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

(10) **DE 600 26 868 T2** 2006.09.07

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 468 521 B1**
(21) Deutsches Aktenzeichen: **600 26 868.3**
(86) PCT-Aktenzeichen: **PCT/US00/16035**
(96) Europäisches Aktenzeichen: **00 970 432.1**
(87) PCT-Veröffentlichungs-Nr.: **WO 2000/078118**
(86) PCT-Anmeldetag: **09.06.2000**
(87) Veröffentlichungstag der PCT-Anmeldung: **28.12.2000**
(97) Erstveröffentlichung durch das EPA: **20.10.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **22.03.2006**
(47) Veröffentlichungstag im Patentblatt: **07.09.2006**

(51) Int Cl.[8]: ***H04L 9/28*** (2006.01)
H04K 1/00 (2006.01)

(30) Unionspriorität:
329139 **09.06.1999** **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:
Microsoft Corp., Redmond, Wash., US

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(72) Erfinder:
VENKATESAN, Ramarathnam, Redmond, WA 98052, US; JAKUBOWSKI, Mariusz, Bellevue, WA 98007, US

(54) Bezeichnung: **EIN EINFACHES IMPLEMENTIERUNGSVERFAHREN FÜR KRYPTOGRAPHISCHE PRIMITIVA MITTELS ELEMENTAR-REGISTER-OPERATIONEN**

Beschreibung

[0001] Die Erfindung betrifft eine Technik zum Implementieren eines Primitivs zum Berechnen von z.B. einer Prüfsumme. Vorteilhafterweise ist diese Technik relativ einfach und verwendet ziemlich elementare Registeroperationen, wodurch beträchtliche Verarbeitungszeit im Vergleich zu derjenigen gespart wird, die herkömmlicherweise zum Berechnen von z.B. einer Nachrichtensignatur (message authentication code) (MAC) oder zum Implementieren eines Datenketten-Codes (stream cipher) erforderlich ist.

[0002] Viele verschiedene kryptografische Techniken, die sich heute in Einsatz befinden, verwenden mathematische Funktionen, die modulare Arithmetik enthalten, wobei typischerweise ein Restbetrag einer Zahl in Bezug auf eine relativ große Primzahl (M) berechnet wird, wie zum Beispiel $2^{31} – 1$ oder größer. Eine solche beispielhafte Funktion f(x) würde die Form f(x) = ax + b mod(M) in einem Galois-Feld (GF) über 2^n aufweisen, wobei n = 2m + 1, und n und m vordefinierte ganze Zahlen in einem Feld Z(mod M) sind. Obwohl die Funktionen selbst sich von einer Technik zur anderen in hohem Maße unterscheiden, erfordern sie üblicherweise die Berechnung einer mod(M)-Operation der einen oder anderen Form und für gewöhnlich auf einer in hohem Maße repetitiven Basis.

[0003] Solche modularen Operationen werden nicht nur zum Verschlüsseln jedes einzelnen Blocks von Klartext in einer Nachricht, um einen entsprechenden Chiffretext-Block zu ergeben, und zum Entschlüsseln des Letzteren zum Wiedergewinnen des dazugehörigen Klartext-Blocks verwendet, sondern auch zum Berechnen von Zwischenabschnitten der Technik, wie beispielsweise einer Nachrichtensignatur (MAC) oder eines Datenketten-Codes.

[0004] Die Durchführung einer einzelnen mod(M)-Operation kann 10–15 Verarbeitungszyklen erfordern, wenn nicht mehr, (basierend auf dem Wert eines Moduls M). Da eine kryptografische Technik eine große Anzahl solcher Operationen erfordert, kann eine beträchtliche Menge an Verarbeitungszeit, die mit dem Einsatz dieser Technik verbunden ist, für die einfache Berechnung von mod(M)-Operationen aufgewendet werden.

[0005] Kryptografische Techniken werden zunehmend dafür eingesetzt, Informationen in einer großen und zunehmenden Bandbreite von in hohem Maße unterschiedlichen Anwendungen zu schützen, ebenso wie in einer immer größer werdenden Reihe von Vorrichtungen, von hochkomplizierten Mehrzweckvorrichtungen, wie beispielsweise Einzelplatzrechnern und Workstations, bis hin zu relativ einfachen zweckbestimmten Vorrichtungen, wie z.B. "Smart Cards", Fernbedienungen und elektronischen Geräten.

[0006] Zum Beispiel erfährt das Internet (unter anderen Netzwerk-Modalitäten) angesichts der Unkompliziertheit und der geringen Kosten einer Kommunikation mittels elektronischer Post ein explosives und exponentielles Wachstum als ein bevorzugtes Kommunikationsmedium. Da das Internet ein öffentlich zugängliches Netzwerk ist, ist es jedoch nicht sicher, und tatsächlich ist es, und wird es in zunehmendem Maß sein, ein Ziel einer großen Bandbreite von Angriffen von verschiedenen Einzelpersonen und Organisationen mit der Absicht, Internet-Nachrichtenverkehr abzuhören, abzufangen und oder anderweitig zu beeinträchtigen oder sogar zu beschädigen oder illegal in Internet-Sites einzudringen. Diese Sicherheitsgefährdung verstärkt angesichts eines zunehmenden Vertrauens, das in die Verwendung des Internets als einem bevorzugten Kommunikationsmedium gesetzt wird, die Bemühungen in dem Fachgebiet, immer stärkere kryptografische Techniken zu entwickeln, die erhöhte Sicherheitsebenen für elektronische Kommunikation, wie beispielsweise Mail-Nachrichten, Daten- und Computerdateien, in Bezug auf Abhören, Abfangen und mögliche Manipulationen bereitstellen. Demzufolge wird die kryptografische Verarbeitung in eine immer größere Reihe von Einzelplatzrechner-Software, insbesondere Web-Browser und andere Betriebssystemkomponenten sowie Programme für elektronische Post und andere Anwendungsprogramme integriert, um eine sichere Internet-Konnektivität bereitzustellen.

[0007] Eine völlig verschiedene kryptografische Anwendung umfasst so genannte "Smart Cards". Hier verwendet eine zweckbestimmte Vorrichtung von Kreditkartengröße einen eher unkomplizierten und kostengünstigen Mikroprozessor, d.h. eine "Smart Card" speichert Bank- und/oder andere finanzielle Salden für eine entsprechende Einzelperson. Der Mikroprozessor, der ein in der Karte gespeichertes Programm verwendet, kann eine Transaktion validieren und jedes solche Saldo basierend auf der Transaktion ändern. Insbesondere kann die Einzelperson eine elektronische Transaktion mit einem Dritten aufrufen, wie beispielsweise einem Anbieter oder einer Bank, indem die Karte einfach in ein entsprechendes Datenendgerät eingeführt wird und die Transaktionsdaten in eine Tastatur eingegeben werden, die an das Endgerät angeschlossen ist, um die Gesamtheit oder einen Teil des auf der Karte gespeicherten Saldos zu belasten und/oder gutzuschreiben. Eine derartige Transaktion stellt einen unverzüglichen Geldtransfer bereit, wobei jeder Bedarf an und Kosten im Zusammen-

hang mit der Verarbeitung von Papiergeld oder dokumentbasierten Geldinstrumenten, wie beispielsweise Schecks, beseitigt werden. Das gespeicherte Programm verwendet extrem starke kryptografische Techniken, um die auf der Karte gespeicherten Informationen, insbesondere die Salden, vor illegalem Zugriff und Manipulation durch Dritte zu schützen.

[0008] Allerdings, wie oben angemerkt, führt Kryptografie zu einem Verarbeitungs-Overhead. Während bei komplizierten Vorrichtungen mit beträchtlicher Verarbeitungskapazität, wie beispielsweise PCs und Workstations, der Overhead den Gesamtsystemdurchsatz reduziert, kann der Overhead in anderen Vorrichtungen mit eher begrenzter Verarbeitungskapazität, wie beispielsweise Smart Cards, Fernbedienungen und anderen "einfachen" Vorrichtungen, bis zu einem Punkt untragbar werden, dass der Einsatz von ausreichend starken kryptografischen Techniken in solchen Vorrichtungen ausgeschlossen ist.

[0009] Daher besteht angesichts des schnell und anscheinend ständig zunehmenden Bedürfnisses in dem Fachgebiet, kryptografische Techniken in eine große Bandbreite von Vorrichtungen zu integrieren, insbesondere diejenigen, die über eine begrenzte Verarbeitungsleistung verfügen, ein Bedarf in dem Fachgebiet, die Verarbeitungszeit zu reduzieren, die zum Implementieren kryptografischer Techniken erforderlich ist.

[0010] Insbesondere könnte der Verarbeitungs-Overhead, der mit gewissen kryptografischen Techniken verbunden ist, insbesondere beim Berechnen einer Prüfsumme, deutlich reduziert werden, wenn eine mod(M)-Operation durch eine oder mehrere, jedoch weniger prozessorintensive Operationen ersetzt werden könnte. Wenn dieses Ergebnis erzielt werden könnte, dann könnte der Gesamtdurchsatz von hochkomplizierten Vorrichtungen, wie beispielsweise Einzelplatzrechnern und Workstations, die verschiedene kryptografische Techniken verwenden, in vorteilhafter Weise erhöht werden. Des Weiteren, wenn ein solcher Overhead reduziert werden könnte, dann könnten starke kryptografische Techniken in eine Vielzahl von rechnerbezogenen Vorrichtungen integriert werden, die bisher eine unzureichende Verarbeitungsleitung aufwiesen, um derartige Techniken angemessen zu unterstützen.

[0011] Denning, D., "Cryptography and Data Security", 1982, XP002934657, Seiten 59–64, 71–74, 90–93, 135–138 und 147 offenbart einen kryptografischen Prozess zum Verschlüsseln/Entschlüsseln eines Blocks von Klartext/Chiffretext, der unter Verwendung einer affinen Abbildung eine Funktion $f(x) = ax + b \bmod(M)$ implementiert.

[0012] Bosselaers, A., "Comparison of Three Modular Reduction Functions", 1994, CRYPTO '83 offenbart drei modulare Reduktionsalgorithmen für große ganze Zahlen und vergleicht sie in Bezug auf erwartete Ausführungszeiten.

[0013] Es ist die Aufgabe der vorliegenden Erfindung, einen kryptografischen Prozess und eine Vorrichtung zum Verschlüsseln oder Entschlüsseln eines Blocks von digitalem Eingabe-Klartext oder -Chiffretext in einen Block von digitalem Ausgabe-Chiffretext oder -Klartext bereitzustellen, wobei die Verarbeitungszeit aufgrund der mod(M)-Berechnungen reduziert wird.

[0014] Diese Aufgabe wird durch den Gegenstand der selbstständigen Ansprüche gelöst.

[0015] Bevorzugte Ausführungsformen werden durch den Gegenstand der Unteransprüche definiert.

[0016] Vorteilhafterweise erfüllt unsere vorliegende Erfindung diese Anforderung durch Implementieren eines Primitivs zum Berechnen einer Prüfsumme, doch vorteilhafterweise ohne eine mod(M)-Operation zu erfordern.

[0017] In Übereinstimmung mit unseren breitgefassten erfinderischen Lehren ersetzt dieses Primitiv die mod(M)-Operation durch eine Reihe von elementaren Registeroperationen. Diese Operationen umfassen mod-2^n-Multiplikationen, Reihenfolgemanipulationen, (z.B. Byte- oder Wort-Swaps), und Additionen – die alle äußerst einfach zu implementieren sind und sehr wenige Verarbeitungszyklen für die Ausführung erfordern. Mit dem Einsatz unserer erfinderischen Technik kann die Verarbeitungszeit im Vergleich zu derjenigen beträchtlich reduziert werden, die herkömmlicherweise erforderlich ist, um verschiedene kryptografische Parameter zu berechnen, wie beispielsweise eine Nachrichtensignatur (MAC), oder um einen Datenketten-Code zu implementieren.

[0018] Insbesondere basiert eine elementare, beispielhafte und nicht-invertierbare Version unserer Technik auf dem Berechnen eines Primitivs durch die folgende Sequenz von Gleichungen:

$X^S \leftarrow$ Wort-Swap (x)

$y \leftarrow A\,x + B\,x^S \bmod (2^n)$

$y^s \leftarrow$ Wort-Swap (y)

$z \leftarrow C\,y^S + y\,D \bmod (2^n)$

$\theta \leftarrow z + y^S\,E \bmod (2^n)$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist; und
θ eine n-Bit-Zeichenfolge ist.

[0019] Für den Einsatz bei der Generierung eines MAC oder anderen kryptografischen Parametern sind die Koeffizienten "geheim"; wenn sie jedoch verwendet werden, um eine Prüfsumme zu generieren, sind diese Koeffizienten allgemein bekannt.

[0020] Vorteilhafterweise weist unsere erfinderische Technik als ihre Merkmale sowohl invertierbare als auch nicht-invertierbare Varianten auf.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0021] Die Lehren der vorliegenden Erfindung sind unter Berücksichtigung der folgenden ausführlichen Beschreibung in Verbindung mit den folgenden begleitenden Zeichnungen leicht zu verstehen:

[0022] Fig. 1 stellt ein Blockschaltbild des gesamten durchgehenden kryptografischen Prozesses **5** dar, der die vorliegenden erfinderischen Lehren einsetzt, um beispielsweise eine Nachrichtensignatur (MAC) zu generieren;

[0023] Fig. 2 stellt ein Blockschaltbild höchster Ebene einer typischen Internet-basierten Client-Server-Verarbeitungsumgebung dar, die beispielsweise die vorliegende Erfindung einsetzt;

[0024] Fig. 3 stellt ein Blockschaltbild eines in Fig. 2 gezeigten Client-Rechners **100** dar;

[0025] Fig. 4 stellt ein Ablaufdiagramm höchster Ebene des MAC-Generierungsprozesses **400** dar, der in dem in Fig. 1 gezeigten Prozess **5** verwendet wird, um eine MAC in Übereinstimmung mit unseren vorliegenden erfinderischen Lehren zu erzeugen;

[0026] Fig. 5 stellt ein Ablaufdiagramm höchster Ebene der Prozedur Summe alternativ berechnen **500** dar, die statt der Prozedur Summe berechnen **430** verwendet werden kann, die einen Teil des in Fig. 4 gezeigten MAC-Generierungsprozesses **400** bildet;

[0027] Fig. 6A stellt eine typische Wort-Swap-Operation dar, wie sie von unserer vorliegenden Erfindung verwendet werden kann; und

[0028] Fig. 6B stellt eine typische Byte-Swap-Operation dar, wie sie von unserer vorliegenden Erfindung verwendet werden kann.

[0029] Zum leichteren Verständnis wurden identische Bezugszeichen verwendet, wo dies möglich war, um identische Elemente zu bezeichnen, welche die Figuren gemeinsam haben.

DETAILLIERTE BESCHREIBUNG

[0030] Unter Berücksichtigung der folgenden Beschreibung wird der Fachmann klar erkennen, dass die Lehren unserer vorliegenden Erfindung in jeder einer breiten Bandbreite von kryptografischen Techniken verwendet werden können, die das Berechnen einer Prüfsumme umfassen. Solche Techniken sind diejenigen, die z.B. Nachrichtensignaturen (MACs) berechnen oder Datenketten-Codes implementieren.

[0031] Um dem Leser das Verständnis zu erleichtern, werden wir unsere Erfindung im Zusammenhang mit

ihrer Verwendung in einer solchen Technik erläutern, wenn auch in sehr allgemeiner Form, die in einer Client-Server-Transaktionsverarbeitungsumgebung verwendet werden könnte, in der Transaktionsnachrichten über ein unsicheres Kommunikationsnetzwerk übertragen werden müssen, wie beispielsweise das Internet, und insbesondere im Zusammenhang mit der Berechnung einer MAC, die in dieser Technik verwendet wird.

A. Übersicht

[0032] Fig. 1 stellt ein Blockschaltbild des gesamten durchgehenden kryptografischen Prozesses **5** dar, der eine MAC unter Verwendung unserer vorliegenden Erfindung generiert.

[0033] Wie gezeigt, werden eingehende Klartext-Informationen in so genannte "Nachrichten" gegliedert. Jede solche Nachricht **7**, die als P bezeichnet wird, wird als N Blöcke (P_1, P_2, ..., P_N) gegliedert, wobei jeder Block n Bits in der Breite beträgt, wobei n hier beispielsweise 32 Bits ist. Jeder derartige Klartext-Block wird, wie durch die Linie **10** symbolisch dargestellt, auf den Verschlüsselungsprozess **20** angewendet. Dieser Prozess umfasst beispielsweise den Nachrichten-Verschlüsselungsprozess **23** und den erfinderischen MAC-Generierungsprozess **400**. Der Prozess **400**, (der im Folgenden ausführlich in Verbindung mit Fig. 4 und Fig. 5 beschrieben wird), generiert unter Vorgabe der Klartext-Nachricht P oder einer geeigneten kryptografischen Manipulation davon als Eingabe in Übereinstimmung mit unserer Erfindung eine MAC, die typischerweise 64 Bits lang ist, die für diese Nachricht eindeutig ist. Der Nachrichten-Verschlüsselungsprozess **23** verschlüsselt die Klartext-Nachricht in Chiffretext und fügt die 64-Bit-MAC in geeigneter Weise in die Nachricht ein, beispielsweise als zwei Blöcke der höchsten Ordnung (C_{N-1}, C_N), (ein Komma, das aufeinander folgende Werte in Klammern trennt, wird hierin im Folgenden als ein Operator zum Bezeichnen der Verkettung dieser Werte verwendet), um die Chiffretext-Nachricht C zu ergeben. Die zwei Blöcke höchster Reihenfolge bilden zusammen die MAC **42**. Abhängig von einem spezifischen Verschlüsselungsprozess, der im Prozess **23** verwendet wird, kann die MAC **42** selbst verschlüsselt werden, wie beispielsweise über eine bekannte DES- (Data Encryption Standard) Verschlüsselung oder eine andere herkömmliche pseudo-zufällige Permutation, oder auch nicht. Die Chiffretext-Nachricht wird aus N aufeinander folgenden n-Bit-Blöcken von Chiffretext ausgebildet.

[0034] Die sich daraus ergebende Chiffretext-Nachricht C wird dann gespeichert oder mittels einer vorgegebenen Modalität, z.B. einem unsicheren Kommunikationskanal, der durch die gestrichelte Linie **45** dargestellt und durch eine Internet-Verbindung verkörpert wird, zu einer Empfänger-Speicherstelle übertragen. Hier wird eine empfangene Version der Chiffretext-Nachricht, die als $\tilde{C}$ bezeichnet wird, (auch gekennzeichnet als Nachricht **40'**), durch den Entschlüsselungsprozess **50** entschlüsselt, um eine wiedergewonnene Klartext-Nachricht **70** zu ergeben, die auch als Klartext-Nachricht $\hat{P}$ bezeichnet wird, die, um gültig und damit für die Stromabwärts-Verwendung geeignet zu sein, in allen Aspekten mit der ursprünglichen Klartext-Nachricht P identisch sein muss. Der Entschlüsselungsprozess **50** enthält den Nachrichten-Entschlüsselungsprozess **60**, den MAC-Generierungsprozess **400** und den Identitäts-Komparator **90**.

[0035] Um zu bestimmen, ob die wiedergewonnene Klartext-Nachricht gültig ist, z.B. nicht geändert worden ist, erzeugt der Nachrichten-Entschlüsselungsprozess **60** nicht nur den wiedergewonnenen Klartext, sondern extrahiert (und entschlüsselt, falls erforderlich,) die MAC aus der Chiffretext-Nachricht $\tilde{C}$. Eine sich daraus ergebende MAC wird, wie durch die Linie **67** symbolisch dargestellt, auf einen Eingang des Komparators **90** angewendet. Der wiedergewonnene Klartext wird ebenfalls, wie durch Linie **77** symbolisch dargestellt, auf den MAC-Generierungsprozess **400** angewendet. Der Prozess **400** berechnet die MAC aus der wiedergewonnenen Klartext-Nachricht $\hat{P}$ neu und wendet, wie durch Linie **80** symbolisch dargestellt, eine sich daraus ergebende neu berechnete MAC auf einen anderen Eingang des Komparators **90** an. Wenn beide dieser MACs, die dann auf die entsprechenden Eingänge des Komparators **90** angewendet sind, miteinander identisch übereinstimmen, generiert der Komparator **90** eine entsprechende Meldung am Ausgang **93**, um anzugeben, dass die wiedergewonnene Klartext-Nachricht $\hat{P}$, die dann an der Ausgangsleitung **73** (output lead) erscheint, für nachfolgende Verwendung gültig ist. Wenn die wiedergewonnenen und neu berechneten MACs jedoch nicht miteinander übereinstimmen, generiert der Komparator **90** eine entsprechende Meldung am Ausgang **97**, um anzugeben, dass die wiedergewonnene Klartext-Nachricht $\hat{P}$, die dann am Ausgang **73** erscheint, ungültig ist und ignoriert werden sollte. Insofern, abgesehen von der Generierung einer MAC, als die spezifische Natur der Verschlüsselungs- und Entschlüsselungstechniken, die jeweils in dem Verschlüsselungsprozess **23** und dem Nachrichten-Entschlüsselungsprozess **60** verwendet werden, für die vorliegende Erfindung und jede einer großen Bandbreite solcher Techniken irrelevant sind, erfolgreich verwendet werden kann, werden wir diese Aspekte nicht weiter ausführlich erläutern. Trotzdem beschreiben und beanspruchen wir eine solche beispielhafte kryptografische Technik in unseren gleichzeitig anhängigen Patentanmeldungen in den Vereinigten Staaten, (auf die der Leser verwiesen wird), mit dem Titel: "Cryptografic Technique That Provides Fast Encryption and Decryption and Assures Integrity of a Ciphertext Message", eingereicht am 20. April 1998, Serialnummer

09/062,836, veröffentlicht unter US 6226742 am 01.05.01; und "Method and Apparatus for Producing A Message Authentication Code", eingereicht am 20. April 1998, Serialnummer 09/062,837 veröffentlicht unter US 6128737 am 03.10.01 – von denen beide dem gemeinsamer Rechtsnachfolger hiervon übertragen sind.

B. Beispielhafte Verarbeitungsumgebung

[0036] Unter Berücksichtigung des Vorgenannten wird auf Fig. 2 Bezug genommen, die ein Blockschaltbild höchster Ebene einer Client-Server-Verarbeitungsumgebung **200** darstellt, welche die vorliegende Erfindung einsetzt.

[0037] Wie gezeigt, enthält diese Umgebung einen Rechner **205**, der den Server **210** implementiert, wobei Letzterer beispielsweise ein Web-Server ist. Eine Reihe von einzelnen, entfernt angeordneten Client-Rechnern, von denen jeder beispielsweise ein Einzelplatzrechner (PC) ist, von denen nur ein solcher Client, d.h. der Client-Rechner **100**, speziell gezeigt ist, ist unter Verwendung entsprechender Kommunikationskanäle, wie beispielsweise die Kanäle **140** und **160**, über ein unsicheres Kommunikationsnetzwerk, hier beispielsweise als Internet **150** gezeigt, mit dem Rechner **205** verbunden. Ein (nicht speziell dargestellter) Benutzer, der sich am Client-Rechner **100** befindet und Informationen von dem Server erhalten möchte, kann das entsprechende Client-Programm **130** am Client-Rechner **100** aufrufen. Das Client-Programm bildet eines von einer Reihe von Anwendungsprogrammen **120**, die insgesamt darin resident sind und durch den Client-Rechner **100** ausgeführt werden. Obwohl das Client-Programm speziell als in den Anwendungsprogrammen resident gezeigt ist, kann Ersteres auch als eine Komponente, wie beispielsweise ein Web-Browser eines Betriebssystems (O/S), implementiert werden, zum Beispiel von dem in Fig. 3 gezeigten O/S **337**. Der in Fig. 2 gezeigte Server **210** kann jede einer breiten Bandbreite von Anwendungsfunktionen implementieren, einschließlich beispielsweise eines Servers für Handel, eines Servers für Bankgeschäfte, eines Servers für elektronische Post oder für Dateien. In Bezug auf elektronischen Handel beabsichtigt der Benutzer eventuell eine Geschäftstransaktion über den Client-Rechner **100** und den Server **210** ausführen, was die Bereitstellung, (wie durch Linie **110** symbolisch dargestellt), von Informationen für den Server, wie beispielsweise eine Kontonummer des Benutzers bei einem Geldinstitut und Zahlungsanweisungen, um Mittel zu einem Zahlungsempfänger zu transferieren, oder das Erhalten von Formationen, (wie durch die Linie **135** symbolisch dargestellt), von dem Server umfasst, wie beispielsweise verfügbare Konten- oder Kreditsalden des Benutzers, die in jedem Fall für diesen Benutzer vertraulich sind. Alternativ kann der Server **210** ein Dateiserver sein, der für den Benutzer den Zugang zu verschiedenen Dateien, die in einem Verwahrungsort gespeichert sind, bereitstellt, von denen jede vom Benutzer heruntergeladen werden kann. Sobald eine solche Datei heruntergeladen ist, kann sie in dem Speicher **330** gespeichert werden, (siehe Fig. 3), der sich in dem Client-Rechner **100** zur dortigen lokalen Verwendung befindet. Eine solche Datei kann jedoch auch geschützte und/oder vertrauliche Informationen enthalten, für die ihr Besitzer den Benutzerzugriff steuern möchte. Beispielsweise kann eine solche Datei eine sich selbst installierende ausführbare Datei für eine Aktualisierung für ein bestimmtes Problem sein, auf die ihr Besitzer, z.B. ein Software-Hersteller, einen illegalen öffentlichen Zugriff verhindern möchte, d.h. verhindern möchte, dass die Aktualisierung von einer Person verwendet wird, die dafür keine entsprechende Bezahlung geleistet hat. Der Server **210** selbst, wie in Fig. 2 gezeigt, kann ebenfalls vertrauliche oder geschützte Informationen, (wie durch Linie **215** symbolisch dargestellt), die von dem Benutzer stammen (und über das Netzwerk (hier das Internet) **150** an den Server übertragen werden), an stromabwärts liegende, (nicht speziell gezeigte) Ausrüstung zur anschließenden Verarbeitung bereitstellen oder, (wie durch Linie **218** symbolisch dargestellt), vertrauliche oder geschützte Informationen von der stromabwärts liegenden Ausrüstung zur möglichen Übertragung über das Netzwerk an den Benutzer empfangen.

[0038] Das Netzwerk **150**, das beispielsweise das Internet ist, ist anfällig für eine Beeinträchtigung durch Dritte. In dieser Hinsicht könnten Dritte eine in herkömmlicher Weise verschlüsselte Nachricht abfangen, die dann über das Netzwerk geleitet wird und z.B. vom Client-Rechner **100** für z.B. eine laufende Finanztransaktion kommt und einen dort befindlichen Benutzer betrifft. Obwohl Dritte eventuell nicht über ausreichende Ressourcen hinsichtlich der verfügbaren Verarbeitungskapazität oder die Zeit zum Knacken eines herkömmlichen Codes verfügen, der zum Verschlüsseln von Nachrichten und Wiedergewinnen des in der übertragenen Nachricht inhärenten Klartexts verwendet wird, können Dritte trotzdem über ausreichende Kenntnis der Chiffretext-Nachricht, insbesondere ihrer strukturellen Gliederung, und die Ausrüstung verfügen, die erforderlich ist, um die Nachricht zum Schaden des Benutzers erfolgreich zu verändern. Diesbezüglich könnten Dritte illegal die Chiffretext-Nachricht manipulieren, indem ein oder mehrere vordefinierte Chiffretext-Blöcke für entsprechende ursprüngliche Chiffretext-Blöcke ersetzt werden und dann eine sich daraus ergebende modifizierte Chiffretext-Nachricht wieder auf das Netzwerk zur Weiterleitung an den Rechner **205** zurückübertragen wird, um dort verarbeitet zu werden.

[0039] Um die vertrauliche oder geschützte Natur der Informationen, die über das Netzwerk **150** zwischen dem Client-Rechner **100** und dem Rechner **205** übertragen werden, vor dem Zugriff Dritter zu schützen, verwenden sowohl das Client-Programm **130** als auch der Server **210** jeweils eine kryptografische Kommunikation durch Integration des darin enthaltenen Verschlüsselungsprozesses **20** und des Entschlüsselungsprozesses **50**. Daher werden Nachrichten, die zur Weiterleitung über das Netzwerk bestimmt sind und von einer gleichrangigen Netzwerk-Anwendung, entweder dem Client-Programm **130** oder dem Server **210**, generiert werden, jeweils durch den darin enthaltenen Verschlüsselungsprozess **20** verschlüsselt, um entsprechende Chiffretext-Nachrichten mit eingebetteten MACs zu ergeben, die dann wiederum jeweils über das Netzwerk **150** zu der anderen gleichrangigen Netzwerk-Anwendung übertragen werden. Auf ähnliche Weise werden von dem Netzwerk empfangene Chiffretext-Nachrichten von jedem der Gleichrangigen durch den darin enthaltenen Entschlüsselungsprozess **50** entschlüsselt, um eine entsprechende wiedergewonnene Klartext-Nachricht und eine Angabe bezüglich ihrer Gültigkeit zu ergeben. Die Verschlüsselungs- und Entschlüsselungs-Prozeduren **20** und **50** sind zueinander umgekehrte Prozeduren.

C. Client-Rechner **100**

[0040] Fig. 3 stellt ein Blockschaltbild eines Client-Rechners (PC) **100** dar.

[0041] Wie gezeigt, umfasst der Client-Rechner **100** Eingabeschnittstellen (I/F) **320**, einen Prozessor **340**, eine Kommunikationsschnittstelle **350**, einen Speicher **330** und Ausgabeschnittstellen **360**, die alle in herkömmlicher Weise durch den Bus **370** verbunden sind. Der Speicher **330** umfasst im Allgemeinen verschiedene Modalitäten, einschließlich beispielsweise einem Direktzugriffsspeicher (RAM) **332** zum temporären Speichern von Daten und Befehlen, Diskettenlaufwerken) **334** zum Austauschen von Informationen per Benutzerbefehl mit Floppy-Disks, und einem nicht-flüchtigen Massenspeicher **335**, der über eine Festplatte implementiert wird und normalerweise magnetischer Natur ist. Der Massenspeicher **335** kann auch eine CD-ROM oder eine andere Lesevorrichtung für (nicht speziell gezeigte) optische Medien (oder eine Schreibvorrichtung) enthalten, um aus entsprechenden optischen Speichermedien Informationen auszulesen (oder auf sie Informationen zu schreiben). Der Massenspeicher speichert das Betriebssystem (O/S) **337** und die Anwendungsprogramme **120**; die Letzteren enthalten beispielsweise das Client-Programm **130**, (siehe Fig. 2), das unsere erfinderische Technik integriert. Das in Fig. 3 gezeigte O/S **337** kann durch jedes herkömmliche Betriebssystem implementiert werden, wie beispielsweise das Betriebssystem WINDOWS NT ("WINDOWS NT" ist ein eingetragenes Warenzeichen der Microsoft Corporation von Redmond, Washington). Wir werden aus dem Grund keine Komponenten des O/S **337** erläutern, da sie alle irrelevant sind. Es genügt zu erwähnen, dass das Client-Programm als eines der Anwendungsprogramme **120** unter der Steuerung des O/S ausgeführt wird.

[0042] Wenn unsere vorliegende erfinderische Technik zur Verwendung in kryptografische Verschlüsselungs- und Entschlüsselungs-Module eingebettet wird, spart sie vorteilhafterweise Verarbeitungszeit, wodurch der Durchsatz sowohl des Client-Rechners **120** als auch des Servers **210** (siehe Fig. 2) erhöht wird.

[0043] Wie in Fig. 3 gezeigt, können eingehende Informationen von zwei beispielhaften externen Quellen stammen: Informationen, die vom Netzwerk, z.B. vom Internet und/oder anderen Netzwerkeinrichtungen, über die Netzwerk-Verbindung **140** zur Kommunikationsschnittstelle **350** oder von einer zweckbestimmten Eingabequelle über Pfade) **310** zu Eingabeschnittstellen **320** zugeführt werden. Eine zweckbestimmte Eingabe kann von einer breiten Bandbreite von Quellen stammen, z.B. einer externen Datenbank. Des Weiteren können Eingabe-Informationen in Form von Dateien oder darin enthaltenem spezifischem Inhalt ebenfalls durch Einsetzen einer Diskette, welche die Informationen enthält, in das Diskettenlaufwerk **334** bereitgestellt werden, von dem aus der Rechner **100** unter Benutzerbefehl auf diese Informationen auf der Diskette zugreift und sie liest. Die Eingabeschnittstellen **320** enthalten entsprechende Schaltungen zum Bereitstellen der notwendigen und entsprechenden elektrischen Verbindungen, die erforderlich sind, um jede verschiedene zweckbestimmte Quelle von Eingabe-Informationen physikalisch mit dem Rechnersystem **100** zu verbinden und mit ihm eine Schnittstelle auszubilden. Unter der Steuerung des Betriebssystems tauschen die Anwendungsprogramme **120** Befehle und Daten mit externen Quellen über die Netzwerkverbindung **140** oder den (oder die) Pfade) **310** aus, um Informationen zu senden oder zu empfangen, die typischerweise von einem Benutzer während der Programmausführung angefordert werden.

[0044] Die Eingabeschnittstellen **320** verbinden auch die Benutzer-Eingabevorrichtungen **395**, wie beispielsweise eine Tastatur und eine Maus, elektrisch mit dem Rechnersystem **100** und bilden eine Schnittstelle mit ihm aus. Die Anzeigevorrichtung **380**, wie beispielsweise ein herkömmlicher Farbmonitor, und der Drucker **385**, wie beispielsweise ein herkömmlicher Laserdrucker, werden jeweils über die Leitungen **363** und **367** an die Ausgabeschnittstellen **360** angeschlossen. Die Ausgabeschnittstellen stellen die erforderlichen Schaltkrei-

se für den elektrischen Anschluss der Anzeigevorrichtung und des Druckers an das Rechnersystem und zur Ausbildung von Schnittstellen mit diesem bereit. Es ist klar, dass unsere vorliegende erfinderische kryptografische Technik mit jeder Art von digitalen Informationen arbeiten kann, ungeachtet der Modalitäten, durch die der Client-Rechner **100** diese Informationen erhält, speichert und/oder überträgt.

[0045] Des Weiteren, da die spezifischen Hardware-Komponenten des Rechnersystems **100** sowie alle anderen Aspekte der im Speicher **335** gespeicherten Software, ausgenommen die Module, welche die vorliegende Erfindung implementieren, herkömmlicher Art und bekannt sind, werden sie nicht weiter ausführlich erläutert. Im Allgemeinen weist der Rechner **205** eine Architektur auf, die derjenigen des Client-Rechners **100** ziemlich ähnlich ist.

D. Einschränkungen durch Modulo-Arithmetik in herkömmlichen kryptografischen Techniken

[0046] Herkömmliche kryptografische Techniken verwenden häufig als ein Primitiv eine Prüfsumme, die das Berechnen von mod(M) erfordert, wobei M eine große Primzahl ist, wie beispielsweise 2^{31}–1 oder größer.

[0047] Leider erfordert eine mod(M)-Operation für die Berechnung eine Größenordung von wenigstens 10–15 Maschinenzyklen, wenn nicht mehr (basierend auf dem Wert des Moduls M). Diese Funktion wird während der beiden herkömmlichen Verschlüsselungs- und Entschlüsselungs-Operationen wiederholt berechnet. Wenn daher eine solche Funktion auf einer Vorrichtung mit beträchtlicher Verarbeitungskapazität implementiert wäre, wie beispielsweise einem PC oder einer Workstation, würden die mod(M)-Berechnungen den Gesamtdurchsatz reduzieren, vielleicht sogar merklich. Dieser Berechnungs-Overhead kann untragbar sein in Vorrichtungen, die eine ziemlich begrenzte Verarbeitungskapazität aufweisen, und schließt daher den Einsatz dieser kryptografischen Technik in diesen Vorrichtungen – in denen ihr Einsatz von großem Nutzen sein könnte – aus.

E. Unsere erfinderische Technik und ihre Implementierung

[0048] Unter Würdigung dieses Mangels des Stands der Technik haben wir eine Technik zum Implementieren einer Prüfsumme entwickelt, die vorteilhafterweise keine mod(M)-Operation erfordert.

[0049] Unsere Technik implementiert die Prüfsumme als eine relativ einfache Reihe von elementaren Registeroperationen. Diese Operationen umfassen mod-2^n-Multiplikationen, Reihenfolgemanipulationen, (die eine Operation sind, welche die Bit-Reihenfolge in einem Block ändert, wie zum Beispiel Byte- oder Wort-Swaps) und Additionen – die alle äußerst einfach zu implementieren sind und sehr wenige Verarbeitungszyklen für die Ausführung erfordern. Die Operationen, die in dem Primitiv verwendet werden, können auch ziemlich effektiv in Pipeline-Verarbeitung bearbeitet werden. Daher kann die Verwendung des Primitivs auf der Basis unserer Erfindung, insbesondere bei Pipeline-Verarbeitung, die Verarbeitungszeit gegenüber derjenigen beträchtlich reduzieren, die herkömmlicherweise zum Berechnen von verschiedenen kryptografischen Parametern erforderlich ist, wie beispielsweise einer Nachrichtensignatur (MAC), oder zum Implementieren eines Datenketten-Codes. Wir glauben, dass unsere erfinderische Technik ebenfalls vorteilhaft in gewisse Codes integriert werden kann, um die Sicherheit dieser Codes gegenüber Klartext-Chiffretext-Angriffen zu erhöhen.

[0050] Wir beginnen mit den folgenden mathematischen Definitionen: $F(x) = \theta$, und einem hochgestellten "S", das wie z.B. in x^S entweder eine entsprechende Byte- oder Wort-Swap-Operation bezeichnet.

[0051] Um kurz abzuschweifen, stellen **Fig. 6A** und **Fig. 6B** jeweils Wort-Swap- und Byte-Swap-Operationen dar. Unter Vorgabe eines n-Bit-Blocks **610**, (der beispielsweise 32 Bits lang ist), mit zwei 16-Bit-Wörtern (z.B. den Wörtern **613** und **617**, die jeweils auch mit L und R für "links" und "rechts" bezeichnet werden), erzeugt eine Wort-Swap-Operation, die durch die Linie **620** symbolisch dargestellt wird, einen n-Bit-Block **630**, bei dem diese Wörter ihre Position vertauscht haben, (d.h. mit den Wörtern **633** und **639**, die jeweils mit den Wörtern **617** und **613** identisch sind). Eine solche Operation kann in einem Verarbeitungszyklus implementiert werden, indem die einzelnen Wörter einfach vertauscht werden, wie durch den Pfeil **625** gezeigt. Unter Vorgabe des n-Bit-Blocks **650**, (der ebenfalls beispielsweise 32 Bits lang ist), mit einzelnen Acht-Bit-Bytes **652**, **564**, **656** und 658 (auch jeweils als Bytes A, B, C D bezeichnet), erzeugt eine Byte-Swap-Operation, die symbolisch durch die Linie **660** dargestellt wird, den n-Bit-Block **670**, wobei die Reihenfolge dieser vier Bytes umgekehrt ist, (d.h. die Bytes **672**, **674**, **676** und **678** sind jeweils identisch mit den Bytes **658**, **656**, **654** und **652**). Die Byte-Swap-Operation kann in einem Verarbeitungszyklus implementiert werden, indem einzelne Bytes parallel vertauscht werden, wie durch die Pfeile **665** gezeigt.

[0052] Unter Berücksichtigung dieser Definitionen berechnet eine nicht-invertierbare Version des Primitivs

F(x), das eine Prüfsumme, insbesondere $f(x) = a_x + b \bmod(M)$ implementiert, die Gleichungen (1)–(5) in Übereinstimmung mit unseren erfinderischen Lehren wie folgt der Reihe nach:

$$x^S \leftarrow \text{Wort-Swap}(x) \quad (1)$$

$$y \leftarrow A\,x + B\,x^S \bmod (2^n) \quad (2)$$

$$y^S \leftarrow \text{Wort-Swap}(y) \quad (3)$$

$$z \leftarrow C\,y^S + y\,D \bmod (2^n) \quad (4)$$

$$\theta \leftarrow z + y^S\,E \bmod (2^n) \quad (5)$$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist; und
θ eine n-Bit-Zeichenfolge ist.

[0053] Wie ersichtlich ist, werden diese Gleichungen unter Verwendung elementarer Registeroperationen implementiert, d.h. Reihenfolgemanipulationen (z.B. Wort- oder Byte-Swaps, Additionen und mod(2^n)-Multiplikationen). Demzufolge können diese Operationen unter Verwendung von relativ wenigen Verarbeitungszyklen durchgeführt werden – mit Sicherheit weniger als die 10–15 Zyklen, die für die Durchführung einer mod(M)-Operation erforderlich sind. Obwohl wir in den Gleichungen (1) und (3) die Verwendung von Wort-Swap-Operationen gezeigt haben, könnten stattdessen auch Byte-Swap-Operationen (oder auch andere Manipulationen, welche die Bit-Reihenfolge verändern) verwendet werden. Für die Verwendung bei der Generierung einer MAC oder anderer verschiedener kryptografischer Ausdrücke sind die Koeffizientenwerte A, B, C, D und E "geheime" Werte, d.h. sie werden nicht öffentlich gemacht.

[0054] Eine invertierbare Version des Primitivs F(x), das f(x) ebenfalls in Übereinstimmung mit unseren erfinderischen Lehren über die Gleichungen (6)–(15) implementiert, ist wie folgt:

$$y \leftarrow A\,x \bmod (2^n) \quad (6)$$

$$y^S \leftarrow \text{Wort-Swap}(y) \quad (7)$$

$$z \leftarrow B\,y^S \bmod (2^n) \quad (8)$$

$$z^S \leftarrow \text{Wort-Swap}(z) \quad (9)$$

$$v \leftarrow C\,z^S \bmod (2^n) \quad (10)$$

$$v^S \leftarrow \text{Wort-Swap}(v) \quad (11)$$

$$w \leftarrow D\,v^S \bmod (2^n) \quad (12)$$

$$w^S \leftarrow \text{Wort-Swap}(w) \quad (13)$$

$$t \leftarrow E\,w^S \bmod (2^n) \quad (14)$$

$$\theta \leftarrow t + L\,y^S \bmod (2^n) \quad (15)$$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist; und
L eine zufällige ganze Zahl ist, die kleiner oder gleich 2^n ist.

[0055] Auch hier sind bei der Generierung einer MAC oder anderer verschiedener kryptografischer Ausdrücke alle Koeffizientenwerte A, B, C, D und E "geheime" Werte. Alternativ könnten die Gleichungen (6)–(12) verwendet werden, um das Primitiv mit F(x) = w zu implementieren. Der Weiteren könnte eine "umgekehrte" Operation, (bei der alle Bits in einem Block in ihrer Abfolge vollständig verändert werden) – was einem anderen Typ von Reihenfolgemanipulation entspricht – statt eines Byte- oder Wort-Swap verwendet werden. Zum Beispiel könnte das Primitiv F(x) für eine invertierbare Form von f(x) in Übereinstimmung mit unserer Erfindung durch

die Gleichungen (16)–(19) wie folgt implementiert werden:

$$y \leftarrow H\ x \bmod (2^n) \tag{16}$$

$$z \leftarrow \text{Umkehrung } (y) \tag{17}$$

$$s \leftarrow J\ z \bmod (2^n) \tag{18}$$

$$\theta \leftarrow S + K \bmod (2^n) \tag{19}$$

wobei: die Koeffizienten H, J und K jeweils eine zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist.

[0056] Wenn dieses Primitiv dazu verwendet würde, eine MAC oder einen anderen kryptografischen Ausdruck zu generieren, wären die Koeffizienten H, J und K "geheime" Werte. Da eine Umkehr-Operation im Vergleich zu einer Byte- oder Wort-Swap-Operation relativ langsam ist, wird die Verwendung von Primitiven, die durch die oben genannten Gleichungen (6)–(12) oder (6)–(15) vorgegeben werden, gegenüber denjenigen bevorzugt, die durch die Gleichungen (16)–(19) vorgegeben werden.

[0057] Es ist offensichtlich, dass der Fachmann basierend auf der obigen Beschreibung problemlos verschiedene andere Primitive F(x) entwickeln kann, die gleichwertige kryptografische Merkmale für f(x) = ax + b mod(M) bereitstellen und die in Übereinstimmung mit unserer Erfindung mod-2^n-Multiplikationen, Reihenfolgemanipulationen und Additionen – aber keine mod(M)-Operation – verwenden und daher die oben beschrieben spezifischen Primitiven ersetzen können.

[0058] Wie oben erläutert, kann ein allgemeines Primitiv, das auf unserer erfinderischen Technik basiert, zum Generieren einer MAC verwendet werden. Dazu wird eine Reihe von Primitiven $F_1(x)$, $F_2(x)$, ..., $F_p(x)$ für die Funktion f(x), die nicht-invertierbar sind und die gleiche Form wie oben angegeben (wie F(x)) aufweisen, ausgewählt, jedoch mit verschiedenen Werten für die entsprechenden "geheimen" Koeffizienten, d.h. wenn $F_1(x)$ "geheime" Koeffizienten A, B, C, D und E besitzt, dann besitzt $F_2(x)$ "geheime" Koeffizienten a, b, c, d und e und so weiter. Danach werden unter Vorgabe einer Eingabesequenz $X = x_1, x_2, ..., x_N$ von n-Bit-Zeichenketten entsprechende Ausgabewerte (Zwischenergebnisse) $Y = y_1, y_2, ..., y_N$ gemäß den Gleichungen (20)–(25) unter Verwendung von aufeinander folgenden dieser p Primitiven, (wobei $p < n$), für die entsprechenden aufeinander folgenden Eingabewerte x_i wie folgt berechnet:

$$y_1 = F_1(x_1) \tag{20}$$

$$y_2 = F_2(x_2 + y_1) \tag{21}$$

$$y_3 = F_3(x_3 + Y_2) \tag{22}$$

$$y_p = F_p(x_p + y_{p-1}) \tag{23}$$

$$Y_{p+1} = F_1(y_p + x_{p+1}) \tag{24}$$

$$y_{p+2} = F_2(y_{p+1} + x_{p+2}) \tag{25}$$

[0059] Die MAC kann dann in Übereinstimmung mit der Gleichung (26) als eine Funktion der Zwischenergebnisse wie folgt gebildet werden:

$$MAC = \left(y_N, \sum_{i=1}^{N} y_i \right) \tag{26}$$

[0060] Zur erhöhten Sicherheit kann die Gleichung (25) durch Einführen einer geheimen oder zufälligen Permutation (γ_i) für jeden y_i Ausdruck in der Summe wie in Gleichung (27) gezeigt wie folgt modifiziert werden:

$$MAC = \left(y_N, \sum_{i=1}^{N} \gamma_i y_i \right) \tag{27}$$

wobei: γ_i zufällig oder als ein "geheimer" vordefinierter Wert in einem Bereich von einschließlich ± k ausgewählt

wird, d.h. $\gamma_i \in \{k, k-1, k-2, ..., 0. -1, -2, ..., -k\}$, wobei k eine vordefinierte ganze Zahl ist. Aus Gründen der Einfachheit kann jedes γ_i auf den Wert +1 oder –1 gesetzt werden mit einer entweder zufälligen, pseudozufälligen oder "geheimen" vordefinierten Variation unter allen solchen γ_i.

[0061] Obwohl die Gleichungen (20)–(25) eine sich wiederholende Reihe der gleichen p Primitiven verwenden, können stattdessen auch verschiedene Reihen verwendet werden. Jede Reihe von Funktionen erzeugt eine separate Ausgabe-Hash-Adresse (output hash value) y, die dann miteinander verkettet werden können, um eine MAC oder eine einzelne Ausgabe y von jedem der Primitiven auszubilden, die durch Verwendung der Gleichung (26) summiert werden können, um den MAC-Wert zu ergeben. Des Weiteren könnte eine Reihe mit Vowärtsverkettung, wie beispielsweise durch die Gleichungen (20)–(23) angegeben, ausgeführt werden. Eine nächste Ausführung der gleichen Reihe, wie derjenigen, die z.B. durch die Gleichungen (24) und (25) angegeben wird, oder eine nächste Ausführung einer verschiedenen Reihe könnte mit "Rückwärts"-Verkettung ausgeführt werden. In Fällen, in denen eine Rückwärtsverkettung verwendet wird, könnten dazugehörige Eingabewerte in umgekehrter Reihenfolge in Bezug auf diejenigen, die mit Vorwärtsverkettung verwendet werden, auf die einzelnen Primitiven in dieser Reihe angewendet werden.

[0062] In Fällen, in denen unsere erfinderische Technik zum Berechnen einer Prüfsumme verwendet wird, sind die Berechnungen in hohem Maß denjenigen ähnlich, wenn nicht sogar identisch mit denjenigen, die zum Berechnen einer MAC verwendet werden, wobei aber alle Koeffizientenwerte sowie alle γ_i-Werte, wenn sie verwendet werden, öffentlich bekannt sind.

[0063] Unter Berücksichtigung des Vorgenannten wenden wir uns im Folgenden der Beschreibung der Software zu, die zum Generieren einer MAC für die Verwendung durch den Verschlüsselungsprozess **20** und in Übereinstimmung mit einem Primitiv zu, das unsere erfinderische Technik implementiert.

[0064] Fig. 4 stellt ein Ablaufdiagramm höchster Ebene des MAC-Generierungsprozesses **400** dar, der in dem in Fig. 1 gezeigten Prozess **5** zum Erzeugen einer MAC verwendet wird. Diese Routine implementiert die Gleichungen (20–(25), wie oben erläutert, unter der Annahme, dass die Primitiven F(x) und (G(x) vollständig ausgewählt worden sind, Insbesondere fährt die Ausführung nach der Eingabe in die Routine **410** während der Ausführung entweder des Verschlüsselungsprozesses **20** oder des Entschlüsselungsprozesses **50** zunächst, wie in Fig. 4 gezeigt, mit Block **410** fort. Dieser Block initialisiert einen Adressenverweis (i) auf eins und eine Summenvariable (y_s) auf Null. Danach gelangt die Ausführung in eine Schleife, die aus den Blöcken **420**, **430**, **440** und **450** gebildet wird, um aufeinander folgende Ausgabewerte y_i für jeden Eingabe-Klartext-Block (P_i), wie eingegeben, zu berechnen und diese Ausgabewerte in die Summenvariable y_s zu summieren.

[0065] Insbesondere fährt die Ausführung, nachdem sie in diese Schleife gelangt ist, zunächst mit Block **420** fort, um einen $F(P_i)$ entsprechenden Ausgabewert y_i zu berechnen. Sobald dies geschieht, fährt die Ausführung mit der Prozedur Summe berechnen **430** fort, die über den Block **435** einfach den Wert der Ausgabe y_i zu der Summenvariablen y_s addiert. Sobald dies geschieht, fährt die Ausführung mit dem Entscheidungsblock **440** fort, um zu bestimmen, ob alle N Blöcke der Eingabe-Klartext-Nachricht P verarbeitet worden sind, d.h. ob ein gegenwärtiger Wert des Adressenverweises dann gleich N ist. Für den Fall, dass noch solche Blöcke übrig sind, d.h. der gegenwärtige Wert von i kleiner als N ist, leitet der Entscheidungsblock **440** die Routine über den NEIN-Pfad **443** zum Block **450**. Dieser letztere Block inkrementiert den Wert des Adressenverweises i um eins und leitet die Ausführung dann über den Rückführpfad **455** zum Block **420** zurück, um den nächsten darauf folgenden Ausgabewert zu berechnen, und so weiter. Zu diesem Zeitpunkt hängen die von Block **420** durchgeführten Berechnungen davon ab, ob für irgendeine über den Block **420** vorgegebene Wiederholung der Wert von i gerade oder ungerade ist; daher der Wechsel zwischen den Primitiven F(x) und G(x) für aufeinander folgende i.

[0066] Sobald alle Ausgabewerte berechnet und summiert worden sind, leitet der Entscheidungsblock **440** die Ausführung über den JA-Pfad **447** zum Block **460**. Dieser letztere Block bildet die MAC, indem der Wert von y_N mit einem gegenwärtigen Wert der Summenvariablen einfach verkettet wird und als Ausgabe ein sich daraus ergebender 64-Bit-Wert als die MAC bereitgestellt wird. Sobald dies geschieht, verlässt die Ausführung die Routine **400**.

[0067] Fig. 5 stellt ein Ablaufdiagramm höchster Ebene der Prozedur Summe alternativ berechnen **500** dar, die statt der Prozedur Summe berechnen **430** verwendet werden kann, die einen Teil des MAC-Generierungsprozesses **400** bildet. Die Prozedur **500** implementiert die oben genannte Gleichung (26).

[0068] Insbesondere fährt die Ausführung nach der Eingabe in die Prozedur **500** zunächst mit dem Block **510**

fort, der einen Wert von γ_i entsprechend einrichtet. Wie oben erwähnt, kann dieser Wert in einem Bereich von einschließlich ± k zufällig, pseudo-zufällig oder vordefiniert sein, (obwohl typischerweise Werte von ± 1 verwendet werden). Sobald dieser Wert eingerichtet worden ist, fährt die Ausführung mit dem Block **520** fort, der den gegenwärtigen Ausgabewert γ_i mit dem entsprechenden Wert von γ_i multipliziert und einen sich daraus ergebenden Wert zu der Summenvariablen y_s addiert. Sobald dies geschieht, verlässt die Ausführung die Prozedur **500**.

[0069] Der Fachmann wird klar erkennen, dass, obwohl die MAC (oder Prüfsumme) als 64 Bits lang beschrieben worden ist, d.h. als zwei 32-Bit-Blöcke, MACs und Prüfsummen von anderen Bit- (und Block-) Größen, wie beispielsweise ein einzelner 32-Bit-Block oder mehr als 64 Bits lang (aber in ganzzahligen Blockgrößen) stattdessen verwendet werden können. Größere MACs stellen größere Sicherheitsebenen in dem gewährleisteten Ausmaß bereit, jedoch wahrscheinlich auf Kosten einer erhöhten Verarbeitungszeit für das Erzeugen der MAC und, falls erforderlich, ihr Verschlüsseln und Entschlüsseln.

[0070] Obwohl eine detaillierte Ausführungsform mit einer Reihe von Variationen, welche die Lehren der vorliegenden Erfindung integriert, beschrieben worden ist, ist der Fachmann problemlos in der Lage, viele andere Ausführungsformen und Anwendungen der vorliegenden Erfindung zu entwickeln, die ebenfalls diese Lehren verwenden.

Patentansprüche

1. Kryptografischer Prozess zur Verwendung in einer Vorrichtung (**100**) zum Verschlüsseln oder Entschlüsseln von jeweils einem Block von digitalem Eingabe-Klartext oder -Chiffretext in jeweils einen Block von digitalem Ausgabe-Chiffretext oder -Klartext, wobei der Prozess ein Primitiv F(x), das einer vordefinierten Funktion $f(x) = a\ x + b \bmod (M)$ entspricht, implementiert, wobei a und b vordefinierte ganze Zahlen sind und M eine vordefinierte ganzzahlige Primzahl ist, wobei die Vorrichtung (**100**) umfasst:
einen Prozessor (**340**); und
einen Speicher (**330**), der an den Prozessor (**340**) angeschlossen ist und ein Computerprogramm aufweist, das aus darin gespeicherten computerausführbaren Befehlen gebildet wird; und
der Prozess gekennzeichnet ist durch das Umfassen des Schritts, der von dem Prozessor durchgeführt und implementiert wird durch die ausführbaren Befehle des:
Umwandelns des digitalen Klartext- oder Chiffretext-Blocks in jeweils den digitalen Ausgabe-Chiffretext- oder Klartextblock durch ein vorgegebenes Verfahren, das als das Primitiv eine vordefinierte Abfolge von Reihenfolgemanipulationen, Additionen und mod(2^n)-Multiplikationsoperationen umfasst, wobei n eine vordefinierte ganze Zahl ist, die kollektiv das Primitiv ohne Durchführung von mod(M)-Berechnungen implementieren;
wobei das Primitiv F(x) in Übereinstimmung mit den folgenden Gleichungen implementiert wird:

$x^S \leftarrow$ Reihenfolgemanipulation (x)

$y \leftarrow A\ x + B\ x^s \bmod (2^n)$

$y^s \leftarrow$ Reihenfolgemanipulation (y)

$z \leftarrow C\ y^s + y\ D \bmod (2^n)$

$\theta \leftarrow z + y^s\ E \bmod (2^n)$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist; und
θ eine Ausgabe-Zeichenfolge ist.

2. Prozess nach Anspruch 1, wobei jede der Reihenfolgemanipulationen eine vordefinierte Operation ist, die eine Bit-Reihenfolge eines Blocks von Daten ändert, an dem jede Reihenfolgemanipulation durchgeführt wird.

3. Prozess nach Anspruch 2, wobei jede Reihenfolgemanipulation ein Byte- oder Wort-Swap oder Umkehroperation ist.

4. Prozess nach irgendeinem der Ansprüche 1 bis 3, wobei die digitalen Blöcke von Klartext und Chiffretext beide n Bits in der Länge sind.

5. Kryptografischer Prozess zur Verwendung in einer Vorrichtung (**100**) zum Verschlüsseln oder Entschlüsseln von jeweils einem Block von digitalem Eingabe-Klartext oder -Chiffretext in jeweils einen Block von digitalem Ausgabe-Chiffretext oder -Klartext, wobei der Prozess ein Primitiv F(x), das einer vordefinierten Funktion $f(x) = a\ x + b \bmod (M)$ entspricht, implementiert, wobei a und b vordefinierte ganze Zahlen sind und M eine vordefinierte ganzzahlige Primzahl ist, wobei die Vorrichtung (**100**) umfasst:
einen Prozessor (**340**); und
einen Speicher (**330**), der an den Prozessor (**340**) angeschlossen ist und ein Computerprogramm aufweist, das aus darin gespeicherten computerausführbaren Befehlen gebildet wird; und
der Prozess gekennzeichnet ist durch das Umfassen des Schritts, der von dem Prozessor durchgeführt und implementiert wird durch die ausführbaren Befehle des:
Umwandelns des digitalen Klartext- oder Chiffretext-Blocks in jeweils den digitalen Ausgabe-Chiffretext- oder Klartextblock durch ein vorgegebenes Verfahren, das als das Primitiv eine vordefinierte Abfolge von Reihenfolgemanipulationen, Additionen und $\bmod(2^n)$-Multiplikationsoperationen umfasst, wobei n eine vordefinierte ganze Zahl ist, die kollektiv das Primitiv ohne Durchführung von mod(M)-Berechnungen implementieren; wobei das Primitiv F(x) in Übereinstimmung mit den folgenden Gleichungen implementiert wird:

$y \leftarrow A\ x \bmod (2^n)$

$y^s \leftarrow$ Reihenfolgemanipulation (y)

$z \leftarrow B\ y^s \bmod (2^n)$

$z^s \leftarrow$ Reihenfolgemanipulation (z)

$v \leftarrow C\ z^s \bmod (2^n)$

$v^s \leftarrow$ Reihenfolgemanipulation (v)

$w \leftarrow D\ v^s \bmod (2^n)$

$w^s \leftarrow$ Reihenfolgemanipulation (w)

$t \leftarrow E\ w^s \bmod (2^n)$

$\theta \leftarrow t + L\ y^s \bmod (2^n)$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist;
L eine zufällige ganze Zahl ist, die kleiner oder gleich 2^n ist; und
θ eine Ausgabe-Zeichenfolge ist.

6. Prozess nach Anspruch 5, wobei die digitalen Blöcke von Klartext und Chiffretext beide n Bits in der Länge sind.

7. Computerlesbares Medium mit darauf gespeicherten computerausführbaren Befehlen zum Ausführen der Schritte von Anspruch 1 oder 5.

8. Vorrichtung (**100**), die zum Verschlüsseln oder Entschlüssel von jeweils einem digitalen Eingabe-Klartext oder -Chiffretext in jeweils einen Block von digitalem Ausgabe-Chiffretext oder Klartext ausgelegt ist, wobei die Vorrichtung (**100**) so ausgelegt ist, dass sie ein Primitiv implementiert, das einer vordefinierten Funktion $f(x) = a\ x + b \bmod (M)$ entspricht, wobei a und b vordefinierte ganze Zahlen sind und M eine vordefinierte ganzzahlige Primzahl ist, wobei die Vorrichtung (**100**) umfasst:
einen Prozessor (**340**); und
einen Speicher (**330**), der an den Prozessor angeschlossen ist und ein Computerprogramm aufweist, das aus darin gespeicherten computerausführbaren Befehlen gebildet wird; und
dadurch gekennzeichnet ist, dass der Prozessor in Reaktion auf die ausführbaren Befehle veranlasst, dass die Vorrichtung (**100**):
den digitalen Klartext- oder Chiffretext-Block in jeweils den digitalen Ausgabe-Chiffretext- oder Klartextblock durch ein vorgegehenes Verfahren, das als das Primitiv eine vordefinierte Abfolge von Reihenfolgemanipulationen, Additionen und $\bmod(2^n)$-Multiplikationsoperationen umfasst, wobei n eine vordefinierte ganze Zahl ist,

ohne Durchführung von mod(M)-Berechnungen umwandelt;
wobei das Primitiv F(x) in Übereinstimmung mit den folgenden Gleichungen implementiert wird:

$x^s \leftarrow$ Reihenfolgemanipulation (x)

$y \leftarrow A\,x + B\,x^s \bmod (2^n)$

$y^s \leftarrow$ Reihenfolgemanipulation (y)

$z \leftarrow C\,y^s + y\,D \bmod (2^n)$

$\theta \leftarrow z + y^s\,E \bmod (2^n)$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist; und
θ eine Ausgabe-Zeichenfolge ist.

9. Vorrichtung (**100**) nach Anspruch 8, wobei jede der Reihenfolgemanipulationen eine vordefinierte Operation ist, die eine Bit-Reihenfolge eines Blocks von Daten ändert, an dem jede Reihenfolgemanipulation durchgeführt wird.

10. Vorrichtung (**100**) nach Anspruch 9, wobei jede Reihenfolgemanipulation ein Byte- oder Wort-Swap oder Umkehroperationen ist.

11. Vorrichtung (**100**) nach irgendeinem der Ansprüche 8 bis 10, wobei die digitalen Blöcke von Klartext und Chiffretext beide n Bits in der Länge sind.

12. Vorrichtung (**100**), die zum Verschlüsseln oder Entschlüssel von jeweils einem digitalen Eingabe-Klartext oder -Chiffretext in jeweils einen Block von digitalem Ausgabe-Chiffretext oder Klartext ausgelegt ist, wobei die Vorrichtung (**100**) so ausgelegt ist, dass sie ein Primitiv implementiert, das einer vordefinierten Funktion $f(x) = a\,x + b \bmod (M)$ entspricht, wobei a und b vordefinierte ganze Zahlen sind und M eine vordefinierte ganzzahlige Primzahl ist, wobei die Vorrichtung (**100**) umfasst:
einen Prozessor (**340**); und
einen Speicher (**330**), der an den Prozessor angeschlossen ist und ein Computerprogramm aufweist, das aus darin gespeicherten computerausführbaren Befehlen gebildet wird;
dadurch gekennzeichnet ist, dass der Prozessor in Reaktion auf die ausführbaren Befehle veranlasst, dass die Vorrichtung (**100**):
den digitalen Klartext- oder Chiffretext-Block in jeweils den digitalen Ausgabe-Chiffretext- oder Klartextblock durch ein vorgegebenes Verfahren, das als das Primitiv eine vordefinierte Abfolge von Reihenfolgemanipulationen, Additionen und mod(2^n)-Multiplikationsoperationen umfasst, wobei n eine vordefinierte ganze Zahl ist, ohne Durchführung von mod(M)-Berechnungen umwandelt;
wobei das Primitiv F(x) in Übereinstimmung mit den folgenden Gleichungen implementiert wird:

$y \leftarrow A\,x \bmod (2^n)$

$y^s \leftarrow$ Reihenfolgemanipulation (y)

$z \leftarrow B\,y^s \bmod (2^n)$

$z^s \leftarrow$ Reihenfolgemanipulation (z)

$v \leftarrow C\,z^s \bmod (2^n)$

$v^s \leftarrow$ Reihenfolgemanipulafion (v)

$w \leftarrow D\,v^s \bmod (2^n)$

$w^s \leftarrow$ Reihenfolgemanipulation (w)

$t \leftarrow E\,w^s \bmod (2^n)$

$\theta \leftarrow t + L\, y^s \bmod (2^n)$

wobei: die Koeffizienten A, B, C, D und E jeweils eine ungerade zufällige ganze Zahl sind, die kleiner oder gleich 2^n ist;
L eine zufällige ganze Zahl ist, die kleiner oder gleich 2" ist; und
θ eine Ausgabe-Zeichenfolge ist.

13. Vorrichtung nach Anspruch 12, wobei die digitalen Blöcke von Klartext und Chiffretext beide n Bits in der Länge sind.

14. Vorrichtung (**100**) nach Anspruch 12, wobei jede der Reihenfolgemanipulationen eine vordefinierte Operation ist, die eine Bit-Reihenfolge eines Blocks von Daten ändert, an dem jede Reihenfolgemanipulation durchgeführt wird.

15. Vorrichtung (**100**) nach Anspruch 12, wobei jede Reihenfolgemanipulation ein Byte- oder Wort-Swat oder Umkehroperationen sind.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

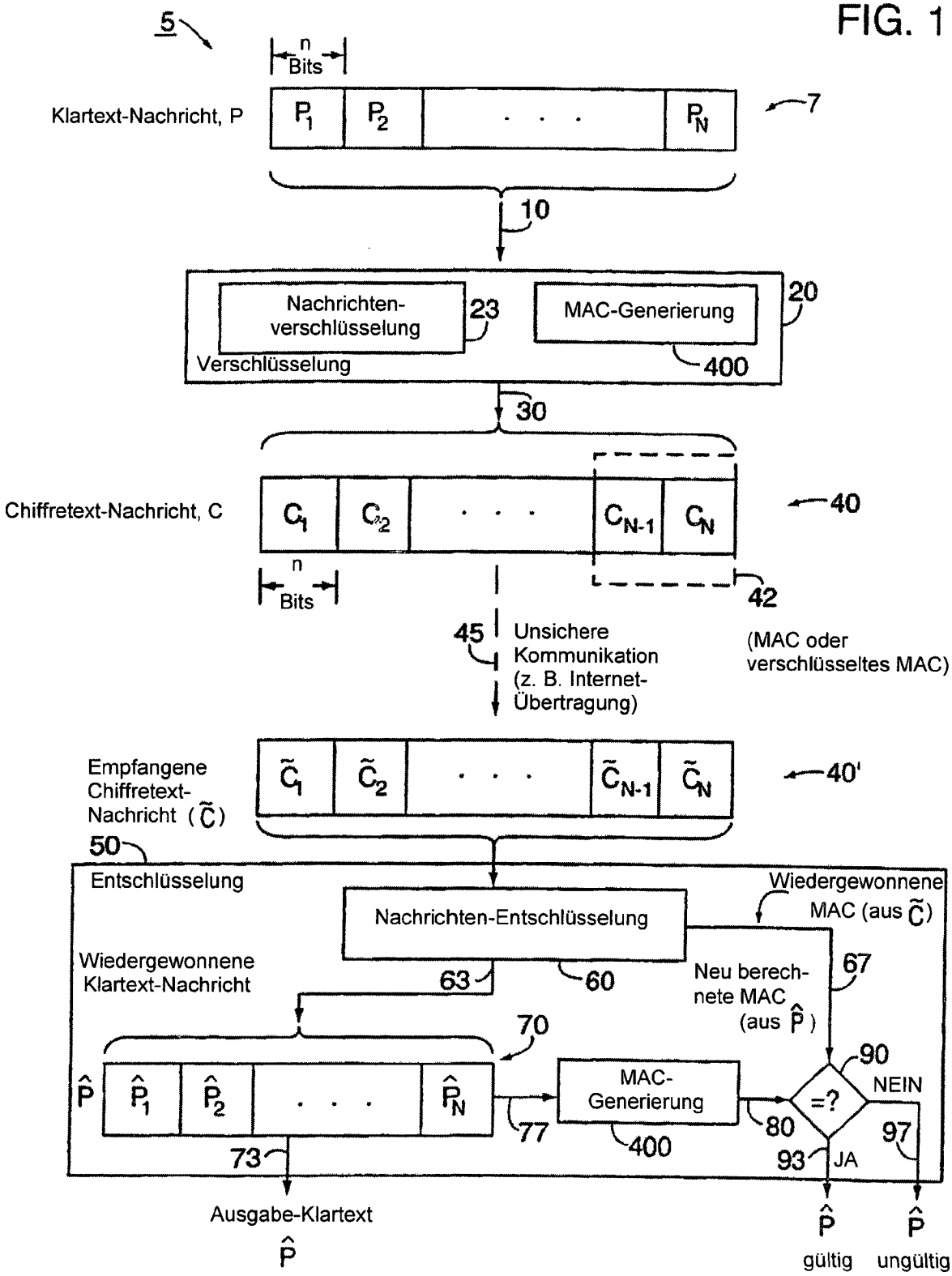
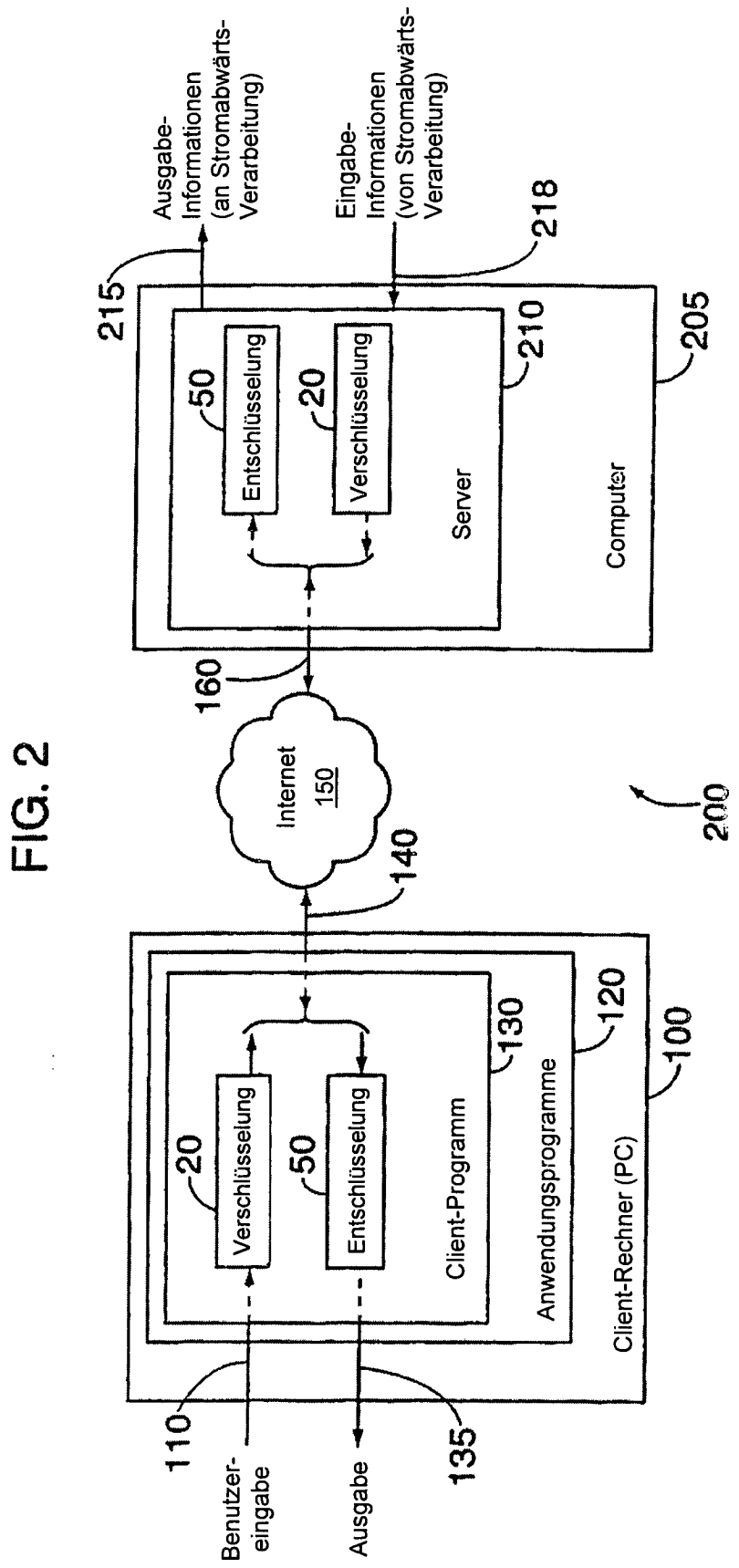


FIG. 2


Ausgabe-Informationen (an Stromabwärts-Verarbeitung)
Eingabe-Informationen (von Stromabwärts-Verarbeitung)
215
218
Entschlüsselung
50
Verschlüsselung
20
Server
210
Computer
205
160
Internet
150
140
200
Verschlüsselung
20
Entschlüsselung
50
Client-Programm
130
Anwendungsprogramme
120
Client-Rechner (PC)
100
110
135
Benutzer-eingabe
Ausgabe

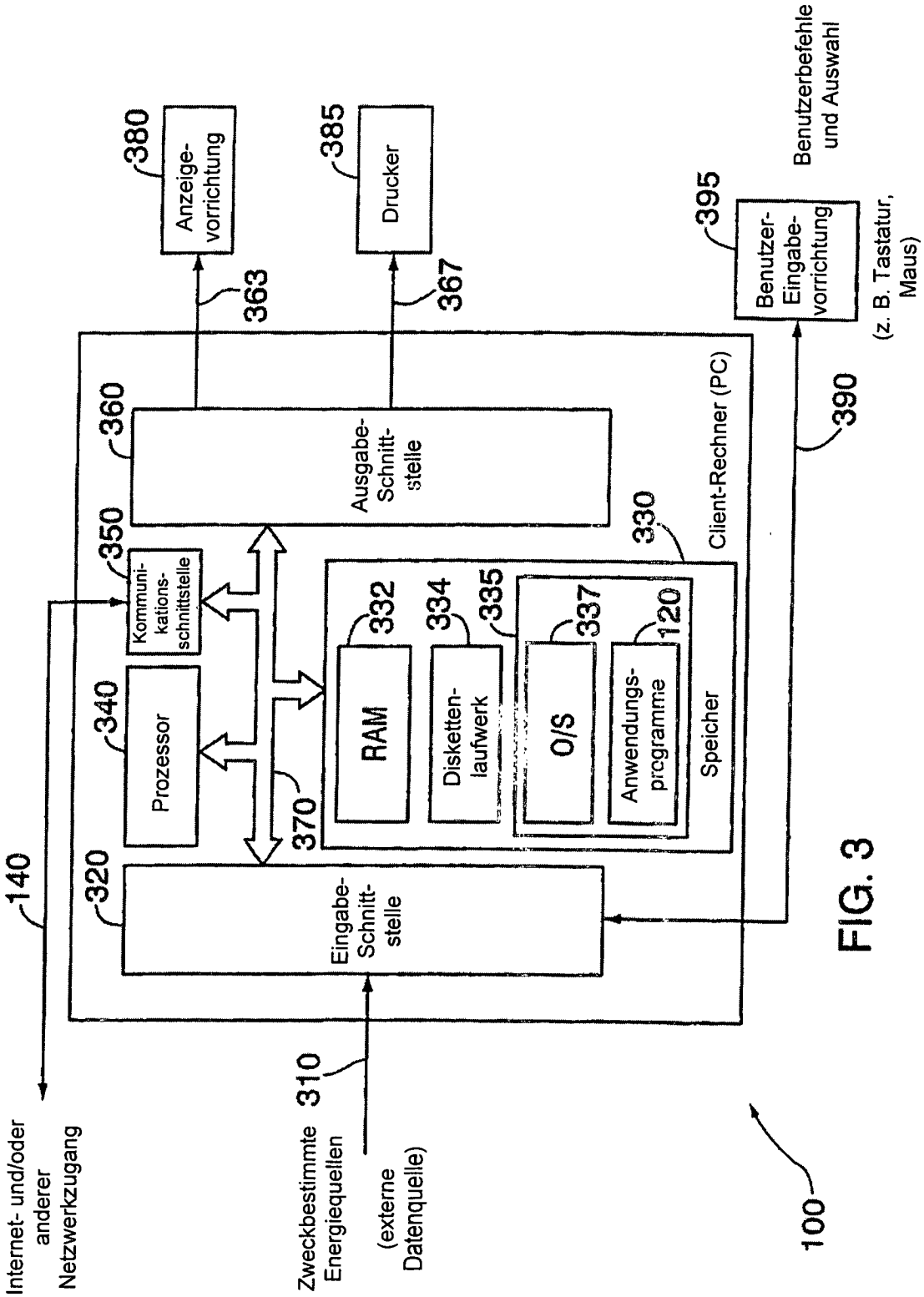
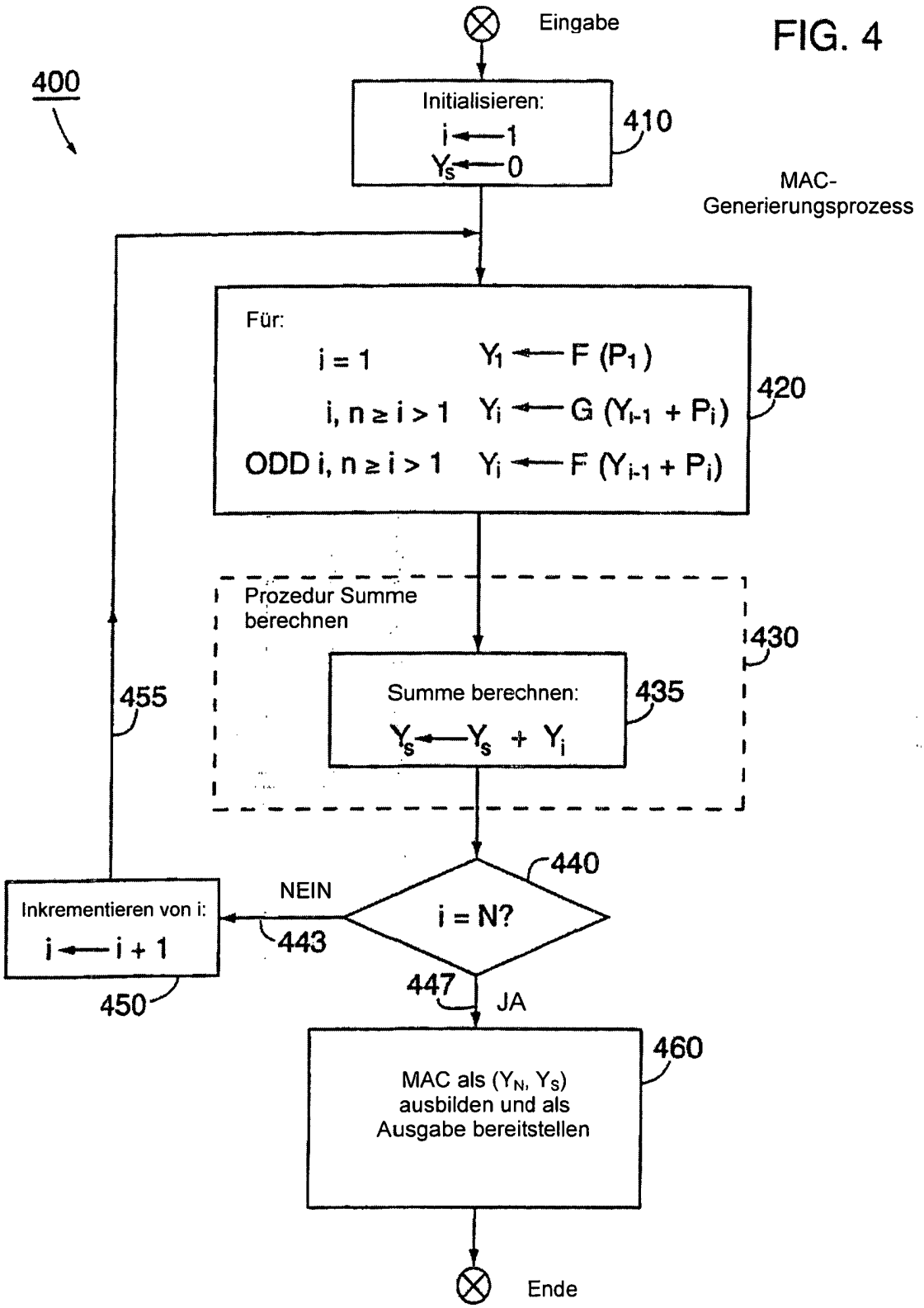

Internet- und/oder anderer Netzwerkzugang
140
Zweckbestimmte Energiequellen
(externe Datenquelle)
310
100
Eingabe-Schnitt-stelle
320
Prozessor
340
Kommuni-kations-schnittstelle
350
370
RAM
332
Disketten-laufwerk
334
O/S
335
337
Anwendungs-programme
120
Speicher
330
Client-Rechner (PC)
Ausgabe-Schnitt-stelle
360
Anzeige-vorrichtung
380
Drucker
385
363
367
Benutzer-Eingabe-vorrichtung
395
(z. B. Tastatur, Maus)
Benutzerbefehle und Auswahl
390

FIG. 3

FIG. 4


400
Eingabe
Initialisieren:
i ← 1
Ys ← 0
410
MAC-Generierungsprozess
Für:
i = 1 Y1 ← F (P1)
i, n ≥ i > 1 Yi ← G (Yi-1 + Pi)
ODD i, n ≥ i > 1 Yi ← F (Yi-1 + Pi)
420
Prozedur Summe berechnen
430
Summe berechnen:
Ys ← Ys + Yi
435
455
440
i = N?
NEIN
443
Inkrementieren von i:
i ← i + 1
450
447
JA
460
MAC als (YN, YS) ausbilden und als Ausgabe bereitstellen
Ende

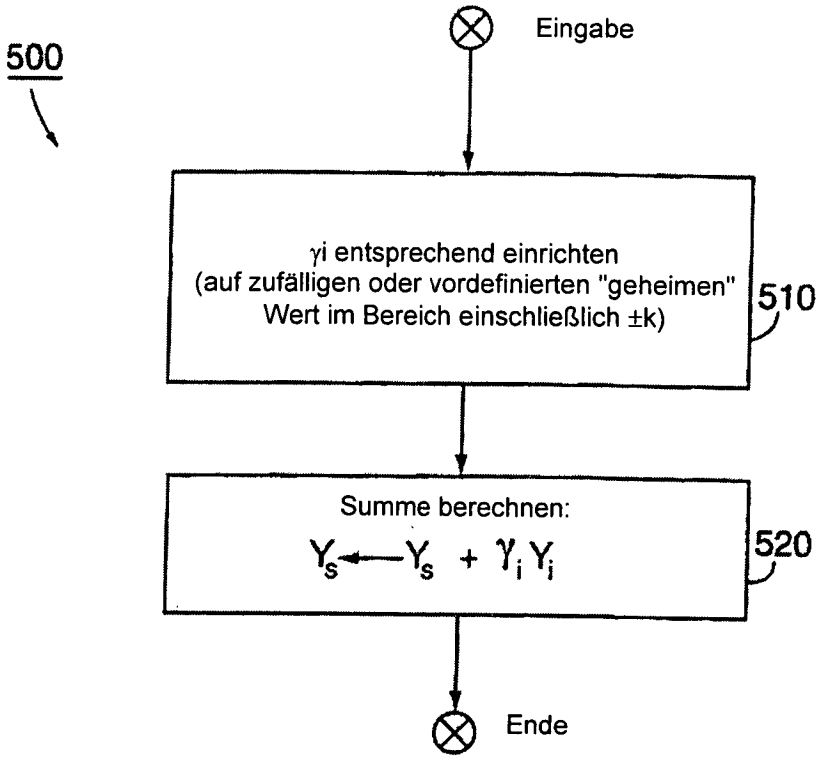


FIG. 5

Prozedur Summe alternativ berechnen

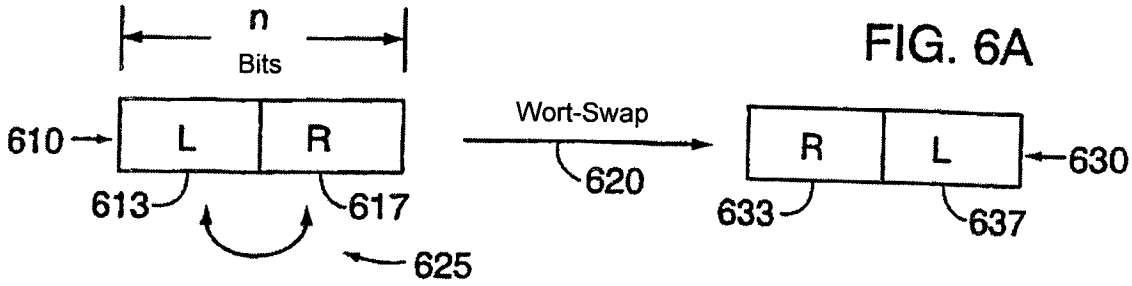


FIG. 6A

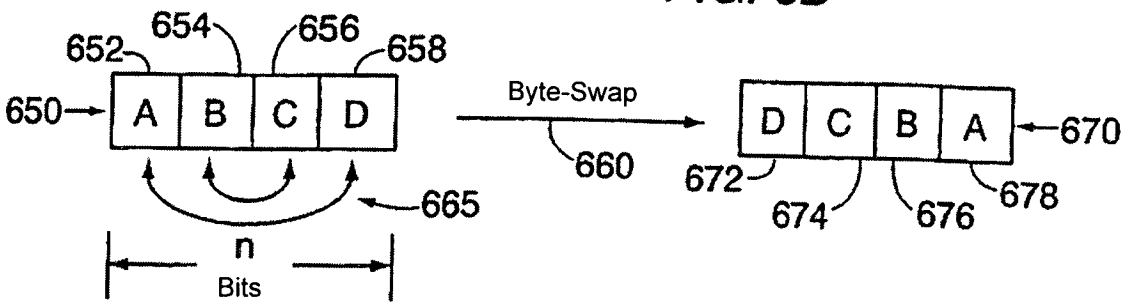


FIG. 6B