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

(10) **DE 699 35 913 T2** 2008.01.10

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 092 297 B1**
(21) Deutsches Aktenzeichen: **699 35 913.9**
(86) PCT-Aktenzeichen: **PCT/US99/15146**
(96) Europäisches Aktenzeichen: **99 940 799.2**
(87) PCT-Veröffentlichungs-Nr.: **WO 2000/002342**
(86) PCT-Anmeldetag: **02.07.1999**
(87) Veröffentlichungstag der PCT-Anmeldung: **13.01.2000**
(97) Erstveröffentlichung durch das EPA: **18.04.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **25.04.2007**
(47) Veröffentlichungstag im Patentblatt: **10.01.2008**

(51) Int Cl.[8]: ***G07F 7/10*** (2006.01)
***H04L 9/08*** (2006.01)

(30) Unionspriorität:
**91644 P** **02.07.1998** **US**

(84) Benannte Vertragsstaaten:
**AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE**

(73) Patentinhaber:
**Cryptography Research Inc., San Francisco, Calif., US**

(72) Erfinder:
**KOCHER, Paul C., 575 Market Street San Francisco,CA 94105, US**

(74) Vertreter:
**Kuhnen & Wacker Patent- und Rechtsanwaltsbüro, 85354 Freising**

(54) Bezeichnung: **LECKRESISTENTE AKTUALISIERUNG EINES INDEXIERTEN KRYPTOGRAPHISCHEN SCHLÜSSELS**

## Beschreibung

### BEREICH DER ERFINDUNG

**[0001]** Die vorliegende Erfindung betrifft Systeme zum sicheren Verwalten und Benutzen von kryptografischen Keys und spezieller Verfahren und Vorrichtungen zum Sichern kryptografischer Geräte (Devices) gegen externe Mithörattacken.

### HINTERGRUND DER ERFINDUNG

**[0002]** Angreifer, die sich Zugang zu kryptografischen Keys und anderen Geheimnissen verschaffen, können potentiell unbefugte Vorgänge ausführen oder Transaktionen fälschen. Daher müssen in vielen Systemen wie z.B. elektronischen Zahlungssystemen auf Chipkartenbasis Geheimnisse in eingriffssicherer Hardware geschützt werden. Kürzliche Arbeiten durch Cryptography Research haben jedoch gezeigt, dass Chipkarten und andere Devices kompromittiert werden können, wenn Informationen über kryptografische Geheimnisse zu Angreifern gelangen, die die externen Kennwerte wie Leistungsaufnahme oder elektromagnetische Strahlung von Devices überwachen.

**[0003]** Sowohl in symmetrischen als auch in asymmetrischen Kryptosystemen müssen geheime Parameter geheim gehalten werden, da ein Angreifer, der einen Key kompromittiert, Kommunikationen entschlüsseln, Unterschriften fälschen, unbefugte Transaktionen ausführen, sich als ein Benutzer ausgeben oder andere Probleme verursachen kann. Methoden zum sicheren Verwalten von Keys unter Verwendung von physisch sicheren, gut abgeschirmten Räumen sind im Stand der Technik bekannt und werden heutzutage weithin eingesetzt. Früher bekannte Methoden zum Schützen von Keys in kostenarmen kryptografischen Devices waren jedoch häufig für viele Anwendungen unzureichend, z.B. solche mit herausfordernden technischen Beschränkungen (Kosten, Größe, Leistung usw.) oder solche, die ein hohes Maß an Eingriffssicherheit benötigen. Attacken wie Rückwärtsentwicklung von ROM mit Mikroskopen, Timing-Attacken-Kryptoanalyse (siehe z.B. P. Kocher in „Timing an Implementations of Diffie-Hellman, RSA, DSS and Other Systems", Advances in Cryptology – CRYPTO '96, Springer-Verlag, Seiten 104-113), und Fehleranalyse (siehe z.B. E. Biham und A. Shamir in „Differential Fault Analysis of Secret Key Cryptosystems", Advances in Cryptology – CRYPTO '97, Springer-Verlag 1997, Seiten 513-525) wurden zum Analysieren von Kryptosystemen beschrieben.

**[0004]** Es sind im Stand der Technik Key-Management-Techniken bekannt, die verhindern, dass Devices kompromittierende Angreifer frühere Keys ableiten können. So definiert z.B. ANSI X9.24, „Financial Services – Retail Management" ein Protokoll, das als Derived Unique Key Per Transaction (DUKPT) bekannt ist und verhindert, dass Angreifer frühere Keys nach vollständigem Kompromittieren des Zustands eines Device ableiten können. Solche Techniken können zwar verhindern, dass Angreifer alte Keys ableiten, aber sie haben praktische Beschränkungen und bieten keinen effektiven Schutz. gegen externe Mithörattacken, bei denen Angreifer partielle Informationen über aktuelle Keys nutzen, um zukünftige zu kompromittieren.

**[0005]** Cryptography Research hat auch Methoden zum Verwenden von wiederholten Hash-Operationen entwickelt, um es einem Client und Server zu ermöglichen, kryptografische Operationen auszuführen, während sich der Client gegen externe Mithörattacken selbst schützt. Bei solchen Methoden wendet der Client wiederholt eine kryptografische Funktion auf sein internes Geheimnis zwischen oder während Transaktionen an, so dass in jeder aus einer Reihe von Transaktionen ausleckende Informationen nicht zum Kompromittieren des Geheimnisses kombiniert werden können.

**[0006]** Dieses soeben beschriebene System hat jedoch den Nachteil, dass der Server eine ähnliche Folge von Operationen ausführen muss, um den in jeder Transaktion verwendeten symmetrischen Session-Key erneut abzuleiten. So kann der Server in Fällen wie z.B. denen, bei denen eine große Zahl von unsychronisierten Server-Devices (z.B. elektronische Bargeldanwendungen, bei denen eine große Zahl von Händlerterminals als unabhängige Server arbeiten) vorliegt oder wenn Server eine begrenzte Speicherkapazität haben, nicht alle möglichen Session-Keys, die Clients benutzen könnten, zuverlässig vorberechnen. Infolgedessen kann die Transaktionsleistung herabgesetzt werden, da möglicherweise eine relativ große Zahl von Operationen nötig ist, damit der Server den richtigen Session-Key erhält. So kann beispielsweise der n-te Client-Session-Key n Server-Operationen zum Ableiten benötigen. Daher wäre ein schnelles und effizientes Verfahren zum Erhalten von leckbeständiger und/oder lecksicherer symmetrischer Key-Vereinbarung von Vorteil.

**[0007]** Das US-Patent 5,633,930 beschreibt ein Speicherwertzahlungssystem, bei dem die Bankkarte (SVC = Stored Value Card) mit einem begrenzten Satz von Device-Keys ausgegeben wird (die in dem Patent abgeleitete Keys genannt werden). Der spezielle Device-Key, der in jeder Transaktion zu verwenden ist, wird extern ausgewählt. Es wird ein Key-Diversifizierungsverfahren beschrieben, das zum Ableiten von Transaktions-Keys von einem Haupt-Secret-Key verwendet werden kann (z.B. einem Device-Key), um zu verhindern, dass Angreifer, die sich einen oder mehrere Transaktions-Keys beschaffen, den Haupt-Device-Key ermitteln können. Die US-Pa-

tente 5,544,086 und 5,559,887 haben im Wesentlichen denselben Inhalt wie 5,633,930.

**[0008]** Das US-Patent 5,761,306 beschreibt ein Online-Protokoll zum Ersetzen eines Public-Key durch einen anderen, wobei der alte Key zum Signieren des Austauschs verwendet wird. Dieses Key-Austauschsystem soll von einer zentralen Behörde verwendet werden, um einen designierten revidierten Key zu einer großen Zahl von Devices zu senden. Empängergeräte können den aktuellen Public-Key zum Verifizieren einer digitalen Signatur auf dem Ersatz vor dem Akzeptieren eines Updates verwenden.

**[0009]** Das US-Patent 4,203,166 beschreibt ein vernetztes kryptografisches Dateisystem. Das beschriebene System beinhaltet die Fähigkeit, zwischen Domains übertragene File-Keys zu verschlüsseln. Es werden Low-Level-Details einer Hardware-Implementation gegeben, inklusive Verfahren zum Ver- und Entschlüsseln von zwischen Servern ausgetauschten Keys.

**[0010]** Die europäische Patentanmeldung 0 582 395 beschreibt die Sicherung von über ein Netzwerk gesendeten Paketen durch Verwenden eines „Pufferwarteschlangenindex"-Wertes, der in dem Paket zum Authentifizieren und Ableiten von Paket-Keys enthalten ist. Das beschriebene Protokoll verwendet statische „Host-zu-Host-Keys", die an jeder Transaktion beteiligt sind und nicht aktualisiert werden.

**[0011]** Die europäische Patentanmeldung 0 529 261 beschreibt ein Online-Protokoll für die Verwendung eines Public-Key-Kryptosystems zum Verteilen eines symmetrischen Key zusammen mit assoziierten Gebrauchsregeln. Diese Regeln sollen gewährleisten, dass Empfänger, die diese einhalten, Keys nicht falsch verwenden.

## ZUSAMMENFASSUNG DER ERFINDUNG

**[0012]** Die Erfindung ist in ihren verschiedenen Aspekten in den nachfolgenden Hauptansprüchen definiert, auf die nunmehr Bezug genommen werden sollte. Vorteilhafte Merkmale sind in den Unteransprüchen dargelegt.

**[0013]** Die vorliegende Erfindung kann benutzt werden, um Chipkarten (und andere kryptografische Client-Devices) selbst dann sicher zu machen, wenn Angreifer in der Lage sind, externe Mithör- (oder sonstige) Attacken zum Sammeln von Informationen in Bezug auf interne Vorgänge des Client-Device auszuführen. In einer Ausgestaltung führt ein kryptografisches Client-Device (z.B. eine Chipkarte) einen Secret-Key-Wert als Teil seines Zustands. Der Client kann seinen Geheimwert jederzeit, z.B. vor jeder Transaktion, mit einem Update-Prozess aktualisieren, der bewirkt, dass partielle Informationen, die möglicherweise zuvor an Angreifer über das Geheimnis ausgeleckt sind, den neuen aktualisierten Geheimwert nicht mehr (oder zu einem geringeren Grad) nützlich beschreiben. (Informationen werden dann als nützlich angesehen, wenn sie einem Angreifer helfen können oder es ihm ermöglichen, eine tatsächliche Attacke auszuführen.) So wird der Secret-Key-Wert oft genug aktualisiert (möglicherweise sogar einmal pro Transaktion), so dass über den Eingangszustand ausleckende Informationen den aktualisierten Zustand nicht so nützlich beschreiben. Durch wiederholtes Anwenden des Update-Prozesses veralten bei kryptografischen Operationen ausleckende Informationen, die von Angreifern gesammelt werden, schnell. So kann ein solches System gegen Attacken sicher bleiben, die wiederholte Messungen der Leistungsaufnahme oder der elektromagnetischen Kennwerte des Device beinhalten, selbst dann, wenn das System mit undichter Hardware und Software implementiert ist (d.h. bei denen Informationen über die Geheimwerte auslecken). Im Gegensatz dazu verwenden herkömmliche Systeme denselben Geheimwert wiederholt und ermöglichen es Angreifern so, von einer großen Zahl von Transaktionen gesammelte Informationen statistisch zu kombinieren.

**[0014]** Die vorliegende Erfindung kann in Verbindung mit einem ersten Device (z.B. einem Client) und einem zweiten Device (z.B. einem Server) unter Verwendung eines solchen Protokolls benutzt werden. Zur Ausführung einer Transaktion mit dem Client holt der Server den aktuellen Transaktionszählwert des Client (oder einen anderen Key-Indexwert) ein. Der Server führt dann eine Reihe von Operationen durch, um die Folge von Transformationen zu ermitteln, die zum erneuten Ableiten des richtigen Session-Key vom anfänglichen Geheimwert des Client benötigt werden. Diese Transformationen werden dann ausgeführt und das Ergebnis wird als Transaktions-Session-Key (oder zum Ableiten eines Session-Key) genutzt.

**[0015]** Die vorliegende Erfindung kann eine Folge von Clientseitigen Aktualisierungsprozessen beinhalten, die erhebliche Verbesserungen der Leistung der entsprechenden Server-Operationen zulassen und dabei leckbeständige und/oder lecksichere Sicherheitskennwerte im Client-Device beibehalten. In einer Ausgestaltung der Erfindung wird jeder Prozess in der Sequenz aus zwei kryptografischen Vorwärtstransformationen ($F_A$ und $F_B$) und deren Umkehrungen ($F_A^{-1}$ und $F_B^{-1}$) ausgewählt. Mit nachfolgend ausführlich beschriebenen Methoden werden solche Update-Funktionen vom Client in einer Sequenz angewendet, die gewährleistet, dass jeder einzelne Geheimwert niemals öfter als eine festgelegte Anzahl (z.B. drei) von Malen benutzt oder abgeleitet wird. Ferner gewährleisten die Update-Funktionen und die Sequenz auch, dass der Zustand (und demgemäß

der darin verwendete Geheim-Session-Key-Wert) einer beliebigen Transaktion effizient von einem Anfangszustand (z.B. dem in der ersten Transaktion verwendeten Zustand) innerhalb einer geringen Anzahl von Anwendungen von $F_A$ und $F_B$ (oder deren Umkehrungen) ableitbar ist.

**[0016]** Wenn die Zahl der Operationen, die von einem Client sicher ausgeführt werden können, n ist (d.h. n unterschiedliche Transaktionen können ausgeführt werden, ohne dass derselbe Geheimwert mehr als eine feste Anzahl von Malen verwendet wird), dann kann ein Server, der den anfänglichen Geheimwert K (oder den diesem entsprechenden Anfangszustand) kennt oder ihn beschaffen kann, einen resultierenden Geheimwert (oder entsprechenden Zustand) in der Serie von Transaktionen erheblich schneller ableiten als durch Ausführen von n entsprechenden Updates. In der Tat kann der Zustand für eine gegebene Transaktion häufig von einem Server mit 0(log n) Berechnungen von $F_A$ und $F_B$ (oder deren Umkehrungen) abgeleitet werden. Wenn der Systemdesigner n groß genug gemacht hat, dann kann es dadurch möglich werden, dass ein praktisch grenzenloser Satz von Transaktionen von Clients ausgeführt wird, während ausgezeichnete Serverleistung erzielt wird. So kann n beispielsweise größer als 50 sein.

## KURZBESCHREIBUNG DER ZEICHNUNGEN

**[0017]** Die Erfindung wird nun ausführlicher beispielhaft mit Bezug auf die Begleitzeichnungen beschrieben. Dabei zeigt:

**[0018]** Fig. 1 eine beispielhafte Ausgestaltung eines Key-Update-Prozesses über eine Reihe von Transaktionen;

**[0019]** Fig. 2 einen beispielhaften Client-seitigen indexierten Key-Update-Prozess;

**[0020]** Fig. 3 einen beispielhaften Server-Prozess zum Ableiten eines Transaktions-Key von einem Key-Index und einem Basis-Key;

**[0021]** Fig. 4 beispielhafte Ausgestaltungen von vier Zustandstransformationsoperationen.

## AUSFÜHRLICHE BESCHREIBUNG DER ERFINDUNG

### Indexiertes Key-Management

**[0022]** Die Erfindung ermöglicht es, dass Personen kryptografische Operationen mit erhöhter Sicherheit gegen externe Mithörattacken ausführen. Es werden zwar Ausführungsbeispiele unter Beteiligung von zwei Parteien an einem ersten Device (z.B. einem „Client") und einem zweiten Device (z.B. einem „Server") beschrieben, aber die Begriffe „Client" und „Server" werden der Einfachheit halber benutzt und entsprechen nicht unbedingt direkt einer besonderen Rolle in einem Systemdesign. So könnte der Client beispielsweise eine Chipkarte und der Server ein Zentralrechner sein oder umgekehrt. Ferner könnten zwar die meisten kryptografischen Operationen zwei Parteien beinhalten (z.B. einen beim Client und einen beim Server), aber die Erfindung kann natürlich auch in Umgebungen zum Einsatz kommen, die nur eine Partei beinhalten (wie z.B. in sicheren Memory- oder Speichersystemen, in denen sowohl Client als auch Server von einer einzigen Partei gesteuert werden oder in einem einzelnen Device kombiniert sind) oder in Umgebungen, die mehr als zwei Parteien und/oder Devices beinhalten.

**[0023]** In einer beispielhaften Ausgestaltung wird der Client mit einem Secret-Key $K_0$ für ein symmetrisches Kryptosystem initialisiert, wobei $K_0$ dem Server ebenfalls bekannt ist (oder von ihm abgeleitet werden kann). Der Key $K_0$ ist gewöhnlich (aber nicht unbedingt) für ein(e) bestimmte(s) Client-Device oder Partei spezifisch. Der Client hat auch einen (typischerweise nicht geheimen) Index oder Transaktionszähler C, der auf null initialisiert werden kann. Ein zusätzlicher Parameter ist eine Indextiefe D. Der Wert von D kann ebenfalls nichtgeheim sein und kann (z.B.) clientspezifisch oder eine systemweite globale Konstante sein. Der Wert von D bestimmt die Zykluslänge des Key-Update-Prozesses.

**[0024]** Ein die Erfindung ausgestaltender beispielhafter Computer-implementierter Prozess soll ein erstes Device (Client) sichern und gleichzeitig Transaktionen mit wenigstens einem zweiten Device (Server) ausführen. Das erste Device beinhaltet einen rechnerlesbaren Speicher, der einen internen Geheimzustand beinhaltet, der einen Key repräsentiert oder von dem er abgeleitet werden kann. Das zweite Device hat Zugang zu einem Basis- oder Anfangswert, der dem Anfangswert des internen Geheimzustands im Speicher des ersten Device entspricht. Ein Indexparameter ist mit dem internen Geheimzustand des Client assoziiert und dieser Indexparameter kann durch mehrere Werte aktualisiert werden. Es sind im Update-Prozess mehrere Transformationsoperationen verfügbar, die verwendet werden können. Das Client-Device benutzt den Indexparameter im Speicher, um eine der Transformationsoperationen auszuwählen. In einem Ausführungsbeispiel gibt es wenigstens zwei Transformationen und die Umkehrungen der beiden Transformationen. Jede Trans formationsoperation wirkt auf den internen Geheimzustand ein, um einen aktualisierten Geheimzustand zu erzeugen. Wenn dies erfolgt ist, wird im Speicher der interne Geheimzustand durch den aktualisierten Geheimzustand ersetzt und der Indexparameter wird durch einen aktualisierten Indexparameter ersetzt. Jetzt wird eine kryptografische Transaktion an Daten

am Client- (ersten) Device ausgeführt, die für das zweite Device gedacht sind, um die Daten unter Verwendung des aktualisierten Geheimzustands zu sichern. Die Operation am ersten Device, insbesondere die Wahl einer Transformation und die Aktualisierung, wird mehrere wiederholt, z.B. vor jeder Transaktion. Die Werte für den aktualisierten internen Geheimzustand werden vom Client-Device nicht mehr als eine feste Anzahl von Malen, in diesem Ausführungsbeispiel dreimal, rekreiert. Die gesicherten Daten werden dann zusammen mit dem Indexparameter zum zweiten Device gesendet. Die gesicherten Daten und der Indexparameter werden am Server empfangen. Unter Verwendung des Indexparameters wird der aktualisierte Geheimzustand jetzt vom Basis- oder Anfangsgeheimzustand erzeugt, zu dem das zweite Device Zugang hat. Der aktualisierte Geheimzustand wird dann benutzt, um die empfangenen gesicherten Daten zu verarbeiten. Das Server-Device für eine solche nachfolgende Transformation kann die Regeneration des aktualisierten Geheimzustands in wesentlich weniger Transformationen als die Gesamtzahl der Wiederholungen von Auswahl und Aktualisierung am ersten Device ausführen. Wie dies erzielt wird, wird nun beschrieben.

**[0025]** Fig. 1 zeigt eine beispielhafte Folge von Client-Device-Geheimzustandswerten, die zum Ausführen einer Reihe von Transaktionen benutzbar sind, typischerweise (aber nicht unbedingt) unter Verwendung von einem Zustand pro Transaktion. (Der zum Erzeugen der Sequenz verwendete Client-Prozess wird mit Bezug auf Fig. 2 beschrieben, der entsprechende Server-Prozess wird mit Bezug auf Fig. 3 beschrieben.) Ein Geheimwert eines Zustands beinhaltet typischerweise, aber nicht unbedingt, einen Secret-Session-Key; daher wird, der Einfachheit halber, der Geheimwert mit K bezeichnet und der Begriff „Geheimwert" kann mit „Key" mehr oder weniger austauschbar verwendet werden. Die Fachperson wird jedoch erkennen, dass sie sich im allgemeinen Fall unterscheiden können. Ebenso zeigt die Figur der Deutlichkeit halber einen beispielhaften Key-Update-Prozess mit D=5, was bedeutet, dass fünf Ebenen von Key-Werten vorhanden sind. Es gibt jedoch keine spezielle Begrenzung für D und die Fachperson wird leicht verstehen, wie die dem Ausführungsbeispiel zu Grunde liegenden allgemeinen Prinzipien auch auf andere solche Zykluslängen anwendbar sind. In der Tat würden kommerziell eingesetzte Systeme normalerweise größere Werte für D verwenden.

**[0026]** Jeder dieser Kästen in der Figur repräsentiert einen Wert des Geheimwertes ($K_C$). Somit repräsentieren mehrere Punkte in einer Box unterschiedliche Zustände, die denselben Geheimwert $K_C$ gemeinsam nutzen. Die obere Reihe (Reihe 0) der Figur enthält eine Box, die dem Anfangszustand $K_0$ **110** sowie nachfolgenden Zuständen $K_{30}$ **140** und $K_{60}$ **170** entspricht, die alle demselben Geheimwert $K_C$ nutzen. Die nächste Reihe (Reihe 1) enthält zwei Boxen, von denen die linke einem Trio von Zuständen ($K_1$ **111**, $K_{15}$ und $K_{29}$) entspricht, die denselben Geheimwert nutzen, und die rechte Box in der zweiten Reihe entspricht einem zweiten Trio von Zuständen ($K_{31}$, $K_{45}$ und $K_{59}$), die noch einen anderen Geheimwert gemeinsam nutzen. Ebenso enthält Reihe **2** vier Boxen, die insgesamt zwölf Zustände repräsentieren, von denen 4 Trios jeweils denselben Geheimwert gemeinsam nutzen. Allgemeiner, in dieser beispielhaften Ausgestaltung enthält Reihe N (wobei N < D-1 ist) $2^N$ Boxen (oder eindeutige Geheimwerte) und $3(2^N)$ Zustände und die letzte Reihe (N=D-1) enthält $2^N$ Boxen und $2^N$ Zustände. Der dickere (gekrümmte) Pfad stellt schematisch den Prozess dar, mit dem die Zustände aktualisiert werden, beginnend mit dem Anfangszustand **110** und weiterführend bis zum Endzustand **170**. Während die Zustände aktualisiert werden, wird der Zähler C aktualisiert (um eins für jedes Update).

**[0027]** Die beispielhaften Zustands-Update-Prozesse beinhalten zwei Funktionen ($F_A$ und $F_B$) und deren Umkehrungen ($F_A^{-1}$ und $F_B^{-1}$) für insgesamt vier Funktionen. In Schritt **100** wird der Client mit einem Startzähler C=0 und einem Startzustand mit einem Anfangsgeheimwert $K_C = K_0$ initialisiert oder personalisiert. In Schritt **110** führt das Device die erste Transaktion mit $K_C$ (oder einem von $K_C$ abgeleiteten Key) durch. Der Key kann in praktisch jeder symmetrischen kryptografischen Transaktion verwendet werden. (So könnte eine solche Transaktion beispielsweise, ohne Begrenzung, die Berechnung oder Verifizierung eines MAC (Message Authentication Code) an einer Meldung, die Ver- oder Entschlüsselung einer Meldung, die Erzeugung eines pseudozufälligen Challenge-Wertes, die Ableitung eines Key usw. beinhalten. Beispiele für Meldungen sind z.B., ohne Begrenzung, Daten, die die Beträge von Fondstransferoperationen vorgeben, Email-Meldungen, Challenge/Response-Authentifizierungsdaten, Parameter-Update-Autorisierungen, Code-Updates, Audiomeldungen, digitalisierte Bilder usw.).

**[0028]** Nach Schritt **110** wird der Geheimwert $K_C$ des Client-Device durch Anwenden der Funktion $F_A$ aktualisiert und der Zähler C wird inkrementiert, d.h. durch Ausführen von $C \leftarrow C + 1$ und $K_C \leftarrow F_A(K_C)$. (Somit sind in Schritt **111** C = 1 und $K_C = F_A(K_0)$.) Der aktualisierte Wert von $K_C$ wird zum Ausführen einer Transaktion in Schritt **111** benutzt. Nach Schritt **111** wird C wieder inkrementiert und $F_A$ wird erneut auf $K_C$ angewendet, d.h. durch Ausführen von $C \leftarrow C + 1$ und $K_{C=2} \leftarrow F_A(K_C)$, was den in Schritt **112** verwendeten Secret-Key ergibt. Dasselbe Paar Operationen ($C \leftarrow C + 1$ und $K_C \leftarrow F_A(K_C)$) wird ebenso zwischen den Schritten **112** und **113** sowie zwischen den Schritten **113** und **114** angewendet.

**[0029]** Die Transaktion in Schritt **115** sollte densel-

ben Wert von $K_C$ verwenden wie auch die Transaktion in Schritt **113**, da die Schritte **113** und **115** in derselben Box erscheinen. So erfolgt nach der Transaktion in Schritt **114** der Update-Prozess durch Berechnen von $C \leftarrow C + 1$ (ergibt C=5) und $K_{C=5} \leftarrow F_A^{-1}(K_C)$. Man beachte, dass $K_{C=5} = F_A^{-1}(K_{C=4}) = F_A^{-1}(F_A(K_{C=3})) = K_{C=3}$ ist. So ist der in Schritt **115** verwendete Wert von $K_C$ derselbe wie der in Schritt **113** verwendete Wert. Nach der Transaktion in Schritt **115** wird $K_C$ mit der Funktion $F_B$ durch Inkrementieren von C und Berechnen von $K_{C=6} \leftarrow F_B(K_C)$ aktualisiert. Nach der Transaktion in Schritt **116** wird der Geheimwert für Transaktion **117** durch Anwenden der Funktion $F_B^{-1}$ auf $K_C$ berechnet.

**[0030]** Der Update-Prozess läuft so ab, dass nach jeder Transaktion ein Key-Zustands-Update-Prozess ausgeführt wird. Das Key-Update beinhaltet das Inkrementieren von C und die Anwendung einer der Funktionen $F_A$, $F_B$, $F_A^{-1}$ oder $F_B^{-1}$ auf den Zustand $K_C$. Die Verwendung von umkehrbaren Funktionen lässt es zu, dass ein erster Zustand und ein zweiter Zustand denselben Geheimwert nutzen, wobei der erste Zustand dem Eintritt in eine untergeordnete (tiefere Ebene) Box von einer übergeordneten (höhere Ebene) Box vorangeht und der zweite Zustand durch erneutes Eintreten in die übergeordnete Box von der untergeordneten Box kreiert wird. Ferner erlaubt die Vielzahl von Funktionen (z.B. $F_A$ und $F_B$ in dem Ausführungsbeispiel) die Schaffung mehrerer untergeordneter Boxen für jede übergeordnete Box und somit eine große Zahl von zulässigen Zuständen, bevor die Sequenz zu Ende ist (z.B. im Endezustand **190**). Beim Übergehen von einem besonderen Zustand in einen anderen besonderen Zustand hängt die Wahl der Funktionen (z.B. im Ausführungsbeispiel von Fig. 1, ob $F_A$, $F_B$, $F_A^{-1}$ oder $F_B^{-1}$ verwendet werden soll) von der/dem aktuellen Richtung und Ort der beiden jeweiligen Zustände ab. Insbesondere wird, wieder mit Bezug auf die in Fig. 1 gezeigte beispielhafte Anordnung, bei der Abwärtsbewegung von einer übergeordneten Box zu einer linken untergeordneten Box, wie z.B. zwischen den Schritten **112** und **113**, $F_A$ durch Berechnen von $K_C \leftarrow F_A(K_C)$ angewendet. Ferner wird bei einer Abwärtsbewegung von einer übergeordneten Box zur rechten untergeordneten Box, z.B. zwischen den Schritten **115** und **116**, $F_B$ angewendet. Ferner wird beim Bewegen von einer linken untergeordneten Box zu ihrer übergeordneten Box, z.B. zwischen den Schritten **114** und **115**, $F_A^{-1}$ durch Berechnen von $K_C \leftarrow F_A^{-1}(K_C)$ angewendet. Schließlich wird bei der Bewegung von einer rechten untergeordneten Box zu ihrer übergeordneten Box, wie z.B. zwischen den Schritten **116** und **117**, $F_B^{-1}$ angewendet. Allgemeiner, die Wahl, welche Funktion in einem bestimmten Zustandsübergang angewendet werden soll, kann einfach in Abhängigkeit von C bestimmt werden, so dass der Client keine Informationen über seinen aktuellen Zustand und seinen aktuellen Zählerwert hinaus zu führen braucht. Dies wird ausführlicher im Abschnitt „Client-seitiges indexiertes Key-Update" unten in Zusammenhang mit dem Ausführungsbeispiel von Fig. 1 beschrieben.

**[0031]** Schließlich kann der Client einen Punkt erreichen, an dem die gesamte Tabelle durchlaufen wurde. So wird z.B. das Ende des Prozesses von Fig. 1 in Schritt **170** erreicht, wo C=60 ist. Nach dieser Transaktion (oder an einem früheren Punkt, wenn die Tabellenlänge die Höchstzahl der vom System zugelassenen Transaktionen übersteigt), könnte sich das Client-Device – und würde sich typischerweise auch – selbst sperren, z.B. durch Löschen seiner internen Geheimnisse. In einigen Fällen werden jedoch möglicherweise andere Aktionen bevorzugt (z.B. durch Wiederholen zurück bis Schritt **110**, Eintreten in einen Zustand, in dem ein Rekeying-Vorgang erforderlich ist usw.). In dem illustrierten Ausführungsbeispiel ist die Zahl der Transaktionen, die vor dem Ende des Prozesses ausgeführt werden kann, gleich:

$$2^{D-1} + \sum_{i=0}^{D-2} 3\left(2^i\right) = 2^{D-1} + 3\left(2^{D-1} - 1\right) = 2^{D+1} - 3$$

(In dem Beispiel mit D=5 kann es somit $2^6$-3 = 61 Transaktionen geben.) Durch Wählen eines ausreichend großen Wertes für D kann ein Systemdesigner die maximale Anzahl von Transaktionen so groß machen, dass das „Ende" niemals erreicht wird. Zum Beispiel lässt D = 39 mehr als eine Billion ($10^{12}$) Transaktionen ohne Wiederholung zu.

**[0032]** Client-seitiges indexiertes Key-Update Für das Ausführungsbeispiel von Fig. 1 können die Prozesse Inkrementieren von C und Wählen, welche Funktion angewendet werden soll ($F_A$, $F_B$, $F_A^{-1}$ oder $F_B^{-1}$) wie in Fig. 2 gezeigt ausgeführt werden. In Schritt **210** prüft das Client-Device, ob C gültig ist, z.B. durch Prüfen, dass C nicht negativ ist und dass C kleiner als $2^{D+1}$-3 ist. (Wenn C ungültig ist, dann verläuft die Transaktion erfolglos und es wird eine entsprechende Aktion ausgeführt.) Da der Client C intern führt, kann Schritt **210** wegfallen, wenn der Client sicher ist, dass C gültig ist. In Schritt **220** initialisiert das Gerät vorübergehende Tiefen- und Zählervariablen N und V, wobei die Werte jeweils in D und C gespeichert sind.

**[0033]** In Schritt **230** prüft das Device, ob die Variable V gleich der Größe $2^N$-3 ist. Wenn dies so ist, dann sollte Funktion $F_A^{-1}$ angewendet werden und die Verarbeitung geht weiter zu Schritt **235**, wo das Device C inkrementiert und $K_C$ durch Berechnen von $K_C \leftarrow F_A^{-1}(K_C)$ aktualisiert. Ansonsten prüft das Device in Schritt **240**, ob die Variable V gleich der Größe $2(2^N\text{-}2)$ ist. Wenn ja, dann sollte Funktion $F_B^{-1}$ angewendet werden und die Verarbeitung geht zu Schritt **245**, wo das Device C inkrementiert und $K_C$ durch Berechnen von $K_C \leftarrow F_B^{-1}(K_C)$ aktualisiert. Ansonsten prüft das Device in Schritt **250**, ob die Variable V gleich null ist. Wenn ja, sollte Funktion $F_A$ angewen-

det werden und die Verarbeitung geht weiter zu Schritt **255**, wo das Device C inkrementiert und $K_C$ durch Berechnen von $K_C \leftarrow F_A(K_C)$ aktualisiert. Ansonsten prüft das Device in Schritt **260**, ob die Variable V gleich der Größe $2^N$-2 ist. Wenn ja, dann sollte Funktion $F_B$ angewendet werden und die Verarbeitung geht weiter zu Schritt **265**, wo das Device C inkrementiert und $K_C$ durch Berechnen von $K_C \leftarrow F_B(K_C)$ aktualisiert.

**[0034]** In Schritt **270** prüft das Device, ob der Wert von V $2^N$-2 übersteigt. Wenn nicht, dann geht die Verarbeitung direkt zu Schritt **280**. Wenn V größer ist als $2^N$-2, dann wird der Wert von V um $2^N$-2 verringert und die Verarbeitung geht zu Schritt **280**. In Schritt **280** werden V und N jeweils dekrementiert, dann geht die Verarbeitung zu Schritt **230**.

**[0035]** Nach der Ausführung einer Zustands-Update-Funktion in Schritt **235**, Schritt **245**, Schritt **255** oder Schritt **265** endet der Client-Prozess erfolgreich mit Schritt **290**. Nach dem erfolgreichen Abschluss des Prozesses von Fig. 2 wird der Geheimwert $K_C$ zum Ausführen einer kryptografischen Transaktion (oder zum Ableiten eines Key, der zum Ausführen der Transaktion verwendet wird, z.B. durch Hashen oder Verschlüsseln von $K_C$, Anhängen eines Salt oder Nonce usw.) verwendet.

**[0036]** Man beachte, dass jede Iteration des Prozesses von Fig. 2 dem Abwärtsbewegen um eine Ebene in der Zeichnung von Fig. 1 entspricht, bis die richtige Update-Operation ermittelt ist. So kann die Zahl der Iterationen der Schleife D nicht übersteigen. Mit Ausnahme der Key-Update-Funktionen (in dem Ausführungsbeispiel $F_A$, $F_B$, $F_A^{-1}$ und $F_B^{-1}$) brauchen nicht alle Implementationen des Funktionauswahlvorgangs leckbeständig zu sein; der Funktionsauswahlvorgang von Fig. 2, sein Eingangswert (d.h. C) und die Wahl der Update-Funktionen brauchen nicht geheim zu sein. Schließlich kann, wie zuvor erwähnt und oben im Falle des Ausführungsbeispiels illustriert wurde, die Auswahl der Funktion, die auf einen besonderen Zustandsübergang angewendet werden soll, allein in Abhängigkeit von C charakterisiert werden, so dass der Client keine Informationen jenseits seines aktuellen Zustands und Zählerwerts zu führen braucht.

Serverseitige indexierte Key-Ableitung

**[0037]** Fig. 3 zeigt einen beispielhaften serverseitigen Prozess, der mit dem beispielhaften Client-seitigen Prozess von Fig. 2 kompatibel ist. Die Fachperson wird verstehen, dass die Schritte von Fig. 3 in der gezeigten beispielhaften Reihenfolge oder in einer anderen Reihenfolge ausgeführt werden können. Vor Beginn des Vorgangs von Fig. 3 holt der Server den Zählerwert C des Client (typischerweise durch Empfangen von C vom Client-Device über eine digitale E/A-Schnittstelle) ein, der als Key-Index verwendet wird. (In diesem Ausführungsbeispiel wird ein Transaktionszähler als Key-Index verwendet, aber alternative Ausgestaltungen können auch einen anderen Wert oder eine Repräsentation des Key-Indexes verwenden.)

**[0038]** Der Server holt auch den Basis-Key-Wert $K_0$ des Client ein (z.B. durch Abrufen von $K_0$ aus dem Speicher des Servers, durch kryptografisches Ableiten von $K_0$ mit anderen Secret-Keys oder Geheimalgorithmen, durch Beschaffen von $K_0$ von Dritten wie z.B. einem Key-Server usw.). Der Server kennt oder beschafft auch D. In Schritt **310** validiert der Server C, um eventuelle mögliche ungültige Werte von C zurückzuweisen. In Schritt **330** werden die temporären Variablen N, V und K jeweils mit den Werten von D, C und $K_0$ initialisiert. In Schritt **330** prüft der Server, ob der Wert von V gleich null ist. Wenn ja, dann ist der Wert von K gleich dem aktuellen Geheimnis ($K_C$) des Client und der Vorgang endet mit Schritt **390**. Ansonsten geht die Verarbeitung zu Schritt **340** weiter, wo der Server testet, ob V gleich dem Wert 2N-2 ist. Wenn ja, dann ist der Wert von K gleich dem aktuellen Geheimnis ($K_C$) des Client und der Vorgang endet mit Schritt **390**. Ansonsten geht die Verarbeitung mit Schritt **350** weiter, wo der Server testet, ob V gleich dem Wert $2(2^N\text{-}2)$ ist. Wenn ja, dann ist der Wert von K gleich dem aktuellen Geheimnis ($K_C$) des Client und der Vorgang endet mit Schritt **390**. Ansonsten prüft der Server in Schritt **360**, ob V größer ist als $2^N$-2. Wenn nicht, dann geht die Verarbeitung in Schritt **370** weiter, wo V dekrementiert, K durch Anwenden von $F_A$ aktualisiert (d.h. $K \leftarrow F_A(K)$) und N dekrementiert werden. Wenn der Test in Schritt **360** ergibt, dass V größer ist als $2^N$-2, dann geht die Verarbeitung weiter zu Schritt **380**, wo der Wert $2^N$-1 von V subtrahiert, K durch Anwenden von $F_B$ aktualisiert (d.h. $K \leftarrow F_B(K)$) und N dekrementiert werden. Nach Schritt **370** oder Schritt **380** geht die Verarbeitung weiter mit Schritt **330**. Die Verarbeitung wird so lange fortgesetzt, bis Schritt **330**, Schritt **340** oder Schritt **350** Abschluss anzeigt. Wenn der Vorgang von Fig. 3 in Schritt **390** zu Ende geht, dann ist der in der Variablen K enthaltene Wert gleich dem Wert von $K_C$ am Client für den Zählerwert C. Client und Server können somit $K=K_C$ verwenden, um eine kryptografische Transaktion zu sichern. Wenn ein Fehler oder eine fehlerverursachende Attacke auftritt, dann unterscheiden sich K und $K_C$ und die kryptografische Transaktion sollte erfolglos verlaufen.

Statustransformationsoperationen

**[0039]** Die obige Diskussion beinhaltete die beispielhaften kryptografischen Operationen $F_A$ und $F_B$ und deren Umkehrungen $F_A^{-1}$ und $F_B^{-1}$, die nun ausführlicher beschrieben werden. Es kann eine Reihe verschiedener solcher Funktionen verwendet werden und die geeignetste Form für diese Funktionen hängt

von den Anforderungen und Charakteristiken des Systems ab.

**[0040]** In den in Fig. 4 gezeigten beispielhaften Funktionen haben der Eingang und der Ausgang jeder Funktion eine Größe von 128 Bit. Für die Funktion $F_A$ wird der Eingangszustand **400** in eine linke Hälfte **405** und eine rechte Hälfte **410** unterteilt, die jeweils 64 Bits haben. Die rechte Hälfte wird als der Eingang zu einer DES-Operation **415** bereitgestellt, die ihren Eingang (rechte Häfte **410**) mit einem Fixed-Key $K_{A1}$ verschlüsselt. Die DES-Operation wird nur als nichtlineare Transformation benutzt, die die Nützlichkeit der partiellen Informationen verringert oder eliminiert, die ein Angreifer über den Eingang haben könnte. Demzufolge braucht der Key $K_{A1}$ nicht geheim zu sein und kann eine veröffentlichte Konstante sein. In Operation **420** wird das Ergebnis der DES-Verschlüsselung einem XOR-Vorgang auf die linke Hälfte des Eingangs unterzogen. Das Ergebnis des XOR wird sowohl zum Ergebnis linke Hälfte **435** als auch zum Eingang in eine zweite DES-Operation **425**. Die zweite DES-Operation verwendet Key $K_{A2}$ zum Erzeugen eines Ergebnisses, das in Operation **430** einem XOR-Vorgang mit dem Eingang rechte Hälfte **410** unterzogen wird. Das XOR-Ergebnis wird zum Ergebnis rechte Hälfte **440**. Das Ergebnis linke Hälfte **435** und das Ergebnis rechte Hälfte **440** werden zum Erzeugen des Endergebnisses **445** kombiniert.

**[0041]** Die Struktur der Funktion $F_B$ kann im Wesentlichen identisch sein, mit der Ausnahme, dass unterschiedliche Keys verwendet werden. Insbesondere verschlüsselt die erste DES-Operation **455** die rechte Hälfte von Eingang **450** mit Key $K_{B1}$ und die DES-Operation **460** verschlüsselt das XOR der linken Hälfte und das erste DES-Ergebnis mit Key $K_{B2}$. Wie bei $F_A$, werden das Ergebnis linke Hälfte **465** und rechte Hälfte **468** zum Erzeugen des Endergebnisses **470** kombiniert.

**[0042]** Die Funktion $F_A^{-1}$ (die Umkehr von $F_A$) wird mit ähnlichen Funktionen berechnet wie $F_A$, aber in der entgegengesetzten Reihenfolge. Der Eingang **475** wird in eine linke Hälfte **476** und eine rechte Hälfte **477** unterteilt. In der DES-Operation **478** wird die linke Hälfte **476** mit dem DES-Key $K_{A2}$ verschlüsselt und das Ergebnis wird mit der rechten Hälfte **477** einem XOR-Vorgang unterzogen. Das XOR-Ergebnis wird zum Ergebnis rechte Hälfte **481** und wird als Eingang in die DES-Operation **479** verwendet, die mit dem Key $K_{A1}$ verschlüsselt. Das Ergebnis der zweiten DES-Operation **479** wird mit dem Eingang linke Hälfte **476** zum Erzeugen des Ergebnisses linke Hälfte **480** einem XOR-Vorgang unterzogen. Schließlich werden das Ergebnis linke Hälfte **480** und rechte Hälfte **481** zum Erzeugen des Endergebnisses **482** kombiniert. Die Funktion $F_B^{-1}$ ist $F_A^{-1}$ ähnlich, mit der Ausnahme, dass der Eingang **485** mit den Keys $K_{B2}$ und $K_{B1}$ anstatt $K_{A2}$ und $K_{A1}$ in den Ausgang **490** transformiert wird.

**[0043]** Hauptziel der Funktionen $F_A$, $F_B$, $F_A^{-1}$ und $F_B^{-1}$ ist es, die Nützlichkeit partieller Informationen über den Eingang zu zerstören, die von einem Angreifer möglicherweise erhalten wurden. So machen beispielsweise die in der in Fig. 4 gezeigten beispielhaften Funktion $F_A$ verwendeten DES-Operationen die Funktion extrem nichtlinear. Ein Angreifer mit statistischen Informationen über den Wert jedes der 128 Eingangsbits (wie z.B. das Raten des Wertes des Bits, was mit einer Wahrscheinlichkeit von geringfügig mehr als 0,5 korrekt ist) ergibt statistische Informationen über den Eingang zur ersten DES-Operation **415**. Der DES-Ausgang wird jedoch effektiv randomisiert – obwohl Angreifer den DES-Key $K_{A1}$ eventuell kennen. Die beiden DES-Operationen in jedem Update-Prozess „mischen" den gesamten Eingangszustand.

**[0044]** So ergeben partielle statistische Informationen über individuelle DES-Eingangsbits keine nützlichen statistischen Informationen über die DES-Ausgangsbits, vorausgesetzt, dass Angreifer niemals genügend Informationen erhalten, um den gesamten Transformationsoperationseingang erraten zu können.

## Andere Ausgestaltungen

**[0045]** Fig. 4 zeigt nur einen beispielhaften Satz von Funktionen für $F_A$ und $F_B$; es können viele weitere Varianten oder alternative Designs verwendet werden. So können beispielsweise mittels zusätzlicher Runden erzeugte Funktionen verwendet werden (z.B. ein Luby-Rackoff-Block-Chiffre mit 3 Runden). Allgemeiner, Ver- und Entschlüsselung mit einem beliebigen Block-Chiffre können für die Funktionen und ihre Umkehrungen verwendet werden. Die zum Konstruieren der Update-Funktion benötigten Grundfunktionen brauchen lediglich zu verhindern, dass über den Eingang ausgeleckte partielle Informationen nützliche Informationen über den Ausgang liefern, so dass die Funktionen nicht unbedingt kryptografisch schwer zu invertieren zu sein brauchen. So können z.B. DES-Varianten mit weniger Runden verwendet werden. Dazu kommt, $F_A$ und $F_B$ in Fig. 4 haben eine ähnliche Struktur, aber dies ist nicht notwendig. $F_A$ und $F_B$ können auch je nach der Zustandsposition gewählt oder modifiziert werden (z.B. mittels unterschiedlicher Funktionen oder modifizierter Funktionen für jede der D-Ebenen).

**[0046]** Es können auch andere Funktionstypen für $F_A$ und $F_B$ verwendet werden. Zum Beispiel, wenn der Eingangszustand ein ungerader Wert zwischen 0 und $2^B$ ist, dann könnten $F_A$ und $F_B$ mit Multiplikation modulo $2^B$ mit ungeraden Konstanten implementiert werden und die Umkehrfunktionen könnten mit Multipli-

kation mit den Umkehrungen der Konstanten ebenfalls mod $2^B$ implementiert werden. (Natürlich können auch andere Operationen wie Multiplikation mit Primmodulen verwendet werden.) Das oben Gesagte ist lediglich beispielhaft; die durchschnittliche Fachperson wird erkennen, dass eine breite Vielfalt anderer Funktionen existiert, die zum Implementieren der Funktionen $F_A$, $F_B$, $F_A^{-1}$ und $F_B^{-1}$ verwendet werden können.

**[0047]** Für zusätzliche Leckbeständigkeit können auch größere Zustände verwendet werden, z.B. kann ein 256-Bit-Zustand durch Verwenden von vier 64-Bit-Blöcken und Verwenden von vier (oder mehr) DES-Operationen zum Aktualisieren des Zustands oder durch Verwenden von zwei (oder mehr) Applikationen einer 128-Bit-Hash-Funktion implementiert werden.

**[0048]** In alternativen Ausgestaltungen der Erfindung können auch andere Key-Update-Prozesse zum Einsatz kommen. So kann beispielsweise durch Verwenden von mehr als zwei Update-Funktionen (und ihrer Umkehrungen) jeder übergeordnete Zustand mehr als zwei untergeordnete Zustände haben. In der Tat können übergeordnete Zustände eine beliebige Anzahl von untergeordneten Zuständen haben, obwohl mit zunehmender Zahl der untergeordneten Zustände die Zahl der kryptografischen Operationen, die den Wert des übergeordneten Zustands beinhalten, und die Zahl der Zustände, die denselben Secret-Key gemeinsam nutzen, ebenfalls zunehmen; dadurch nimmt die Gelegenheit für eine Attacke auf das System durch einen Angreifer potentiell zu.

**[0049]** Der Typ des Zustandsaktualisierungsprozesses, der illustrativ mit Bezug auf **Fig. 1** beschrieben wurde, ist deshalb vorteilhaft, weil er sehr wenig Speicher und sehr wenig Verarbeitungs-Overhead benutzt, während die maximale Anzahl an Transaktionen unter Verwendung desselben Geheimwertes gering ist. (Je öfter solche Geheimwerte verwendet werden, desto wahrscheinlicher wird eine erfolgreiche externe Mithörattacke.) Daher werden in einer alternativen Ausgestaltung Transformationen unter Verwendung nur der Zustände auf der untersten Ebene des Diagramms (die nur einmal erzeugt werden) ausgeführt, so dass Geheimwerte nicht wiederverwendet werden. Dies verringert die Möglichkeit, dass Informationen auslecken, erhöht aber das Verarbeitungs-Overhead pro Transaktion auf durchschnittlich etwa vier Updates. (Auch ist die Menge an Zeit pro Transaktion nicht genau, da die Anzahl der Update-Prozesse von 2 bis 2D-2 reicht. Dies ist jedoch häufig kein Problem, da nur wenige Anwendungen jemals Werte von D benötigen, die größer als etwa 40 sind, und viele Devices Tausende von kryptografischen Operationen pro Sekunde ausführen können.)

**[0050]** In noch einer anderen alternativen Ausgestaltung kann der Client einen Wert in jeder vertikalen Ebene oder Reihe cachen. Wenn höhere Werte gecacht werden, brauchen keine Umkehroperationen ausgeführt zu werden, aber es wird etwas mehr Speicherplatz benötigt. Bei einer solchen Ausgestaltung werden durchschnittlich zwei Anwendungen von $F_A$ oder $F_B$ (die in einer solchen Ausgestaltung keine leichten Umkehrfunktionen zu haben brauchen) pro Operation benötigt, wenn nur Zustände auf unterster Ebene (Einzelbenutzung) für Transaktionen verwendet werden. Ein Diagramm der Zustands-Update-Prozesse für eine solche Implementation würde einer Hash-Baumstruktur ähneln. Für Implementationen, die Konstantzeit- oder vorhersehbarere Leistung benötigen, kann die bei Operationen, die nur eine einzige Anwendung von $F_A$ oder $F_B$ brauchen, verfügbare zusätzliche Verarbeitungszeit zum Vorberechnen von Werten benutzt werden, die in der Zukunft benötigt werden, und dadurch die Ausführungszeit auf zwei $F_A$ oder $F_R$ Operationen pro Transaktion begrenzen.

**[0051]** In anderen Ausgestaltungen kann der vom Server verwendete Key-Index ein anderer Wert als ein Transaktionszähler sein, da der Server lediglich genügend Informationen benötigt, um den aktuellen Transaktions-Key vom Root-Key abzuleiten.

**[0052]** In einigen Anwendungen kann C periodisch (z.B. wenn C von einem Timer angesteuert wird) oder von einem anderen Event als ausgeführten Transaktionen inkrementiert werden. In solchen Ausgestaltungen verläuft, wenn der Client (oder Server) C nicht korrekt aktualisiert und den entsprechenden aktualisierten Key ableitet, die Transaktion erfolglos. Wenn der erste vom Client (oder Server) versuchte Wert von C erfolglos ist, dann können andere wahrscheinliche Session-Key-Werte (wie z.B. die mit nahen Werten von C) versucht werden. (Wenn natürlich die Client- und Server-Versionen von C zu weit divergieren, dann läuft die Transaktion nicht ab.) Während der Key-Index (z.B. C) normalerweise explizit ausgetauscht wird, könnte in Fällen wie diesem der Server möglicherweise C indirekt erraten oder beschaffen können.

**[0053]** Wenn sowohl der Client als auch der Server vor externen Mithörattacken geschützt werden müssen, dann kann die Transaktion mit dem größeren der Transaktionszähler C der beiden Parteien ausgeführt werden. Insbesondere können Client und Server Zählerwerte austauschen und (wenn die Zähler nicht gleich sind) jedes Device kann seinen Zählerwert auf gleich dem größeren aus seinem Wert und dem empfangenen Wert einstellen. Das Device mit dem niedrigeren Wert aktualisiert sein Geheimnis, um den entsprechenden Transaktions-Key abzuleiten. Dieses Update kann durch Anwenden einer Kombination aus den gewöhnlichen Update-Funktionen und deren Umkehrungen implementiert werden. (Zum Beispiel,

mit Bezug auf die in Fig. 2 exemplifizierte Technik, ein Client in Zustand **117** könnte zu Zustand **136** springen, indem er $F_A^{-1}$ zweimal, dann $F_B$ dreimal anwendet. Im Allgemeinen sollte die benötigte Gesamtzahl der Update-Funktionen kleiner als 2D-1 sein. Diese „Schnellvorlauf"-Fähigkeit bewahrt die Eigenschaft, dass kein Zustand mehr als eine finite Anzahl von – hier drei – Malen benutzt oder abgeleitet wird.) In Devices, die diese Fähigkeit implementieren, ist sorgfältig darauf zu achten, dass gewährleistet wird, dass das System nicht erfolglos verläuft, wenn ein großer, inkorrekter Wert von C auftritt. (Zum Beispiel können Devices zu große Sprünge in C zurückweisen oder können zusätzliche kryptografische Authentifizierung benötigen, z.B. die höchstwertigen Bits von C.) Ein solches Protokoll kann zum Vereinbaren eines Transaktionszählers für Ausgestaltungen verwendet werden, die mehr als zwei Parteien in kryptografischen Transaktionen involvieren.

**[0054]** Schließlich kann der für den Transaktions-Key verwendete Ist-Wert der Wert sein, der von der Transformationsfunktion erzeugt wird, und es kann ein Wert verwendet werden, der vom Transformationsergebnis abgeleitet wird. So kann beispielsweise das Transformationsergebnis zum Erzeugen des Session-Key verschlüsselt oder gehasht werden. Ein Hash-Schritt kann dabei helfen, die Zahl der von irgendeinem Key ausgeführten Operationen zu begrenzen und somit dabei beizutragen, die Menge an Informationen über den Key zu begrenzen, die an Angreifer auslecken kann. Alternativ oder zusätzlich können zusätzliche Hash-Operationen periodisch während der Verwendung des Session-Key ausgeführt oder frische Session-Keys periodisch benötigt werden.

**[0055]** Um die größtmögliche Anzahl von Transaktionen mit einem bestimmten Secret-Key zu beobachten, könnte ein Angreifer versuchen, ein Zieldevice zurückzusetzen, bevor der Speicher des Device mit dem neuen Wert von $K_C$ aktualisiert werden kann (z.B. während oder unmittelbar nach der Berechnung von $F_A$ oder $F_B$). Ein solches Reset bedeutet jedoch nicht unbedingt, dass eine Attacke abläuft, da Resets während des normalen Betriebs vieler Systeme auftreten können. (Zum Beispiel kann die Stromzufuhr unterbrochen werden, wenn eine Chipkarte bei einer Transaktion entfernt wird.) Daher wird in einer bevorzugten Ausgestaltung ein im nichtflüchtigen Speicher gespeicherter Störungszähler vor jedem Update-Vorgang aktualisiert. Vor Beginn des Updates wird der Zähler getestet, um zu ermitteln, ob die Zahl sequentieller Fehler einen Maximalwert übersteigt, und wenn nicht, dann läuft die Transaktion normal ab. Wenn der neue Wert von $K_C$ berechnet und sicher auf den Speicher geschrieben und C inkrementiert wurde, dann wird der Störungszähler zurückgesetzt. Die Wahrscheinlichkeit, dass der Zählerschwellenwert während des normalen Betriebs des Device überschritten wird (d.h. wenn keine Attacke abläuft), ist gering, besonders dann, wenn der Update-Vorgang schnell ist.

**[0056]** Der mit Bezug auf die Fig. 1, Fig. 2 und Fig. 3 beschriebene beispielhafte Key-Update-Prozess gewährleistet, dass kein Secret-Key-Wert jemals in mehr als nur einer relativ geringen Zahl (hier drei) von Transaktionen benutzt wird. Angreifer haben so die Gelegenheit, Informationen über den Geheimzustand während den drei Transaktionen selbst, den drei Key-Update-Prozessen, die die Transaktions-Keys erzeugen, und den drei Update-Prozessen, die die Transaktions-Keys nach den Transaktionen transformieren, zu sammeln. Implementierer müssen sicherstellen, dass die Gesamtzahl der Informationen über die Geheimnisse, die bei diesen Vorgängen an Angreifer auslecken, nicht ausreicht, um den Geheimzustand zu kompromittieren. Beim Charakterisieren eines Designs ist es häufig nützlich, die maximale Menge an Informationen, die von jeder Transaktion auslecken kann, ohne die Sicherheit zu kompromittieren, zu ermitteln oder zu schätzen.

## Weitere Überlegungen

**[0057]** Kryptografische Operationen sollten normalerweise geprüft werden, um sicherzustellen, dass falsche Berechnungen Keys nicht kompromittieren oder andere Attacken ermöglichen. Kryptografische Implementationen der vorliegenden Erfindung können, und werden in einer bevorzugten Ausgestaltung der Erfindung auch, mit Fehlererkennungs- und/oder Fehlerkorrekturlogik kombiniert, um zu gewährleisten, dass kryptografische Operationen korrekt ausgeführt werden. So besteht eine einfache und effektive Technik beispielsweise darin, kryptografische Operationen zweimal auszuführen, idealerweise mit zwei unabhängigen Hardware-Prozessoren und Implementationen, mit einem Komparator, um zu prüfen, ob beide identische Ergebnisse produzieren. Wenn die von den beiden Einheiten produzierten Ergebnisse nicht übereinstimmen, dann verhindert der Komparator eine Benutzung von beiden. In Situationen, bei denen Sicherheit wichtiger ist als Zuverlässigkeit, kann der Komparator das Gerät veranlassen, sich selbst zu zerstören, wenn ernsthafte Fehler auftreten. So könnte der Komparator beispielsweise eine Selbstzerstörung veranlassen, wenn zwei defekte DES-Operationen sequentiell auftreten oder wenn fünf defekte DES-Operationen während der Lebensdauer des Device auftreten. In einigen Kryptosystemen ist keine Redundanz notwendig. So können beispielsweise mit RSA Selbstprüffunktionen in die Kryptosystemimplementation selbst eingebaut werden oder eine Verifikation kann nach den Operationen ausgeführt werden.

**[0058]** Selbstdiagnostikfunktionen wie POST (Power On Self Test) sollten auch eingebaut werden, um

sicherzustellen, dass keine kryptografischen Funktionen beschädigt sind. In einigen Chipkarten und anderen Devices muss ATR (Answer To Reset) vorgesehen werden, bevor ein umfassender Selbsttest vollzogen werden kann. In solchen Fällen kann der Selbsttest bis zur ersten Transaktion oder bis zu einer ausreichenden Ruheperiode aufgeschoben werden. So kann beispielsweise nach der Initialisierung ein Flag gesetzt werden, der einen erfolgreichen POST-Abschluss anzeigt. Während die Karte auf einen Befehl vom Hostsystem wartet, kann sie den POST versuchen. Sämtliche während des POST empfangenen E/A verursachen einen Interrupt, der den POST storniert (und den POST-beendet-Flag auf null lassen). Wenn eine kryptografische Funktion aufgerufen wird, dann prüft das Device den POST-Flag und führt (wenn er nicht gesetzt ist) zuerst den POST aus.

Schlussfolgerungen

**[0059]** Die Erfindung umfasst daher eine Familie von verwandten Techniken, die die Konstruktion von Devices ermöglichen, die erheblich beständiger gegen Attacken sind als Devices zu ähnlichen Kosten und mit ähnlicher Komplexität, die die Erfindung nicht anwenden. Zudem könnten mehrere Sicherheitstechniken erforderlich sein, um ein System sicher zu machen; und Leckbeständigkeit könnte in Zusammenhang mit anderen Sicherheitsmethoden oder Gegenmaßnahmen eingesetzt werden.

**[0060]** Wie die Fachperson verstehen wird, sind die oben beschriebenen Techniken nicht auf besondere Wirtsumgebungen oder Formfaktoren begrenzt. Stattdessen können sie in einer breiten Vielfalt von Anwendungen eingesetzt werden, einschließlich und ohne Begrenzung: kryptografische Chipkarten jeder Art einschließlich, ohne Begrenzung, Chipkarten, im Wesentlichen gemäß ISO 7816-1, ISO 7816-2 und ISO 7816-3 („Chipkarten gemäß ISO 7816"); kontaklose und auf Annäherung basierende Chipkarten und kryptografische Tokens; Speicherwertkarten und -systeme; kryptografisch gesicherte Kredit- und Debitkarten; Kundentreuekarten und -systeme; kryptografisch authentifizierte Kreditkarten; kryptografische Beschleuniger; Spiel- und Wettsysteme; sichere kryptografische Chips; eingriffsresistente Mikroprozessoren; Software-Programme (einschließlich, ohne Begrenzung, Programme für die Verwendung auf Personal Computern, Servern usw. sowie Programme, die auf kryptografische Devices geladen oder darin eingebettet sein können); Key-Management-Devices; Banking-Key-Management-Systeme; sichere Webserver; elektronische Zahlungssysteme; Mikrozahlungssysteme und -zähler; vorausbezahlte Telefonkarten; kryptografische Identifikationskarten und sonstige Identitätsprüfsysteme; Systeme für elektronische Geldübertragungen; Bankautomaten; Verkaufsstellenterminals; Zertifikatausstellungssysteme; elektronische Marken; Zutrittssysteme; physikalische Schlösser jeder Art, die mit kryptografischen Keys arbeiten; Systeme zum Entschlüsseln von Fernsehsignalen (einschließlich, ohne Begrenzung, Rundfunkfernsehen, Satellitenfernsehen und Kabelfernsehen); Systeme zum Entschlüsseln von verschlüsselter Musik und sonstigem Toninhalt (einschließlich über Computernetze verteilter Musik); Systeme zum Schützen von Videosignalen jeder Art; Systeme zum Schützen von geistigem Eigentum und von Urheberrechten (wie z.B. die, die zum Verhindern des unbefugten Kopierens oder Gebrauchs von Kinofilmen, Toninhalt, Computerprogrammen, Videospielen, Bildern, Text, Datenbanken usw. verwendet werden); Verwürfelung von Zellulartelefon und Authentifizierungssysteme (inkl.

**[0061]** Telefonauthentifizierungs-Chipkarten); sichere Telefone (einschließlich Key-Speichergeräte für solche Telefone); kryptografische PCMCIA-Karten; tragbare kryptografische Tokens und kryptografische Datenprüfsysteme.

**[0062]** Das oben Gesagte illustriert beispielhafte Ausgestaltungen und Anwendungen der Erfindung, anhand derer verwandte Variationen, Erweiterungen und Modifikationen offensichtlich werden, ohne vom Umfang der Erfindung abzuweichen.

**Patentansprüche**

1. Rechnerimplementiertes Verfahren zum Sichern eines ersten Gerätes während der Durchführung von Transaktionen mit wenigstens einem zweiten Gerät, wobei das erste Gerät einen rechnerlesbaren Speicher mit einem internen Geheimzustand beinhaltet und wobei das wenigstens eine zweite Gerät Zugang zu einem Grundgeheimzustand hat, der dem anfänglichen internen Geheimzustand des ersten Gerätes entspricht, umfassend die folgenden, von dem ersten Gerät ausgeführten Schritte:
(a) (i) Lesen (**220**) eines mit dem internen Geheimzustand assoziierten Indexparameters aus dem Speicher, der eine Mehrzahl von Werten gemäß einer definierten Sequenz annimmt, wobei jeder Wert auf einem von mehreren Pegeln ist;
(ii) Benutzen des Indexparameters zum Auswählen (**230**, **240**, **250**, **260**) von wenigstens einer aus mehreren Transformationsoperationen (**235**, **245**, **255**, **265**), die einen Eingangsgeheimzustand in einen Ausgangsgeheimzustand umwandeln, wobei der Eingangsgeheimzustand und der Ausgangsgeheimzustand auf einem höheren oder tieferen Pegel relativ zueinander innerhalb der mehreren Pegel von internen Geheimzuständen sind;
(b) Anwenden wenigstens der ausgewählten Transformationsoperation (**235**, **245**, **255**, **265**) auf den internen Geheimzustand als Eingangsgeheimzustand, um einen aktualisierten Geheimzustand als den Ausgangsgeheimzustand zu erzeugen, wobei jede aus-

gewählte Transformationsoperation eine nichtlineare kryptografische Funktion umfasst, die Teile des Eingangsgeheimzustands mischt, um einen Ausgangsgeheimzustand zu erzeugen, auf eine solche Weise, dass ein Leck von partieller statistischer Information über den Eingangsgeheimzustand keine nützliche Information über den Ausgangsgeheimzustand ergibt, selbst dann, wenn die kryptografische Funktion bekannt ist;
wobei eine Wiederholung der obigen Schritte (a) und (b) gemäß Schritt (e) die Sequenz von Werten von internen Geheimzuständen definiert, die die Mehrzahl von Pegeln von internen Geheimzuständen vom Grundgeheimzustand zu dem aktualisierten Geheimzustand durchlaufen, wobei wenigstens einige der internen Geheimzustände in der Sequenz dieselben Werte haben; und
für wenigstens einige Wiederholungen der obigen Schritte (a) und (b) gemäß Schritt (e) Ausführen der folgenden Schritte (c) und (d):
(c) Ersetzen (**235**, **245**, **255**, **265**) in dem Speicher:
(i) den internen Geheimzustand durch den aktualisierten Geheimzustand, wobei der aktualisierte Geheimzustand bei einem nachfolgenden Auftreten von Schritt (b) als der Eingangsgeheimzustand verwendet wird, wenn Schritt (b) gemäß Schritt (e) wiederholt wird; und
(ii) den Indexparameter durch einen aktualisierten Indexparameter;
(d) Ausführen einer kryptografischen Transaktion mit dem wenigstens einen zweiten Gerät durch Übertragen des aktualisierten Indexparameters und von mit dem aktualisierten Geheimzustand gesicherten Daten zu dem wenigstens einen zweiten Gerät, wobei das zweite Gerät für die folgenden Aufgaben konfiguriert ist:
(i) Regenerieren des aktualisierten Geheimzustands von dem Grundgeheimzustand, zu dem das zweite Gerät Zugang hat, mittels des aktualisierten Indexparameters, durch Ausführen nur von Vorwärtstransformationsoperationen aus den mehreren Transformationsoperationen bei (ii) von einem höheren Pegel zu einem tieferen Pegel innerhalb der mehreren Pegel, wobei die Anzahl der Ausführungen erheblich geringer ist als die Anzahl der Wiederholungen der Schritte (a) bis (d) am ersten Gerät gemäß Schritt (e); und
(ii) Verwenden des abgeleiteten aktualisierten Geheimzustands zum Verarbeiten der gesicherten Daten; und
(e) Wiederholen der Schritte (a) bis (d) mehrere Male über eine Mehrzahl von durch das erste Gerät ausgeführten Transaktionen.

2. Verfahren nach Anspruch 1, wobei die Transaktion mit einem kryptografischen Schlüssel ausgeführt wird, der der Wert des Geheimzustands ist.

3. Verfahren nach Anspruch 1, wobei die Transaktion mit einem kryptografischen Schlüssel ausgeführt wird, der vom Wert des Geheimzustands abgeleitet wird.

4. Verfahren nach Anspruch 1, 2 oder 3, wobei die Transformationsoperationen (**235**, **245**, **255**, **265**) so gewählt werden (**230**, **240**, **250**, **260**), dass Werte für den aktualisierten Geheimzustand niemals mehr als eine festgelegte Anzahl von Malen neu geschaffen werden, wenn Schritt (b) eine große Anzahl von Malen wiederholt wird.

5. Verfahren nach Anspruch 4, wobei die feste Zahl drei ist.

6. Verfahren nach einem der Ansprüche 1 bis 5, das ferner einen Schritt des Verifizierens beinhaltet, dass die gewählte Transformation korrekt berechnet wurde.

7. Verfahren nach einem der Ansprüche 1 bis 6, das ferner die Schritte des Inkrementierens eines Fehlerzählers vor Schritt (b), das Anhalten, wenn der Fehlerzähler einen Höchstwert überschreitet und das Zurückstellen des Fehlerzählers nach dem Vollzug von dem Schritt (b) beinhaltet.

8. Verfahren nach einem der Ansprüche 1 bis 7, das in einer Chipkarte gemäß ISO 7816 implementiert ist.

9. Verfahren nach Anspruch 8, wobei die Chipkarte eine Karte mit gespeichertem Wert ist.

10. Verfahren nach einem der Ansprüche 1 bis 9, wobei die Transaktionen eine sichere Bezahlung für einen Kauf beinhalten.

11. Verfahren nach einem der Ansprüche 1 bis 9, wobei die Transaktionen das Autorisieren des Zugriffs auf einen Dienst beinhalten.

12. Verfahren nach Anspruch 11, wobei der Dienst den Zugriff auf einen Webserver beinhaltet.

13. Erstes kryptografisches Gerät zum Ausführen von Transaktionen mit wenigstens einem zweiten kryptografischen Gerät, das Folgendes umfasst:
(a) wenigstens einen Speicher, der einen internen Geheimzustand enthält, der mehrere Werte gemäß einer definierten Sequenz mit einem Grundgeheimzustand annimmt, wobei jeder Wert auf einem von mehreren Pegeln ist;
(b) einen Prozessor, der so konfiguriert ist, dass er wiederholt eine Mehrzahl von kryptografischen Transaktionen mit dem zweiten kryptografischen Gerät ausführt, wobei jede Transaktion kryptografisch verarbeitete Daten beinhaltet, wobei:
(i) jede der kryptografischen Transaktionen (**235**, **245**, **255**, **265**) mit dem internen Geheimzustand ausgeführt wird,
(ii) zwischen den wiederholten Transaktionen we-

nigstens eine von mehreren Transformationsoperationen (**235**, **245**, **255**, **265**), die einen Eingangsgeheimzustand in einen Ausgangsgeheimzustand umwandeln, ausgewählt und auf den internen Geheimzustand als den Eingangsgeheimzustand angewendet wird, um einen aktualisierten Geheimzustand als den Ausgangsgeheimzustand zu erzeugen, wobei der Eingangsgeheimzustand und der Ausgangsgeheimzustand auf einem höheren oder tieferen Pegel relativ zueinander innerhalb der mehreren Pegel von internen Geheimzuständen sind, wobei die Transformationsoperation eine nichtlineare kryptografische Funktion umfasst, die Teile des Eingangsgeheimzustands mischt, um einen Ausgangsgeheimzustand zu erzeugen, auf eine solche Weise, dass ein Leck von partieller statistischer Information über den Eingangsgeheimzustand keine nützlichen Informationen über den Ausgangsgeheimzustand ergibt, selbst dann, wenn die kryptografische Funktion bekannt ist;
(iii) nach der Transformationsoperation der Wert des aktualisierten Geheimzustands in dem wenigstens einen Speicher zur Nutzung in wenigstens einer Folgetransaktion gespeichert wird; und
(iv) die Schritte (i) bis (iii) mehrere Male über mehrere durch das erste Gerät ausgeführte Transaktionen wiederholt werden, wobei eine Wiederholung der Schritte (i) bis (iii) die Sequenz von Werten von internen Geheimzuständen definiert, die die mehreren Pegel von internen Geheimzuständen vom Grundgeheimzustand zum aktualisierten Geheimzustand durchlaufen, wobei wenigstens einige der internen Geheimzustände in der Sequenz dieselben Werte haben;
(c) eine Schnittstelle, die so konfiguriert ist, dass sie die mit dem aktualisierten Geheimzustand gesicherten Daten zu dem zweiten Gerät überträgt, in dem der Wert des aktualisierten Geheimzustands, nachdem der Prozessor mehrere Transformationsoperationen ausgeführt hat, vom Grundgeheimzustand durch Ausführen einer erheblich kleineren Zahl von Transformationsoperationen als die Zahl der vom ersten Gerät ausgeführten Wiederholungen abgeleitet werden kann.

14. Erstes Gerät nach Anspruch 13, wobei die Transaktion mit einem kryptografischen Schlüssel ausgeführt wird, der der Wert des Geheimzustands ist.

15. Erstes Gerät nach Anspruch 13, wobei die Transaktion mit einem kryptografischen Schlüssel ausgeführt wird, der vom Wert des Geheimzustands abgeleitet wird.

16. Erstes Gerät nach Anspruch 13, 14 oder 15, wobei das erste Gerät eine Chipkarte gemäß ISO 7816 ist.

17. Erstes Gerät nach einem der Ansprüche 13 bis 16, bei dem: (i) die Mehrzahl von Transformationsoperationen n Mal ausgeführt wird; und (ii) die Transformationsoperationen (**235**, **245**, **255**, **265**) so gewählt werden (**230**, **240**, **250**, **26**), dass der Wert des aktualisierten Geheimzustands, nachdem der Prozessor die n Transformationsoperationen ausgeführt hat, durch das zweite Gerät vom Wert des internen Geheimzustands vor den n Transformationsoperationen mit erheblich weniger Rechenaufwand abgeleitet werden kann, als zum Ausführen von n Transformationsoperationen nötig wäre.

18. Erstes Gerät nach Anspruch 17, wobei n größer als 50 ist.

19. Erstes Gerät nach Anspruch 17 oder 18, wobei die Zahl der vom zweiten Gerät ausgeführten Transformationsoperationen durch den Logarithmus der Zahl der vom Prozessor ausgeführten Transformationsoperationen (**235**, **245**, **255**, **265**) ermittelt wird.

20. Erstes Gerät nach Anspruch 17, 18 oder 19, wobei der wenigstens eine Speicher ferner einen Indexparameter enthält, und wobei der Prozessor so konfiguriert ist, dass er den Wert des Indexparameters jedes Mal inkrementiert, wenn der Wert des internen Geheimzustands aktualisiert wird.

21. Erstes Gerät nach Anspruch 20, wobei der Prozessor so konfiguriert ist, dass er den Wert des internen Geheimzustands durch Auswählen (**230**, **240**, **250**, **260**) von wenigstens einer kryptografischen Transformation aus mehreren vordefinierten kryptografischen Transformationen (**235**, **245**, **255**, **265**) und Anwenden der wenigstens einen kryptografischen Transformation auf den internen Geheimzustand aktualisiert.

22. Erstes Gerät nach Anspruch 21, wobei der wenigstens eine Speicher ferner einen Tiefenparameter D enthält und wobei der Prozessor so konfiguriert ist, dass er die wenigstens eine kryptografische Transformation (**235**, **245**, **255**, **265**) auf der Basis des aktuellen Wertes des Indexparameters und des Tiefenparameters D auswählt (**230**, **240**, **250**, **260**).

23. Erstes Gerät nach Anspruch 22, wobei der Geheimzustand D Unterelemente beinhaltet und die wenigstens eine kryptografische Transformation (**235**, **245**, **255**, **265**) wenigstens eines der Unterelemente modifiziert und wobei die Wahl (**230**, **240**, **250**, **260**) des wenigstens einen zu modifizierenden Unterelementes vom Indexparameter abhängt.

24. Erstes Gerät nach Anspruch 23, wobei die mehreren von vordefinierten kryptografischen Transformationen (**235**, **245**, **255**, **265**) wenigstens zwei Transformationen (**235**, **245**) und die Umkehrungen der beiden Transformationen (**255**, **265**) beinhaltet.

25. Erstes Gerät nach Anspruch 24, wobei die mehreren kryptografischen Transformationen (**235**, **245**, **255**, **265**) eine Blockverschlüsselung beinhalten.

26. Erstes Gerät nach Anspruch 23 oder 24, wobei der Speicher so initialisiert (**220**) wird, dass der Anfangswert des Indexparameters null ist, und der Prozessor so konfiguriert ist, dass er eine erste kryptografische Transformation (**235**, **245**, **255**, **265**) auswählt (**230**, **240**, **250**, **260**), wenn der aktuelle Wert des Indexparameters kleiner als D-1 ist.

27. Erstes Gerät nach Anspruch 26, wobei der Prozessor so konfiguriert ist, dass er die Umkehrung der ersten kryptografischen Transformation wählt, wenn der aktuelle Wert des Indexparameters gleich D-1 ist, und wobei der Prozessor so konfiguriert ist, dass er eine zweite kryptografische Transformation wählt, wenn der aktuelle Wert des Indexparameters gleich D ist, und wobei der Prozessor so konfiguriert ist, dass er die Umkehrung der zweiten kryptografischen Transformation wählt, wenn der aktuelle Wert des Indexparameters gleich D+1 ist.

28. Erstes Gerät nach einem der Ansprüche 13 bis 27, wobei der Prozessor Überlauferkennungslogik beinhaltet, die so konfiguriert ist, dass sie prüft, ob der aktuelle Wert des Indexparameters gültig ist.

29. Zweites kryptografisches Gerät zum Ausführen von Transaktionen mit einem ersten kryptografischen Gerät, das Folgende umfassend:
(a) eine Schnittstelle zum Empfangen eines Wertes (**320**) eines Indexparameters und von mit einem aktuellen Wert eines internen Geheimzustands gesicherten Daten, wobei der aktuelle Wert des internen Geheimzustands vom ersten Gerät durch Ausführen einer Sequenz von durch den Indexparameter (**230**, **240**, **250**, **260**) definierten Transformationsoperationen (**235**, **245**, **255**, **265**) erzeugt wurde, beginnend mit einem Basiswert des internen Geheimzustands und endend mit dem aktuellen Wert des internen Geheimzustands, wobei jeder Wert auf einem von mehreren Pegeln ist, wobei jede Transformationsoperation einen Eingangsgeheimzustand in einen Ausgangsgeheimzustand umwandelt, wobei der Eingangsgeheimzustand und der Ausgangsgeheimzustand auf einem höheren oder tieferen Pegel relativ zueinander innerhalb mehreren Pegeln von internen Geheimzuständen sind; und
(b) einen Prozessor, der so konfiguriert ist, dass er den aktuellen Wert des internen Geheimzustands vom Basiswert des internen Geheimzustands erneut ableitet, zu dem das zweite Gerät Zugang hat, wobei:
(i) die Neuableitung die Ausführung einer erheblich kleineren Zahl von Transformationsoperationen (**370**, **380**) relativ zur Zahl der vom ersten Gerät ausgeführten Transformationsoperationen beinhaltet, indem nur Vorwärtstransformationsoperationen aus den Transformationsoperationen bei (a) von einem höheren Pegel zu einem tieferen Pegel innerhalb der mehreren Pegel ausgeführt werden, wobei jede Transformationsoperation eine nichtlineare kryptografische Funktion beinhaltet, die Teile des Eingangsgeheimzustands mischt, um den Ausgangsgeheimzustand zu erzeugen;
(ii) der Indexparameter die kürzere Sequenz von Transformationsoperationen (**370**, **380**) in (i) ermittelt; und
(iii) die Höchstzahl der in der kürzeren Sequenz benötigten Transformationsoperationen (**370**, **380**) erheblich geringer ist als die Anzahl der zulässigen Werte für den Indexparameter.

30. Zweites Gerät nach Anspruch 29, wobei die Höchstzahl der Transformationsoperationen in (b) (iii) durch den Logarithmus der Anzahl der zulässigen Werte für den Indexparameter bestimmt wird.

31. Zweites Gerät nach Anspruch 29 oder 30, wobei die empfangenen Daten mit einem Schlüssel gesichert werden, der vom aktuellen Wert des internen Geheimzustands abgeleitet wurde.

32. Zweites Gerät nach Anspruch 29, 30 oder 31, wobei das erste Gerät eine Chipkarte gemäß ISO 7816 umfasst.

33. Zweites Gerät nach einem der Ansprüche 29 bis 32, wobei das zweite Gerät ein Handelsterminal für ein Zahlungssystem ist.

34. Zweites Gerät nach Anspruch 33, wobei der Prozessor so konfiguriert ist, dass er den aktuellen Wert des internen Geheimzustands ableitet, indem er für jede Wiederholung der Geheimzustandstransformationsschleife einen Wert benutzt, der vom Indexparameter abgeleitet wurde, um wenigstens eine kryptografische Transformation aus einer Mehrzahl von kryptografischen Transformationen (**370**, **380**) auszuwählen (**360**), und die wenigstens eine kryptografische Transformation (**370**, **380**) auf den internen Geheimzustand anwendet.

35. Kryptografisches System, das ein erstes Gerät nach Anspruch 13 und ein zweites Gerät nach Anspruch 29 zum Ausführen von Transaktionen dazwischen umfasst.

36. Verfahren, das von einem kryptografischen Servergerät ausgeführt wird und Folgendes beinhaltet:
(a) Empfangen, von einem Client-Gerät, eines Wertes (**320**) eines Indexparameters und von mit einem aktuellen Wert eines internen Geheimzustands gesicherten Daten, wobei der aktuelle Wert des internen Geheimzustands vom Client-Gerät durch Ausführen einer Sequenz von durch den Indexparameter (**230**, **240**, **250**, **260**) definierten Transformationsoperatio-

nen (**235**, **245**, **255**, **265**) erzeugt wurde, beginnend mit einem Basiswert des internen Geheimzustands und endend mit dem aktuellen Wert des internen Geheimzustands, wobei jeder Wert auf einem von mehreren Pegeln ist, wobei jede Transformationsoperation einen Eingangsgeheimzustand in einen Ausgangsgeheimzustand umwandelt, wobei der Eingangsgeheimzustand und der Ausgangsgeheimzustand auf einem höheren oder tieferen Pegel relativ zueinander innerhalb mehrerer Pegel von internen Geheimzuständen sind; und
(b) im Server-Gerät, erneutes Ableiten des aktuellen Wertes des internen Geheimzustands vom Basiswert des internen Geheimzustands, zu dem das Servergerät Zugang hat, wobei:
(i) die Neuableitung die Ausführung einer erheblich kleineren Zahl von Transformationsoperationen (**370**, **380**) relativ zu der Anzahl der vom Client-Gerät ausgeführten Transformationsoperationen beinhaltet, indem nur Vorwärtstransformationsoperationen aus den Transformationsoperationen bei (a) von einem höheren Pegel zu einem tieferen Pegel innerhalb der mehreren Pegel ausgeführt werden, wobei jede Transformationsoperation eine nichtlineare kryptografische Funktion beinhaltet, die Teile des Eingangsgeheimzustands mischt, um einen Ausgangsgeheimzustand zu erzeugen;
(ii) der Indexparameter die kürzere Sequenz von Transformationsoperationen (**370**, **380**) in (i) bestimmt; und
(iii) die Höchstzahl der in der kürzeren Sequenz benötigten Transformationsoperationen (**370**, **380**) erheblich kleiner ist als die Anzahl der zulässigen Werte für den Indexparameter.

37. Verfahren nach Anspruch 36, wobei das Client-Gerät eine Chipkarte gemäß ISO 7816 ist.

38. Verfahren nach Anspruch 36 oder 37, wobei das Client-Gerät und das Servergerät Komponenten eines größeren Gerätes sind.

39. Verfahren nach Anspruch 36, 37 oder 38, wobei das Servergerät auch einen Indexparameter enthält, und das ferner die folgenden Schritte beinhaltet:
(a) Auswählen des größeren Indexparameters der beiden Geräte,
(b) Verwenden des größeren Indexparameterwertes zum Sichern der Transaktion, und
(c) durch beide Geräte das Inkrementieren und Speichern des größeren Indexparameterwertes für die Verwendung in Folgetransaktionen.

40. Verfahren nach einem der Ansprüche 36 bis 39, wobei die Schritte in der gegebenen Reihenfolge ausgeführt werden.

41. Verfahren nach einem der Ansprüche **36** bis **40**, wobei die Transformationsoperation Folgendes beinhaltet:
(a) Teilen des Wertes des internen Geheimzustands (**400**, **450**, **475**, **485**) in wenigstens zwei Unterwerte (**405**, **410**, **476**, **477**);
(b) Verschlüsseln (**415**, **460**, **478**) eines ersten der Unterwerte (**410**, **476**) zum Erzeugen eines verschlüsselten Unterwertes;
(c) Verwenden einer XOR-Operation (**420**) zum Kombinieren des verschlüsselten Unterwertes mit einem zweiten der Unterwerte (**405**, **477**), um einen ersten Teil des Ergebnisses der Transformationsoperation zu bilden;
(d) Verschlüsseln (**425**, **460**, **479**) des ersten Ergebnisteils; und
(e) Unterziehen der Verschlüsselung (**425**, **460**, **479**) des ersten Ergebnisteils einem XOR-Operation (**430**) mit dem ersten Unterwert (**410**, **476**), um einen zweiten Teil des Ergebnisses der Transformationsoperation zu erzeugen.

42. Verfahren nach Anspruch 41, wobei die Verschlüsselungsschritte die Anwendung des DES-Algorithmus beinhalten.

43. Verfahren nach einem der Ansprüche 36 bis 39, wobei die Schritte in einer anderen als der gegebenen Reihenfolge ausgeführt werden.

Es folgen 4 Blatt Zeichnungen

**Anhängende Zeichnungen**

FIG. 1

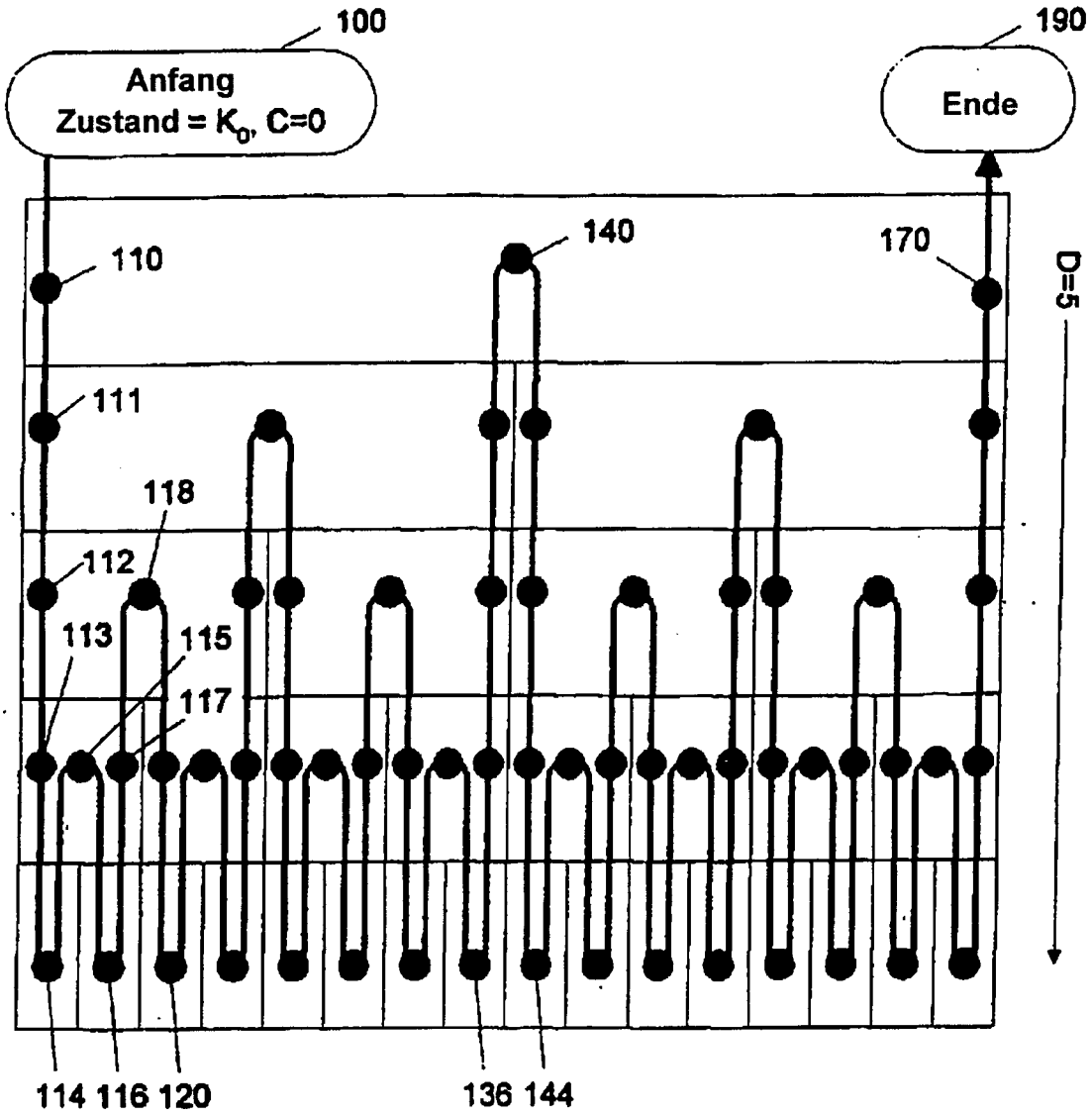


FIG. 2

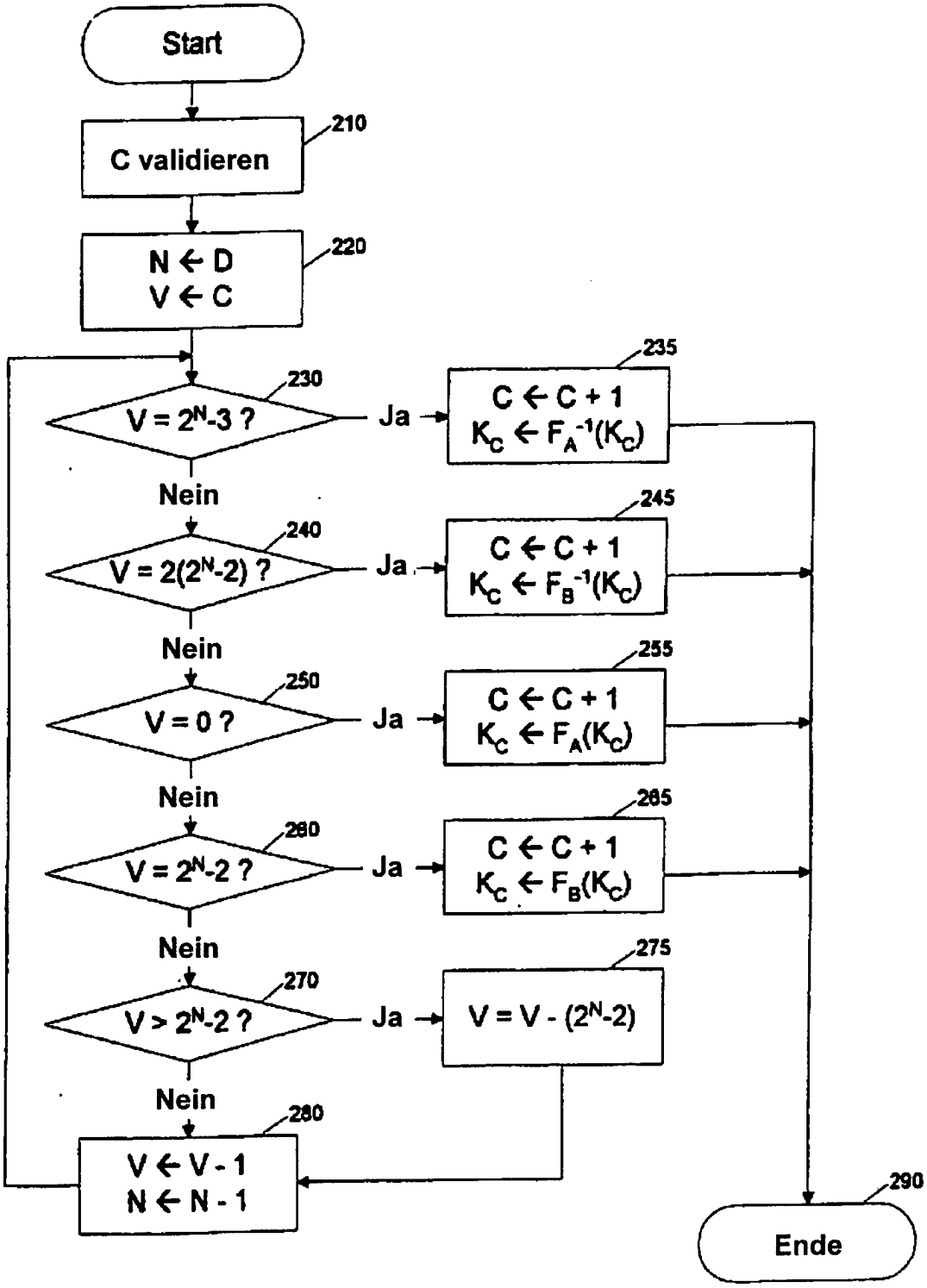

Start
C validieren
210
N ← D
V ← C
220
V = 2^N-3 ?
230
Ja
C ← C + 1
K_C ← F_A^-1(K_C)
235
Nein
V = 2(2^N-2) ?
240
Ja
C ← C + 1
K_C ← F_B^-1(K_C)
245
Nein
V = 0 ?
250
Ja
C ← C + 1
K_C ← F_A(K_C)
255
Nein
V = 2^N-2 ?
260
Ja
C ← C + 1
K_C ← F_B(K_C)
265
Nein
V > 2^N-2 ?
270
Ja
V = V - (2^N-2)
275
Nein
V ← V - 1
N ← N - 1
280
Ende
290

FIG. 3

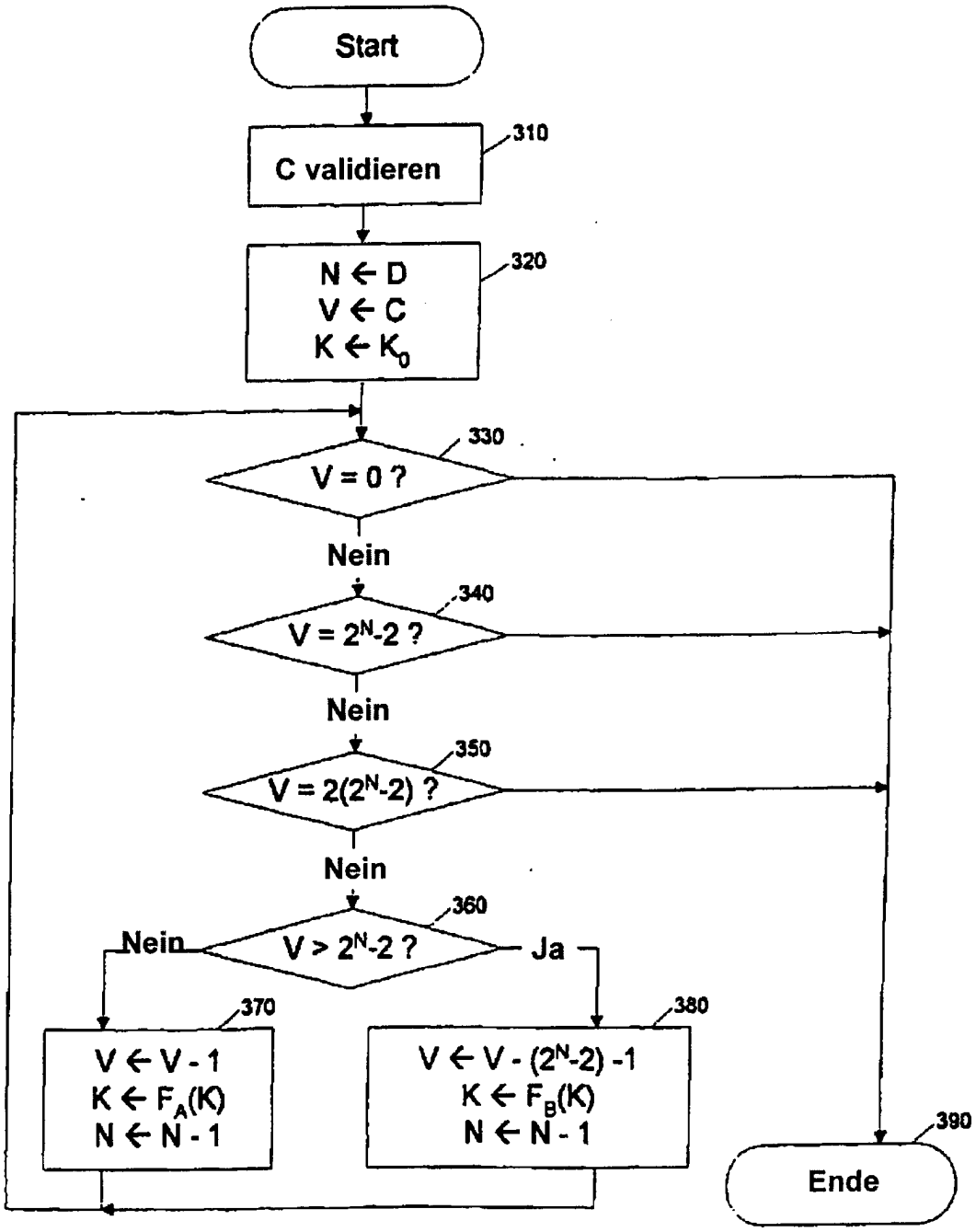
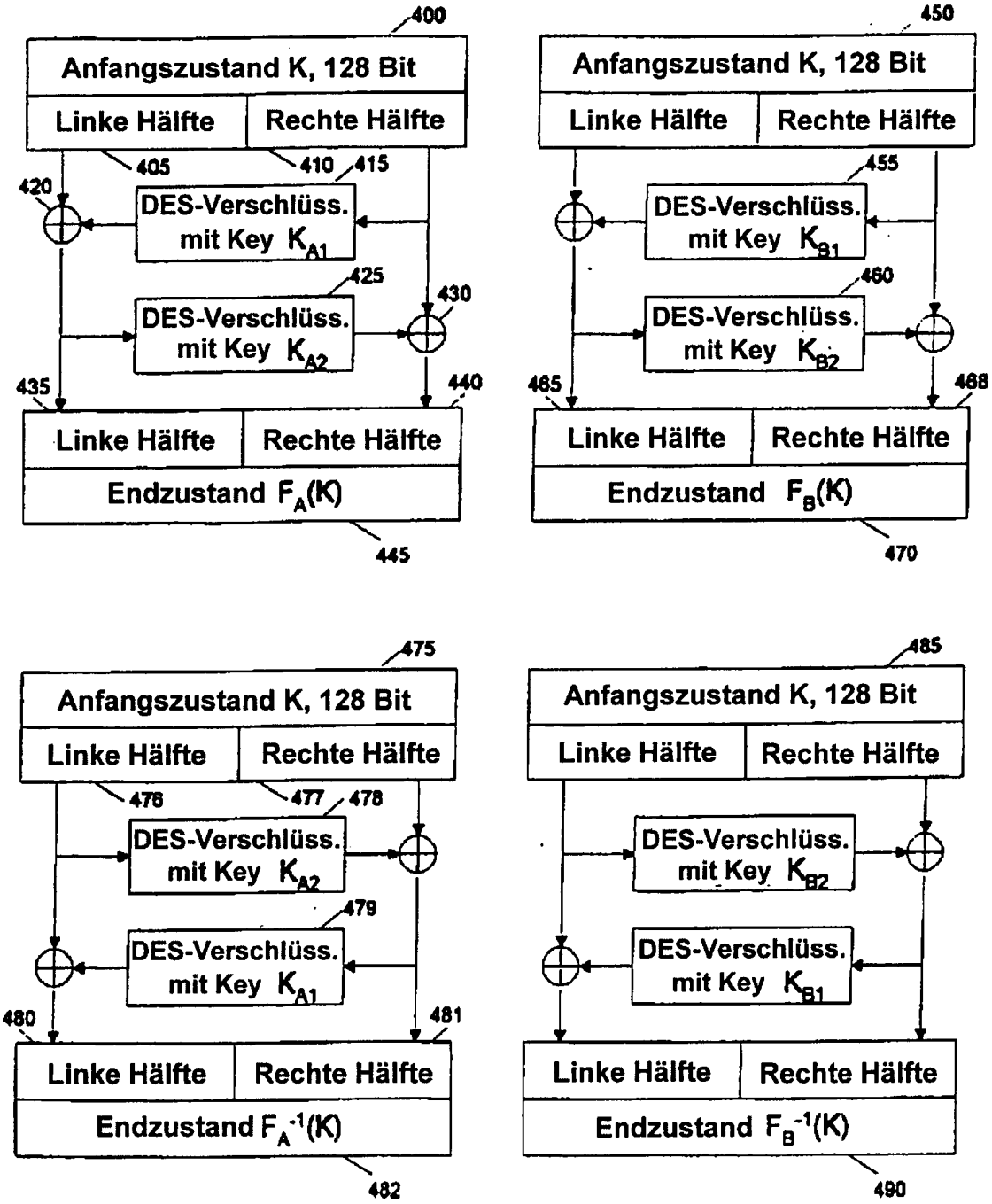

Start
C validieren
310
N ← D
V ← C
K ← K0
320
V = 0 ?
330
Nein
V = 2N-2 ?
340
Nein
V = 2(2N-2) ?
350
Nein
V > 2N-2 ?
360
Nein
Ja
V ← V - 1
K ← FA(K)
N ← N - 1
370
V ← V - (2N-2) -1
K ← FB(K)
N ← N - 1
380
Ende
390

FIG. 4


400
Anfangszustand K, 128 Bit
Linke Hälfte
Rechte Hälfte
405
410
415
420
DES-Verschlüss. mit Key $K_{A1}$
425
DES-Verschlüss. mit Key $K_{A2}$
430
435
440
Linke Hälfte
Rechte Hälfte
Endzustand $F_A(K)$
445
450
Anfangszustand K, 128 Bit
Linke Hälfte
Rechte Hälfte
455
DES-Verschlüss. mit Key $K_{B1}$
460
DES-Verschlüss. mit Key $K_{B2}$
465
468
Linke Hälfte
Rechte Hälfte
Endzustand $F_B(K)$
470
475
Anfangszustand K, 128 Bit
Linke Hälfte
Rechte Hälfte
476
477
478
DES-Verschlüss. mit Key $K_{A2}$
479
DES-Verschlüss. mit Key $K_{A1}$
480
481
Linke Hälfte
Rechte Hälfte
Endzustand $F_A^{-1}(K)$
482
485
Anfangszustand K, 128 Bit
Linke Hälfte
Rechte Hälfte
DES-Verschlüss. mit Key $K_{B2}$
DES-Verschlüss. mit Key $K_{B1}$
Linke Hälfte
Rechte Hälfte
Endzustand $F_B^{-1}(K)$
490