiel kann die Verwaltungsschicht 80 die nachstehend beschriebenen Funktionen bereitstellen. Eine Ressourcen-Bereitstellung 81 stellt die dynamische Beschaffung von Datenverarbeitungsressourcen sowie anderen Ressourcen bereit, die zum Durchführen von Aufgaben innerhalb der Cloud-Computing-Umgebung verwendet werden. Ein Messen und eine Preisfindung 82 stellen die Kostenverfolgung beim Verwenden von Ressourcen innerhalb der Cloud-Computing-Umgebung sowie die Abrechnung oder Rechnungsstellung für den Verbrauch dieser Ressourcen bereit. In einem Beispiel können zu diesen Ressourcen Anwendungs-Software-Lizenzen gehören. Ein Benutzerportal 83 stellt Nutzern und Systemadministratoren den Zugang zu der Cloud-Computing-Umgebung bereit. Eine Verwaltung des Dienstumfangs 84 stellt die Zuordnung und Verwaltung von Cloud-Computing-Ressourcen bereit, so dass die benötigten Dienstziele erreicht werden. Ein Planen und Erfüllen von Vereinbarungen zum Dienstumfang (SLA) 85 stellt die Anordnung vorab und die Beschaffung von Cloud-Computing-Ressourcen, für die eine zukünftige Anforderung vorausgesehen wird, gemäß einem SLA bereit. Die Sicherheit 86 stellt die Identitätsüberprüfung für Cloud-Nutzer und Aufgaben sowie Schutz für Daten und andere Ressourcen bereit. In verschiedenen Ausführungsformen enthält die Sicherheit 86 eine oder mehrere systembasierte Funktionen und/oder abonnierte Services (z.B. das sichere Datenverteilungsprogramm 200 und das sichere Datenwiederherstellungsprogramm 300), die Daten eines Benutzers oder Unternehmens schützen, indem codierte und in Blöcke aufgeteilte Daten auf eine Mehrzahl von Servern oder Speichersystemen der Hardware- und Software-Schicht 60 an verschiedenen physischen Positionen verteilt werden. In einigen Ausführungsformen verwendet die Sicherheit 86 weitere Aspekte der Verwaltungsschicht 80, um Zugriff auf Ressourcen innerhalb der Hardware- und Software-Schicht 60 von verschiedenen Lieferanten oder Cloud-Speicher-Anbietern zu erhalten, um die codierten und in Blöcke aufgeteilten Daten eines Benutzers oder Unternehmens zu speichern.

[0094] Eine Arbeitslastschicht 90 stellt Beispiele für die Funktionalität bereit, für welche die Cloud-Computing-Umgebung verwendet werden kann. Zu Beispielen für Arbeitslasten und Funktionen, die von dieser Schicht bereitgestellt werden können, gehören: Abbildung und Navigation 91; Software-Entwicklung und Lebenszyklusverwaltung 92; Bereitstellung von Ausbildung in virtuellen Klassenzimmern 93; Datenanalytikverarbeitung 94; Transaktionsverarbeitung 95; und Sicherheit 96. In verschiedenen Ausführungsformen enthält eine Datensicherheit 96 alternativ oder zusätzlich zur Sicherheit 86 eine oder mehrere Softwareanwendungen wie beispielsweise das sichere Datenverteilungsprogramm 200 und das sichere Datenwiederherstellungsprogramm 300, die ein Benutzer aufruft, um Daten innerhalb der Cloud zu sichern, indem codierte und in Blöcke aufgeteilte Daten auf eine Mehrzahl von Servern oder Speichersystemen der Hardware- und Softwareschicht 60 an verschiedenen physischen Positionen verteilt werden.

[0095] Die hierin beschriebenen Programme werden auf der Grundlage der Anwendung, für die sie in einer bestimmten Ausführungsform der Erfindung

ausgeführt werden, ermittelt. Es ist jedoch darauf hinzuweisen, dass jedwede bestimmte Programm-Nomenklatur hierin lediglich aus Gründen der Zweckmäßigkeit verwendet wird, und folglich sollte die Erfindung nicht auf eine ausschließliche Verwendung in einer bestimmten Anwendung, die von dieser Nomenklatur ermittelt und/oder durch diese Nomenklatur stillschweigend vorausgesetzt wird, beschränkt sein.

[0096] Bei der vorliegenden Erfindung kann es sich um ein System, ein Verfahren und/oder ein Computerprogrammprodukt handeln. Das Computerprogrammprodukt kann ein durch einen Computer lesbares Speichermedium (oder -medien) mit durch einen Computer lesbaren Programmanweisungen darauf umfassen, um einen Prozessor dazu zu veranlassen, Aspekte der vorliegenden Erfindung auszuführen.

[0097] Bei dem durch einen Computer lesbaren Speichermedium kann es sich um eine physische Einheit handeln, die Anweisungen zur Verwendung durch ein System zur Ausführung von Anweisungen behalten und speichern kann. Bei dem durch einen Computer lesbaren Speichermedium kann es sich zum Beispiel um eine elektronische Speichereinheit, eine magnetische Speichereinheit, eine optische Speichereinheit, eine elektromagnetische Speichereinheit, eine Halbleiterspeichereinheit oder jede geeignete Kombination daraus handeln, ohne auf diese beschränkt zu sein. Zu einer nicht erschöpfenden Liste spezifischerer Beispiele des durch einen Computer lesbaren Speichermediums gehören die folgenden: eine tragbare Computerdiskette, eine Festplatte, ein Direktzugriffsspeicher (RAM), ein Nur-Lese-Speicher (ROM), ein löschbarer programmierbarer Nur-Lese-Speicher (EPROM bzw. Flash-Speicher), ein statischer Direktzugriffsspeicher (SRAM), ein tragbarer Kompaktspeicherplatte-Nur-Lese-Speicher (CD-ROM), eine DVD (digital versatile disc), ein Speicher-Stick, eine Diskette, eine mechanisch kodierte Einheit wie zum Beispiel Lochkarten oder erhabene Strukturen in einer Rille, auf denen Anweisungen gespeichert sind, und jede geeignete Kombination daraus. Ein durch einen Computer lesbares Speichermedium soll in der Verwendung hierin nicht als flüchtige Signale an sich aufgefasst werden, wie zum Beispiel Funkwellen oder andere sich frei ausbreitende elektromagnetische Wellen, elektromagnetische Wellen, die sich durch einen Wellenleiter oder ein anderes Übertragungsmedium ausbreiten (z.B. Lichtwellenleiterkabel durchlaufende Lichtimpulse) oder durch einen Draht übermittelte elektrische Signale.

[0098] Hierin beschriebene, durch einen Computer lesbare Programmanweisungen können von einem durch einen Computer lesbaren Speichermedium auf jeweilige Datenverarbeitungs-/Verarbeitungseinheiten oder über ein Netzwerk wie zum Beispiel das Internet, ein lokales Netzwerk, ein Weitverkehrsnetz und/oder ein drahtloses Netzwerk auf einen externen Computer oder eine externe Speichereinheit heruntergeladen werden. Das Netzwerk kann Kupferübertragungskabel, Lichtwellenübertragungsleiter, drahtlose Übertragung, Leitwegrechner, Firewalls, Vermittlungseinheiten, Gateway-Computer und/oder Edge-Server aufweisen. Eine Netzwerkadapterkarte oder Netzwerkschnittstelle in jeder Datenverarbeitungs-/Verarbeitungseinheit empfängt durch einen Computer lesbare Programmanweisungen aus dem Netzwerk und leitet die durch einen Computer lesbaren Programmanweisungen zur Speicherung in einem durch einen Computer lesbaren Speichermedium innerhalb der entsprechenden Datenverarbeitungs-/Verarbeitungseinheit weiter.

[0099] Bei durch einen Computer lesbaren Programmanweisungen zum Ausführen von Arbeitsschritten der vorliegenden Erfindung kann es sich um Assembler-Anweisungen, ISA-Anweisungen (Instruction-Set-Architecture), Maschinenanweisungen, maschinenabhängige Anweisungen, Mikrocode, Firmware-Anweisungen, zustandssetzende Daten oder entweder Quellcode oder Objektcode handeln, die in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen geschrieben werden, darunter objektorientierte Programmiersprachen wie Smalltalk, C++ o.ä. sowie herkömmliche prozedurale Programmiersprachen wie die Programmiersprache „C“ oder ähnliche Programmiersprachen. Die durch einen Computer lesbaren Programmanweisungen können vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem fernen Computer oder vollständig auf dem fernen Computer oder Server ausgeführt werden. In letzterem Fall kann der entfernt angeordnete Computer mit dem Computer des Benutzers durch eine beliebige Art Netzwerk verbunden sein, darunter ein lokales Netzwerk (LAN) oder ein Weitverkehrsnetz (WAN), oder die Verbindung kann mit einem externen Computer hergestellt werden (zum Beispiel über das Internet unter Verwendung eines Internet-Dienstanbieters). In einigen Ausführungsformen können elektronische Schaltungen, darunter zum Beispiel programmierbare Logikschaltungen, vor Ort programmierbare Gatter-Anordnungen (FPGA, field-programmable gate arrays) oder programmierbare Logikanordnungen (PLA, programmable logic arrays) die durch einen Computer lesbaren Programmanweisungen ausführen, indem sie Zustandsinformationen der durch einen Computer lesbaren Programmanweisungen nutzen, um die elektronischen Schaltungen zu personalisieren, um Aspekte der vorliegenden Erfindung durchzuführen.

[0100] Aspekte der vorliegenden Erfindung sind hierin unter Bezugnahme auf Ablaufpläne und/oder Blockschaltbilder bzw. Schaubilder von Verfahren, Vorrichtungen (Systemen) und Computerprogrammprodukten gemäß Ausführungsformen der Erfindung beschrieben. Es wird darauf hingewiesen, dass jeder Block der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder sowie Kombinationen von Blöcken in den Ablaufplänen und/oder den Blockschaltbildern bzw. Schaubildern mittels durch einen Computer lesbare Programmanweisungen ausgeführt werden können.

[0101] Diese durch einen Computer lesbaren Programmanweisungen können einem Prozessor eines Universalcomputers, eines Spezialcomputers oder einer anderen programmierbaren Datenverarbeitungsvorrichtung bereitgestellt werden, um eine Maschine zu erzeugen, so dass die über den Prozessor des Computers bzw. der anderen programmierbaren Datenverarbeitungsvorrichtung ausgeführten Anweisungen ein Mittel zur Umsetzung der in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte erzeugen. Diese durch einen Computer lesbaren Programmanweisungen können auch auf einem durch einen Computer lesbaren Speichermedium gespeichert sein, das einen Computer, eine programmierbare Datenverarbeitungsvorrichtung und/oder andere Einheiten so steuern kann, dass sie auf eine bestimmte Art funktionieren, so dass das durch einen Computer lesbare Speichermedium, auf dem Anweisungen gespeichert sind, ein Herstellungsprodukt aufweist, darunter Anweisungen, welche Aspekte der/des in dem Block bzw. den Blöcken des Ablaufplans und/oder der Blockschaltbilder bzw. Schaubilder angegebenen Funktion/Schritts umsetzen.

[0102] Die durch einen Computer lesbaren Programmanweisungen können auch auf einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder eine andere Einheit geladen werden, um das Ausführen einer Reihe von Prozessschritten auf dem Computer bzw. der anderen programmierbaren Vorrichtung oder anderen Einheit zu verursachen, um einen auf einem Computer ausgeführten Prozess zu erzeugen, so dass die auf dem Computer, einer anderen programmierbaren Vorrichtung oder einer anderen Einheit ausgeführten Anweisungen die in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte umsetzen.

[0103] Die Ablaufpläne und die Blockschaltbilder bzw. Schaubilder in den Figuren veranschaulichen die Architektur, die Funktionalität und den Betrieb möglicher Ausführungen von Systemen, Verfahren und Computerprogrammprodukten gemäß verschiedenen Ausführungsformen der vorliegenden Erfindung. In diesem Zusammenhang kann jeder Block in den Ablaufplänen oder Blockschaltbildern bzw. Schaubildern ein Modul, ein Segment oder einen Teil von Anweisungen darstellen, die eine oder mehrere ausführbare Anweisungen zur Ausführung der bestimmten logischen Funktion(en) aufweisen. In einigen alternativen Ausführungen können die in dem Block angegebenen Funktionen in einer anderen Reihenfolge als in den Figuren gezeigt stattfinden. Zwei nacheinander gezeigte Blöcke können zum Beispiel in Wirklichkeit im Wesentlichen gleichzeitig ausgeführt werden, oder die Blöcke können manchmal je nach entsprechender Funktionalität in umgekehrter Reihenfolge ausgeführt werden. Es ist ferner anzumerken, dass jeder Block der Blockschaltbilder bzw. Schaubilder und/oder der Ablaufpläne sowie Kombinationen aus Blöcken in den Blockschaltbildern bzw. Schaubildern und/oder den Ablaufplänen durch spezielle auf Hardware beruhende Systeme umgesetzt werden können, welche die festgelegten Funktionen oder Schritte durchführen, oder Kombinationen aus Spezial-Hardware und Computeranweisungen ausführen.

[0104] Die Beschreibungen der verschiedenen Ausführungsformen der vorliegenden Erfindung erfolgten zum Zweck der Veranschaulichung, sollen jedoch nicht erschöpfend oder auf die offenbarten Ausführungsformen beschränkt sein. Viele Änderungen und Varianten sind für den Fachmann erkennbar, ohne vom Umfang und Wesen der Erfindung abzuweichen. Die hierin verwendete Terminologie wurde gewählt, um die Grundgedanken der Ausführungsform, die praktische Anwendung oder technische Verbesserung gegenüber auf dem Markt befindlicher Technologien am besten zu erklären bzw. um anderen Fachleuten das Verständnis der hierin offenbarten Ausführungsformen zu ermöglichen.

Patentansprüche

1. Verfahren zum sicheren Speichern von Daten (104), wobei das Verfahren aufweist:
Empfangen, durch einen oder mehrere Computerprozessoren (501), einer Anforderung für das Speichern von Daten (104), wobei die Daten (104) eine Mehrzahl von Elementen aufweisen;
Umwandlung, durch den einen oder die mehreren Computerprozessoren (501), der Mehrzahl von Elementen der Daten (104) in eine Mehrzahl von entsprechenden numerischen Werten, wobei die Mehrzahl der entsprechenden numerischen Werte jeweils mit Indexwerten assoziiert sind, und wobei der Indexwert [i] bei einem Indexwert von Null beginnt;
Erzeugen (206), durch einen oder mehrere Computerprozessoren (501), einer Mehrzahl von Elementen von codierten Datenwerten, durch Anwenden eines oder mehrerer Codierungsalgorithmen auf die Mehrzahl entsprechender numerischer Werte

der umgewandelten Mehrzahl von Elementen der Daten, auf Grundlage von einem oder mehreren Diktaten, wobei
(i) die Menge der Mehrzahl von Elementen von codierten Daten gleich einer Menge der Mehrzahl von umgewandelten Elementen der Daten ist;
(ii) die Mehrzahl der Elemente der codierten Datenwerte als numerische Werte ausgedrückt wird;
(iii) ein erster Codierungsalgorithmus ein erstes Element aus der Mehrzahl von Elementen codierter Datenwerte, der einem Indexwert von [i]=0 zugeordnet ist, von einem ersten Diktat ausschließt;
(iv) das erste Diktat den ersten Codierungsalgorithmus anweist, den numerischen Wert eines codierten Datenelementwertes, der mit dem Indexwert [i-1] assoziiert ist, und den numerischen Wert eines umgewandelten Elementes der Daten aus der Mehrzahl der Elemente der Daten, der dem Indexwert [i -1] zugeordnet ist, zu verwenden, um einen codierten Datenwert zu erzeugen, der einem umgewandelten Element der Daten der Mehrzahl der Elemente der Daten, das dem Indexwert [i] zugeordnet ist, entspricht;
Verteilen (208), durch einen oder mehrere Computerprozessoren (501), der Mehrzahl von Elementen der codierten Daten (410) auf zwei oder mehrere Teilgruppen von codierten Datenwerten ohne Duplizierung; und
Übertragen (210), durch einen oder mehrere Computerprozessoren (501), der zwei oder mehreren Teilgruppen der codierten Datenwerte an eine entsprechende Menge von zwei oder mehreren Speichersystemen (120A, 120N),
wobei das Verteilen (208) der Mehrzahl von Elementen der codierten Datenwerte auf die zwei oder mehreren Teilgruppen der codierten Datenwerte ohne Duplizierung des Weiteren aufweist:
Feststellen, durch einen oder mehrere Computerprozessoren (501), auf Grundlage einer Struktur von mindestens einem ersten Codierungsalgorithmus, dass ein Wert, der einem vorherigen Element der codierten Daten entspricht, verwendet wird, um ein nachfolgendes Element der Daten der Mehrzahl von Elementen der Daten zu codieren; und
Verteilen, durch einen oder mehrere Computerprozessoren, der Mehrzahl von Elementen der codierten Datenwerte auf die zwei oder mehreren Teilgruppen der codierten Datenwerte, so dass das nachfolgende Element der codierten Datenwerte einer Teilgruppe von codierten Datenwerte zugewiesen wird, die sich von einer weiteren Teilgruppe der codierten Datenwerte unterscheidet, welche das vorherige Element der codierten Datenwerte enthält, das verwendet wird, um das nachfolgende Element der Mehrzahl der Elemente der codierten Datenwerte zu codieren.

2. Verfahren nach Anspruch 1, das des Weiteren aufweist:
Festlegen, durch einen oder mehrere Computerprozessoren (501), auf Grundlage einer Sicherheitsvorschrift, das Codieren der Mehrzahl von Elementen der Daten (104) weiter zu verschleiern;
Auswählen, durch einen oder mehrere Computerprozessoren (501), eines Datenschlüssels (403), der während des Codierens der Mehrzahl von Elementen der Daten angewendet wird, wobei der Datenschlüssel (403) zwei oder mehrere Elemente aufweist, und wobei eine Anzahl von numerischen Werten, die in dem Datenschlüssel (403) enthalten sind, kleiner als die Anzahl an Datenelementen ist, die in der umgewandelten Mehrzahl von Elementen der Daten enthalten ist; und
Anwenden, durch einen oder mehrere Computerprozessoren, eines numerischen Wertes, der aus dem Datenschlüssel zyklisch ausgewählt ist, auf die umgewandelte Mehrzahl von Werten der Daten durch Anwenden des ersten Codierungsalgorithmus, wobei das zyklische Auswählen des numerischen Wertes aus dem Datenschlüssel (403) ferner ein Erzeugen eines Elementes E[i] aus der Mehrzahl der codierten Datenwerte mit dem Index [i] gemäß folgender Formel aufweist:
E[i] = (E[i-1] + D[i-1] + D[i] + K[i % key_size]) % max_value, wobei Folgendes gilt:
E[i]: codierter Datenelementwert für den Index [i],
D[i]: zu codierender Datenelementwert für den Index [i],
K[i]: Schlüsselelementwert für den Index [i] und
max_value: ein numerischer Wert, der um mindestens einen Wert von 1 größer als ein maximale Wert eines beliebigen Elements von zu codierenden Elementen von Datenelementwerten ist.

3. Verfahren nach Anspruch 1, wobei das Verteilen der Mehrzahl von Elementen der codierten Datenwerte auf die zwei oder mehreren Teilgruppen der codierten Datenwerte ohne Duplizierung des Weiteren aufweist:
Verteilen, durch einen oder mehrere Computerprozessoren (501), auf zyklische Weise von nachfolgenden Elementen der Mehrzahl von Elementen der codierten Datenwerte (410) auf die zwei oder mehreren Teilgruppen der codierten Datenwerte (410).

4. Verfahren nach Anspruch 1, wobei es sich bei den zwei oder mehreren Speichersystemen (120A, 120N) um Speichersysteme (120A, 120N) an unterschiedlichen Standorten handelt.

5. Computerprogrammprodukt zum sicheren Speichern von Daten, wobei das Computerprogrammprodukt aufweist:
ein oder mehrere durch einen Computer (500) lesbare Speichermedien und Programmanweisungen, die auf dem einen oder den mehreren durch einen Computer lesbaren Speichermedien gespeichert sind, wobei die Programmanweisungen durch einen oder mehrere Computerprozessoren (501)

lesbar sind, um das Verfahren gemäß einem der Ansprüche 1 bis 4 auszuführen.

6. Computersystem (500) zum sicheren Speichern von Daten (104), wobei das Computersystem (500) aufweist:
einen oder mehrere Computerprozessoren (501);
ein oder mehrere durch einen Computer lesbare Speichermedien; und
Programmanweisungen, die auf den durch einen Computer lesbaren Speichermedien zur Ausführung durch mindestens einen des einen oder der mehreren Computerprozessoren (501) gespeichert sind, um das Verfahren gemäß einem der Ansprüche 1 bis 4 auszuführen.

Es folgen 7 Seiten Zeichnungen

Anhängende Zeichnungen

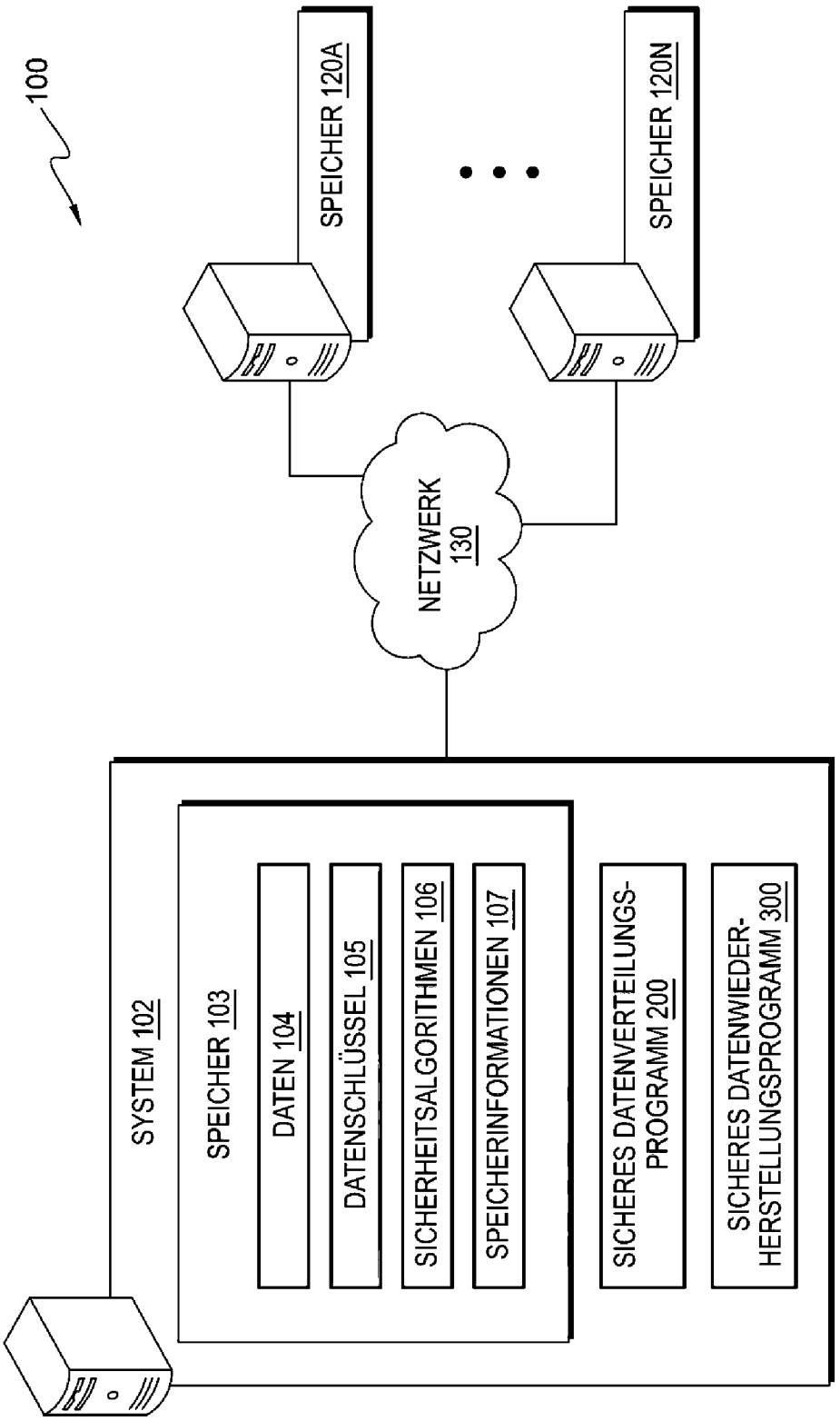


FIG. 1

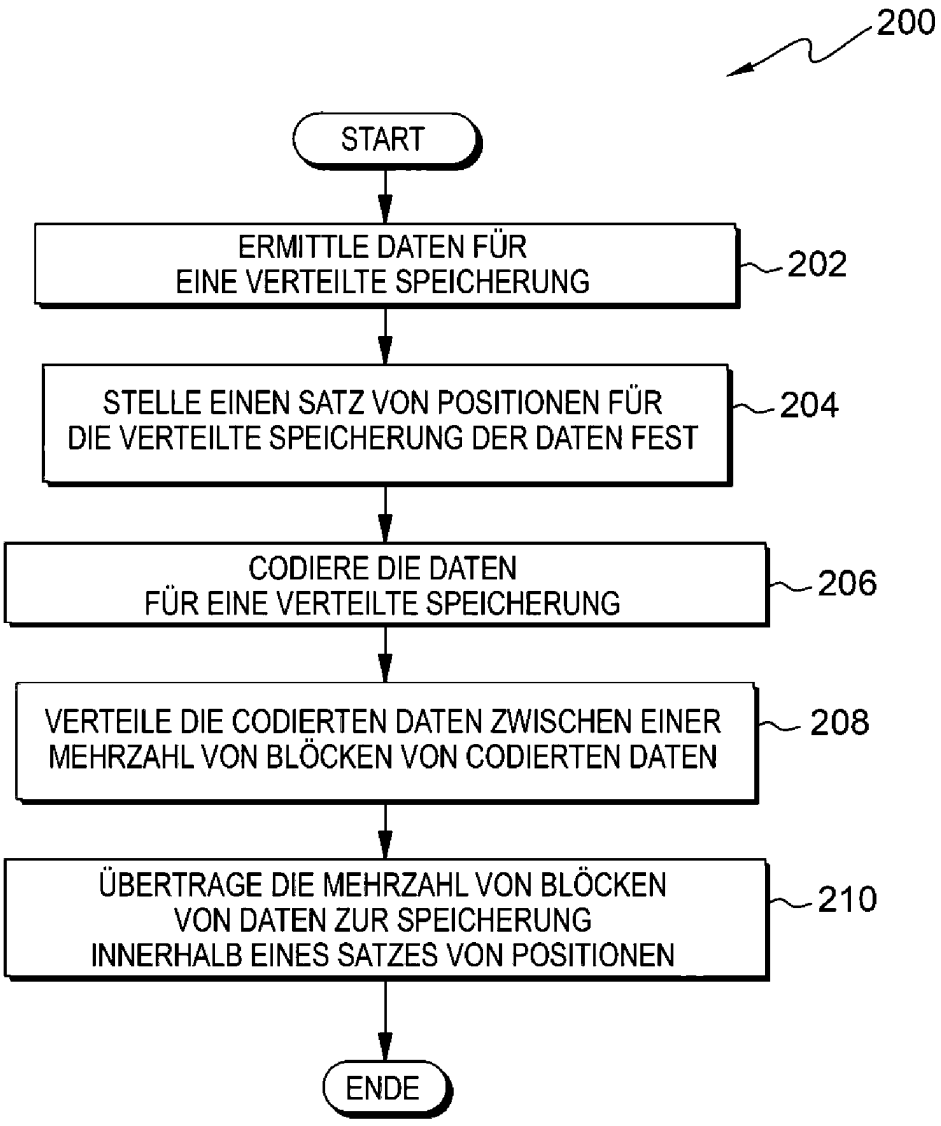

200
START
ERMITTLE DATEN FÜR EINE VERTEILTE SPEICHERUNG
202
STELLE EINEN SATZ VON POSITIONEN FÜR DIE VERTEILTE SPEICHERUNG DER DATEN FEST
204
CODIERE DIE DATEN FÜR EINE VERTEILTE SPEICHERUNG
206
VERTEILE DIE CODIERTEN DATEN ZWISCHEN EINER MEHRZAHL VON BLÖCKEN VON CODIERTEN DATEN
208
ÜBERTRAGE DIE MEHRZAHL VON BLÖCKEN VON DATEN ZUR SPEICHERUNG INNERHALB EINES SATZES VON POSITIONEN
210
ENDE

FIG. 2

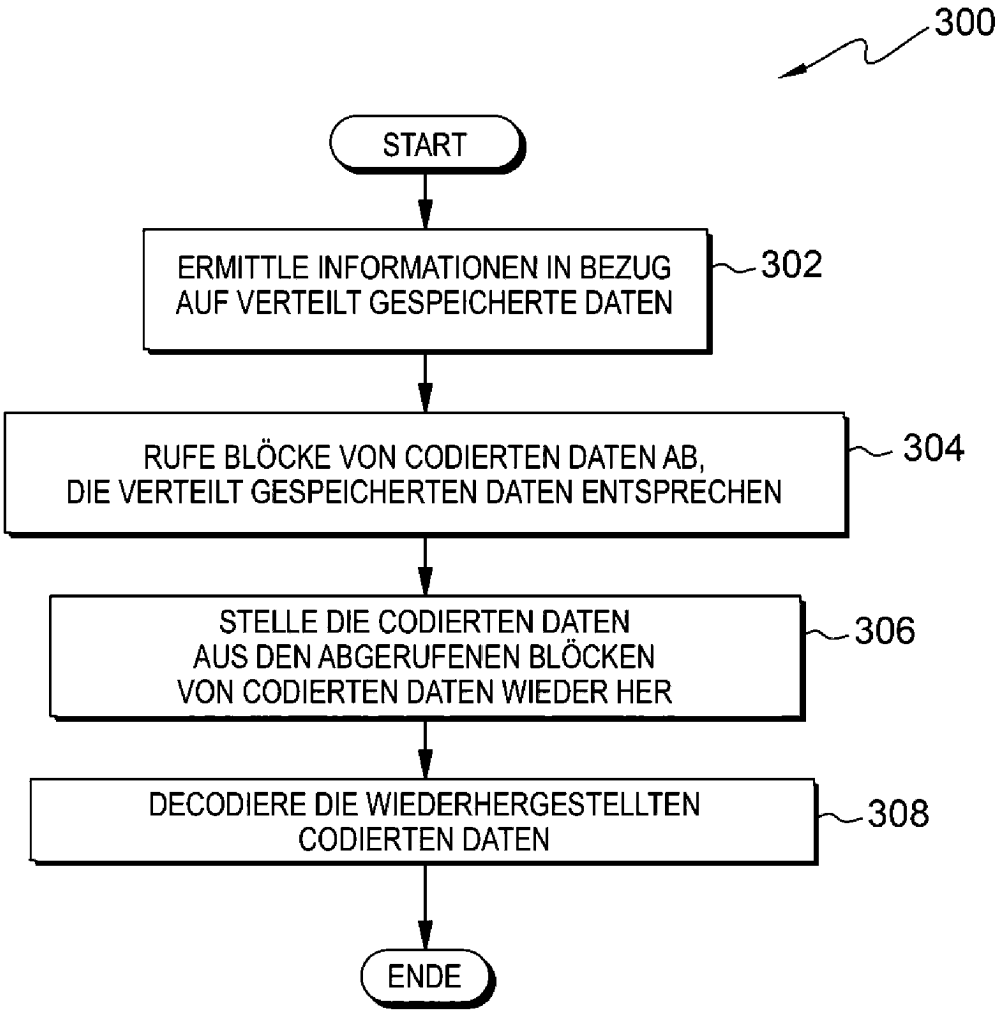

300
START
ERMITTLE INFORMATIONEN IN BEZUG AUF VERTEILT GESPEICHERTE DATEN
302
RUFE BLÖCKE VON CODIERTEN DATEN AB, DIE VERTEILT GESPEICHERTEN DATEN ENTSPRECHEN
304
STELLE DIE CODIERTEN DATEN AUS DEN ABGERUFENEN BLÖCKEN VON CODIERTEN DATEN WIEDER HER
306
DECODIERE DIE WIEDERHERGESTELLTEN CODIERTEN DATEN
308
ENDE

FIG. 3

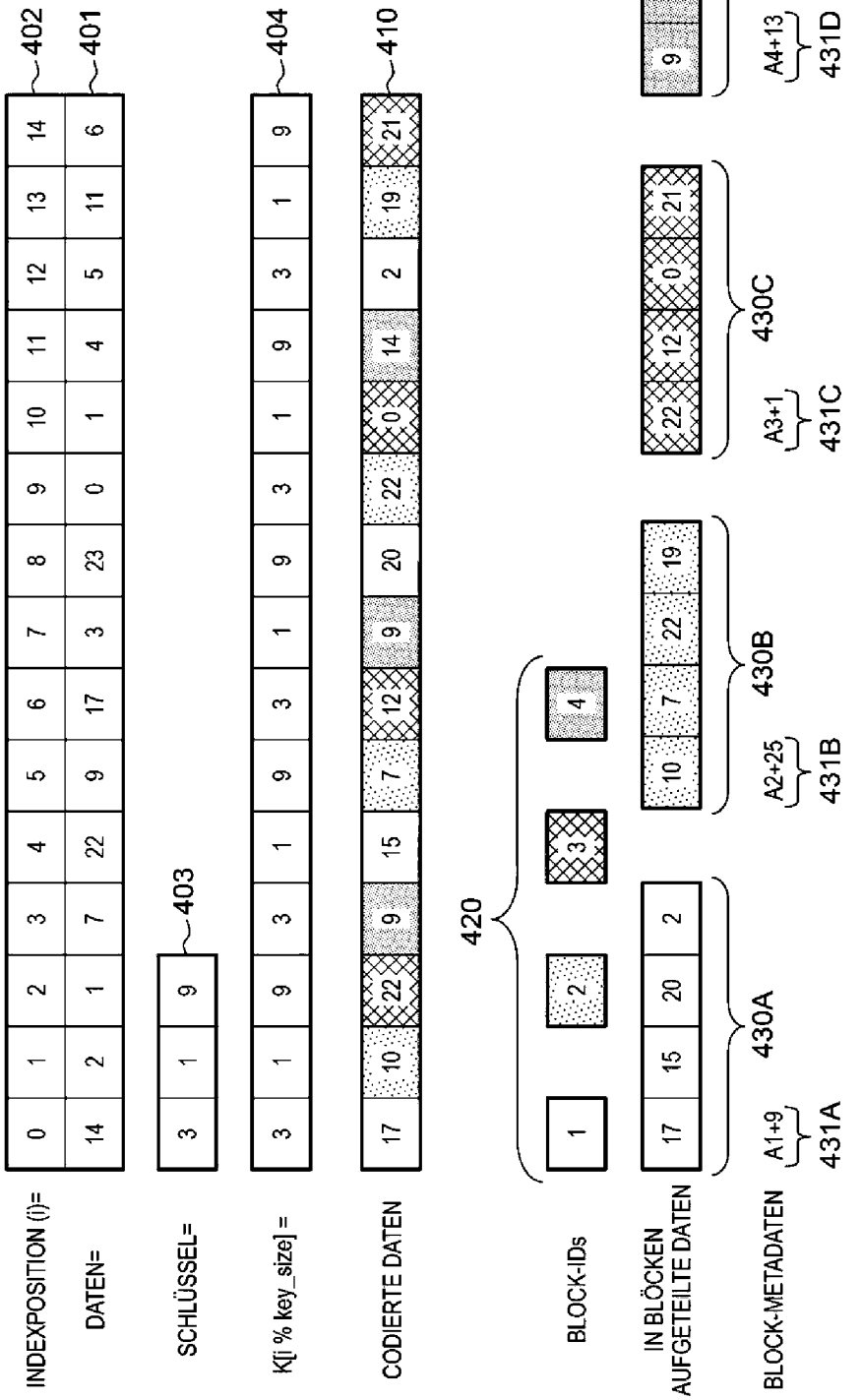

INDEXPOSITION (i)=
DATEN=
SCHLÜSSEL=
K[i % key_size] =
CODIERTE DATEN
BLOCK-IDs
IN BLÖCKEN AUFGETEILTE DATEN
BLOCK-METADATEN
402
401
403
404
410
420
430A
430B
430C
430D
431A
431B
431C
431D
A1+9
A2+25
A3+1
A4+13

FIG. 4

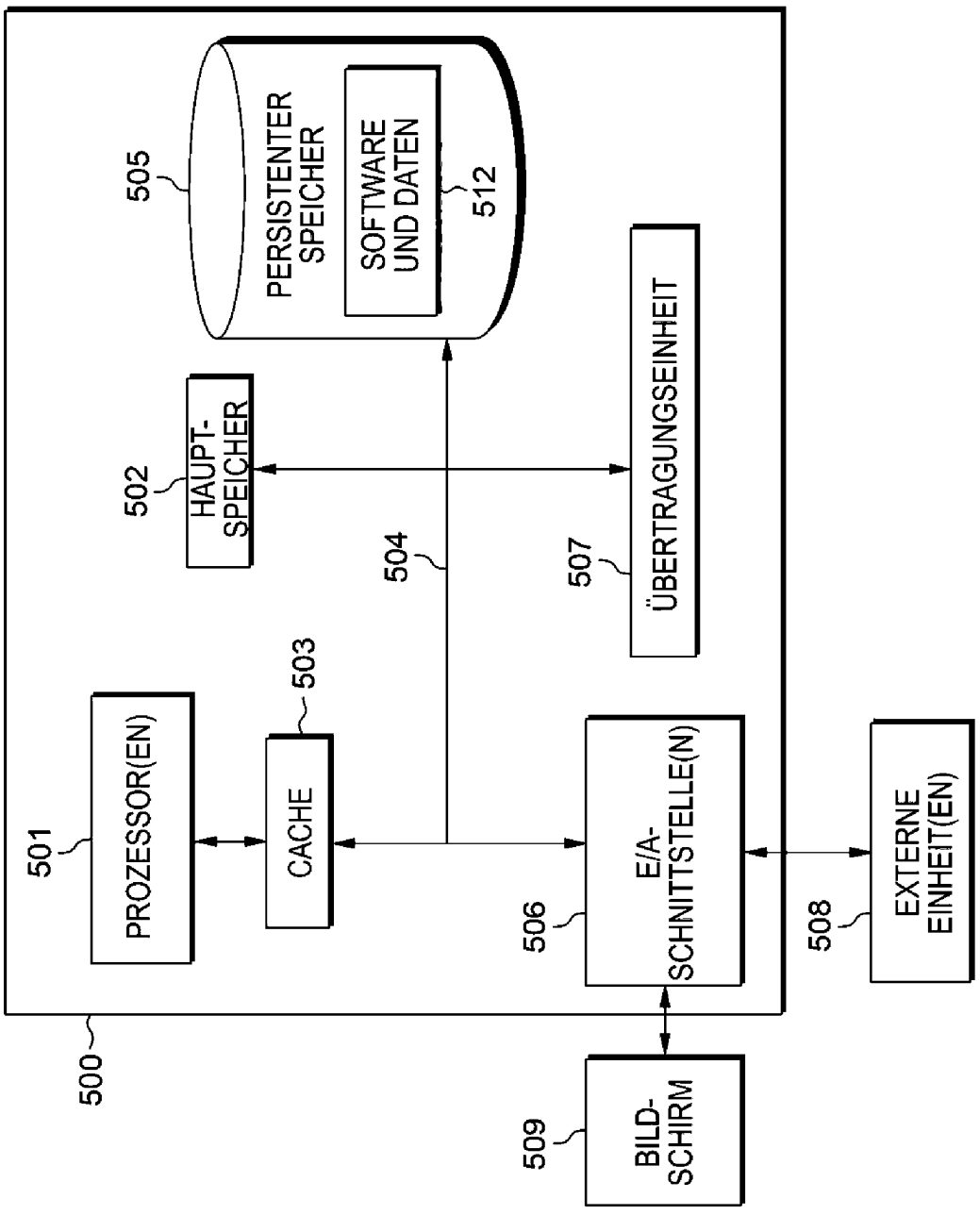

10
500
501
PROZESSOR(EN)
503
CACHE
504
505
PERSISTENTER SPEICHER
SOFTWARE UND DATEN
512
502
HAUPT-SPEICHER
507
ÜBERTRAGUNGSEINHEIT
506
E/A-SCHNITTSTELLE(N)
508
EXTERNE EINHEIT(EN)
509
BILD-SCHIRM

FIG. 5

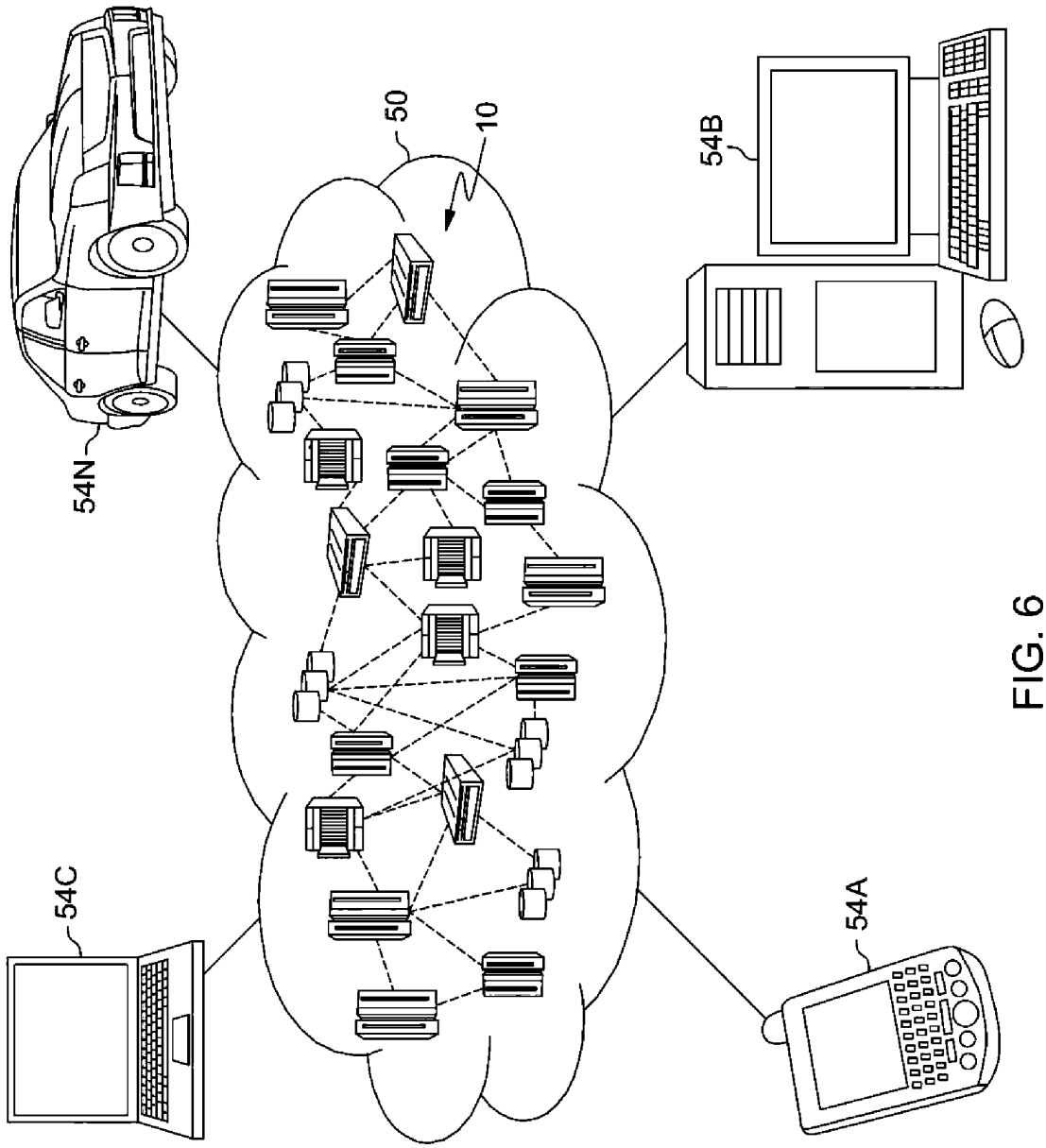


FIG. 6

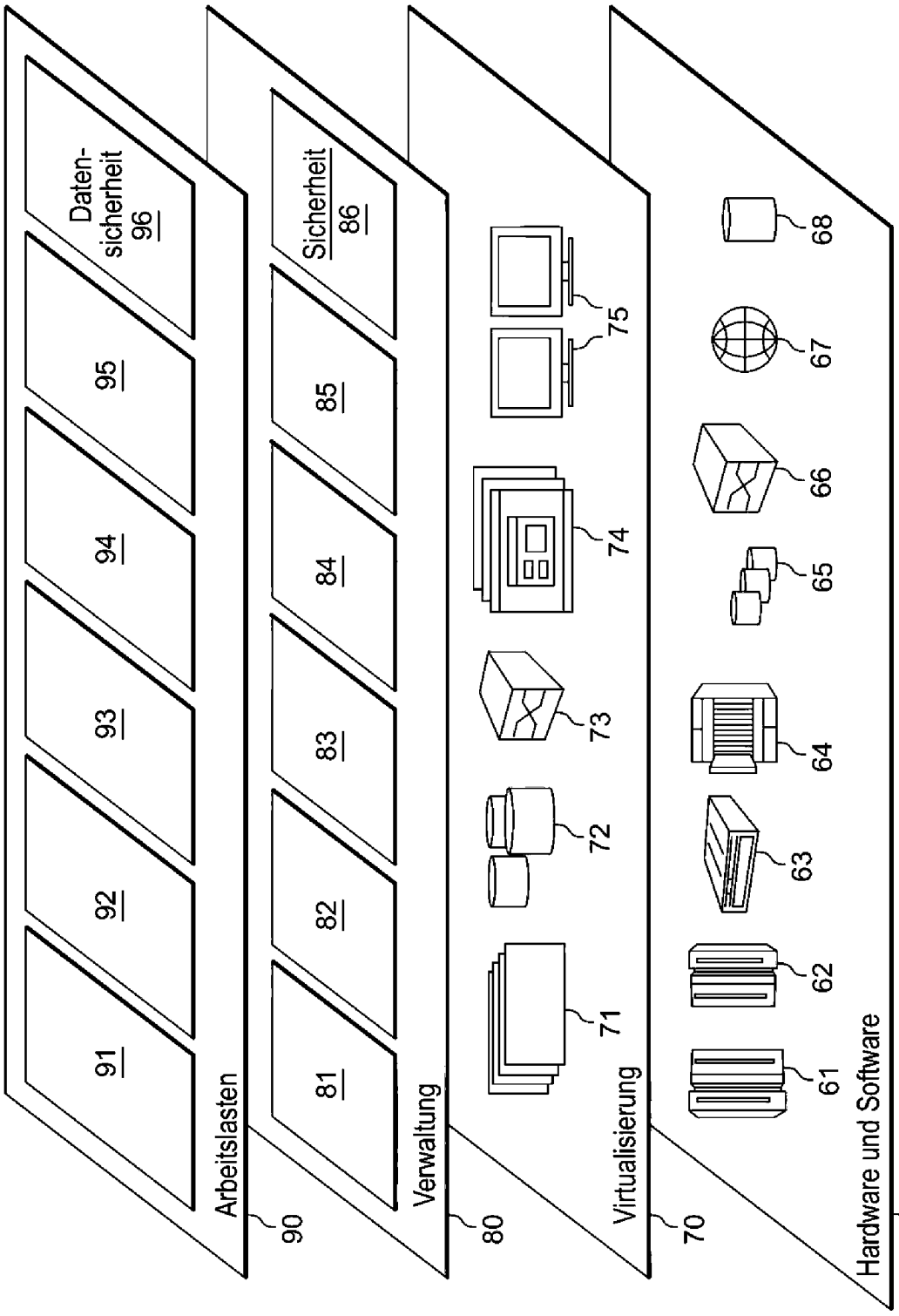


FIG. 7

 Das Dokument wurde durch die Firma Luminess hergestellt.