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

(10) **DE 601 03 737 T2** 2005.07.07

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 257 084 B1**
(21) Deutsches Aktenzeichen: **601 03 737.5**
(96) Europäisches Aktenzeichen: **01 309 740.7**
(96) Europäischer Anmeldetag: **19.11.2001**
(97) Erstveröffentlichung durch das EPA: **13.11.2002**
(97) Veröffentlichungstag
der Patenterteilung beim EPA: **09.06.2004**
(47) Veröffentlichungstag im Patentblatt: **07.07.2005**

(51) Int Cl.[7]: **H04L 9/32**

(30) Unionspriorität:
854251 **11.05.2001** **US**

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

(73) Patentinhaber:
Lucent Technologies Inc., Murray Hill, N.J., US

(72) Erfinder:
Patel, Sarvar, Montville, New Jersey 07045, US

(74) Vertreter:
derzeit kein Vertreter bestellt

(54) Bezeichnung: **System und Verfahren zur Nachrichtenauthentisierung**

Beschreibung

1. Erfindungsgebiet

[0001] Die vorliegende Erfindung bezieht sich auf Kommunikationen und insbesondere auf die Authentifizierung von Nachrichten.

2. Beschreibung des Standes der Technik

[0002] Fig. 1 zeigt eine schematische Darstellung erster und zweiter drahtloser Kommunikationssysteme, die drahtlose Kommunikationsdienste für drahtlose Einheiten (z.B. drahtlose Einheiten **12a–c**) bereitstellen, die innerhalb der geografischen Regionen **14** bzw. **16** liegen. Eine Mobilfunkvermittlungsstelle (z.B. MSCs **20** und **24**) ist u.a. verantwortlich für die Herstellung und Aufrechterhaltung von Gesprächen zwischen drahtlosen Einheiten, Gesprächen zwischen einer drahtlosen Einheit und einer Festnetzeinheit (z.B. Festnetzeinheit **25**), und/oder Verbindungen zwischen einer drahtlosen Einheit und einem PDN (Packet Data Network) wie z.B. dem Internet. In dieser Eigenschaft schaltet die MSC die drahtlosen Einheiten innerhalb ihrer geografischen Region mit einem PSTN (öffentliches Fernsprechnetz) **28** und/oder einem PDN **29** zusammen. Der von der MSC abgedeckte geografische Bereich ist in räumlich getrennte Bereiche, sog. „Zellen", unterteilt. Wie in Fig. 1 dargestellt, ist jede Zelle schematisch durch ein Hexagon im Honigwabenmuster dargestellt; in der Praxis hat jede Zelle jedoch eine unregelmäßige Form, die sich nach der Topographie des die Zelle umgebenden Terrains richtet.

[0003] Typisch enthält jede Zelle eine Basisstation (z.B. Basisstationen **22a–e** und **26a–e**), die Radios und Antennen aufweist, die von der Basisstation zum Kommunizieren mit den drahtlosen Einheiten in dieser Zelle benutzt werden. Die Basisstationen weisen ferner die Übertragungsausrüstung auf, die von der Basisstation zum Kommunizieren mit der MSC in dem geografischen Bereich benutzt wird. Beispielsweise ist MSC **20** mit den Basisstationen **22a–e** im geografischen Bereich **14** verbunden, und eine MSC **24** ist mit den Basisstationen **26a–e** in der geografischen Region **16** verbunden. Innerhalb einer geografischen Region vermittelt die MSC Gespräche zwischen den Basisstationen in Echtzeit, während die drahtlose Einheit von einer zu einer anderen Zelle geht, was als Hand-Over bezeichnet wird. Je nach Ausführungsform kann ein Basisstationscontroller (BSC) ein separater, an mehrere Basisstationen angeschlossener oder in jeder Basisstation angeordneter (nicht dargestellter) Basisstationscontroller (BSC) sein, der die Radioressourcen für die Basisstationen verwaltet und Informationen an die MSC zurücksendet.

[0004] Die MSCs **20** und **24** benutzen ein Signalisierungsnetzwerk **32** wie z.B. ein Signalisierungsnetzwerk, welches dem Standard TIA/EIA-41-D, betitelt „Cellular Radiotelecommunications Intersystem Operations", Dezember 1997, („IS-41") entspricht, der den Austausch von Information über die drahtlosen Einheiten ermöglicht, die zwischen den betreffenden geografischen Bereichen **14** und **16** vagabundieren. Zum Beispiel vagabundiert eine drahtlose Einheit **12a**, wenn sie den geografischen Bereich **14** der MSC **20** (wie Home MSC) verlässt, der sie ursprünglich zugeordnet war. Um sicherzustellen, dass eine vagabundierende drahtlose Einheit ein Gespräch empfangen kann, meldet sich die vagabundierende drahtlose Einheit **12a** bei der MSC **24** an, in der sie sich gerade aufhält (d.h. der Besucher-MSC), indem sie die Besucher-MSC **24** von ihrer Gegenwart benachrichtigt. Sobald eine vagabundierende drahtlose Einheit **12a** von einer Besucher-MSC **24** erkannt wird, sendet die Besucher-MSC **24** eine Anmeldeaufforderung über das Signalisierungsnetzwerk **32** an die Home-MSC **20**, und die Home MSC **20** schreibt die Kennung der Besucher-MSC **24** in eine Datenbank **34**, das sog. Heimatstandortregister HLR, wodurch die Home-MSC **20** weiß, an welchem Standort sich die vagabundierende drahtlose Einheit **12a** aufhält. Nach Authentifizierung der vagabundierenden drahtlosen Einheit schickt die Home-MSC **20** der Besucher-MSC **24** ein Kundenprofil. Nach Erhalt des Kundenprofils aktualisiert die Besucher-MSC **24** eine Datenbank **36**, das sog. Besucherstandortregister (VLR), um die gleichen Merkmale wie die Home-MSC **20** bereitzustellen. Das HLR, das VLR und/oder das Authentifizierungszentrum (AC) können ebenfalls in der MSC angeordnet oder für entfernten Zugriff eingerichtet sein.

[0005] Wenn im UMTS (Universal Mobile Telecommunications System) oder 3G IS-41 eine drahtlose Einheit ein Gespräch initiiert oder empfängt, wird sie identifiziert, bevor sie wählen kann. Nach der Authentifizierung wird ein 128-Bit-Integritätsschlüssel (IK), der mit Hilfe eines Geheimschlüssels erzeugt wurde, aktiviert und kann zum Prüfen der Integrität einer Nachricht, die zwischen der drahtlosen Einheit und der System- oder Nachrichtenauthentifizierung gesendet wurde, verwendet werden.

[0006] Der Entwurf guter Nachrichtenauthentifizierungssysteme ist einer der wichtigsten Bereiche in der Kryptographie. Das Ziel von Nachrichtenauthentifizierungssystemen besteht darin, Nachrichten effizient derart

von einer zu einer anderen Partei zu übertragen, dass die empfangende Partei ermitteln kann, ob die von ihr erhaltene Nachricht in unerlaubter Weise geändert wurde. Fig. 2 zeigt, wie bei einer drahtlosen Einheit in einem drahtlosen Kommunikationssystem eine Nachrichtenauthentifizierung durchgeführt wird. An der Einstellung sind zwei Parteien – die drahtlose Einheit und das drahtlose Kommunikationssystem – beteiligt, die sich auf einen Geheimschlüssel k geeinigt haben. Es werden zwei Algorithmen benutzt: ein Bereitschaftsalgorithmus S_k und ein Prüfungsalgorithmus V_k. Wenn die drahtlose Einheit eine Nachricht M an das drahtlose Kommunikationssystem schicken will, berechnet sie unter Einsatz des MAC-Generators **50** als erstes ein Kennzeichen (TAG) oder einen Nachrichtenauthentifizierungscode (MAC), $\mu=S_k(M)$. Die Einheit sendet die Nachricht und das Tag-Paar (M,μ) an das drahtlose Kommunikationssystem, und nach Empfang des Paares (M,μ) berechnet das drahtlose Kommunikationssystem $V_k(M,\mu)$, welches **1** zurückgibt, wenn der MAC gültig ist, oder 0, wenn nicht. Aus Fig. 2 ist ersichtlich, dass das drahtlose Kommunikationssystem die Nachricht und das k in den MAC-Generator **52** eingibt, der ein Tag' erzeugt, und es wird ein Vergleich **54** zwischen dem von der drahtlosen Einheit empfangenen Tag (μ) und dem im System erzeugten Tag' vorgenommen. Wenn beide gleich sind, wird die Nachricht als gültig akzeptiert; andernfalls wird die Nachricht abgelehnt. Ohne den Geheimschlüssel k zu kennen, ist es praktisch unmöglich für einen Gegner, eine Nachricht und einen entsprechenden MAC zu konstruieren, die der Prüfungsalgorithmus als gültig akzeptiert.

[0007] Das gleiche Nachrichtenauthentifizierungssystem kommt bei der Übertragung von Nachrichten vom drahtlosen Kommunikationssystem zur drahtlosen Einheit zum Einsatz. Zum Beispiel zeigt Fig. 3, wie das drahtlose Kommunikationssystem durch Generieren eines Tags mit einem MAC-Generator **56** eine geschützte Nachricht an eine drahtlose Einheit sendet, wobei als Eingaben die Nachricht und ein Geheimschlüssel k verwendet werden. Das drahtlose Kommunikationssystem sendet eine Nachricht zusammen mit dem Tag an eine drahtlose Einheit, die die Nachricht und den Geheimschlüssel k in einen MAC-Generator **58** eingibt, um ein Tag' zu generieren. Die drahtlose Einheit stellt einen Vergleich **60** zwischen Tag' und dem vom drahtlosen Kommunikationssystem empfangenen Tag an. Wenn die Tags übereinstimmen, wird die Nachricht als gültig akzeptiert. Wenn nicht, wird die Nachricht als geändert oder ungültig abgelehnt.

[0008] Das Sicherheitserfordernis für einen Nachrichtenauthentifizierungscode lässt sich wie folgt erklären: Ein Gegner fälscht einen MAC, wenn er, vorausgesetzt, dass er den MAC S_k, V_k für bestimmte Nachrichten, bei denen k geheim bleibt, abfragen kann, ein gültiges Paar (M^*,μ^*) finden kann, so dass sich $V_k(M^*\mu^*)$ zu 1 ergibt, wobei die Nachricht M* jedoch nie als Eingabe für S_x verwendet wurde.

[0009] Bei einem in der Praxis allgemein benutzten Ansatz für Nachrichtenauthentifizierungssysteme kommen kryptographische Hash-Funktionen zum Einsatz. Eine Hash-Funktion kann typisch als Funktion beschrieben werden, welche Eingaben einer bestimmten Länge auf Ausgaben einer kürzeren Länge abbildet. Ferner ist es schwierig, zwei Eingaben zu finden, die sich auf die gleiche Ausgabe abbilden lassen. Diese auf kryptographischen Hash-Funktionen basierten MAC-Systeme sind gut, weil sie schnelle und sichere kryptographische Bausteine benutzen. Kryptographische Hash-Funktionen F(x) sind typisch öffentliche, schlüssellose und kollisionsresistente Funktionen, die beliebig lange Eingaben x auf kurze Ausgaben abbilden. Kollisionsresistenz bedeutet, dass es von der Berechnung her unmöglich sein müsste, zwei Nachrichten x_1 und x_2 zu finden, die sich zu $F(x_1)=F(x_2)$ ergeben. MD5, SHA-1 und RIPE-MD sind weit verbreitet benutzte kryptographische Hash-Funktionen. Neben der Kollisionsresistenz werden Hash-Funktionen gewöhnlich noch mit anderen Merkmalen versehen, sowohl um die Funktion für andere Zwecke einsetzen zu können als auch um die Wahrscheinlichkeit der Kollisionsresistenz zu erhöhen.

[0010] Die meisten kryptographischen Hash-Funktionen wie MD5 und SHA-1 benutzen eine iterierte Konstruktion, in der die Eingangsnachricht Block für Block verarbeitet wird. Wie in Fig. 4 dargestellt, wird der grundlegende Block die Komprimierungsfunktion f genannt, welche eine Hash-Funktion ist, die zwei Eingaben der Größe t und b nimmt und sie auf eine kürzere Ausgabe der Länge t abbildet. In MD5 hat die Eingabe der Größe t eine Länge von 128 Bits und die Eingabe der Größe b eine Länge von 512 Bits. In SHA-1 hat die Eingabe der Größe t eine Länge von 160 Bits und die Eingabe der Größe b eine Länge von 512 Bits. Die Eingabe der Größe t wird die Verkettungsvariable genannt und die Eingabe der Größe b oder Nutzinformation oder Block dient zur eigentlichen Verarbeitung der Nachricht x in jeweils b Bits. Wie in Fig. 5 dargestellt, wird also die Hash-Funktion F(x) dadurch geformt, dass die Komprimierungsfunktion f über die Nachricht m iteriert wird, wobei h_i als Verkettungsvariable und x_i als Nutzinformation, wie in den folgenden Schritten angegeben, verwendet werden:

1. Benutzen Sie ein geeignetes Verfahren zum Auffüllen der Nachrichtenlänge, und zwar so, dass für die Eingabe ein Mehrfaches der Blockgröße b erhalten wird. Die Eingabe kann in Blockgrößenstücke $x=x_1....x_n$ aufgeteilt werden.
2. h_o = IV, eine feste Konstante

3. Für i = 1 bis n
4. hi = f (hi-1,xi)
5. Ausgabe h_n als F (x)

[0011] Zum Beispiel hat, wenn eine SHA-1-Hash-Funktion verwendet wird, jeder Aufruf an die SHA-1 Hash-Funktion einen 160-Bit-Anfangsvektor (IV), wobei eine 512-Bit-Eingabe oder -Nutzinformation verwendet wird, die auf eine 160-Bit-Ausgabe abgebildet wird. Der IV wird auf den IV eingestellt, der im Standard für die SHA-1-Hash-Funktion (National Institute of Standards and Technology, NIST FIPS PUB 180 „Secure Hash Standard", U.S. Department of Commerce, Mai 1993) definiert ist.

[0012] Kryptographische Hash-Funktionen werden ohne Schlüssel konzipiert. Da jedoch die Nachrichtenauthentifizierung den Einsatz eines Geheimschlüssels erfordert, wird ein Verfahren benötigt, durch welches die Hash-Funktion mit einem Schlüssel versehen wird. Eine Möglichkeit zum Erstellen eines Schlüssels für die Hash-Funktion besteht darin, einen Geheimschlüssel, anstatt des festen und bekannten IV, zu benutzen. Wie in **Fig. 6** dargestellt, wird die Verkettungsvariable in der Komprimierungsfunktion f (Verkettungsvariable,x1) durch den Schlüssel k ersetzt, um $f_k(x1) = f(k,x1)$ zu bilden, worin x1 die Blockgröße b hat. Die iterierte Hash-Funktion F(IV,x) wird modifiziert, indem der feste IV durch den Geheimschlüssel k ersetzt wird, um $F_k(x)= F(k,x)$ zu bilden. Die Kollisionsresistenz für eine Schlüssel-Funktion ist anders als bei schlüssellosen Funktionen, weil der Gegner Fk(x) zu keinem Zeitpunkt ohne Abfragen des Benutzers auswerten kann. Dieses Erfordernis ist schwächer als das Standardkollisionserfordernis, deshalb wollen wir die Funktion Fk(x) als schwach kollisionsresistent bezeichnen.

[0013] Eine allgemeine Beschreibung der iterierten Hash-Funktionen und Schlüssel-Hash-Funktionen ist im HANDBOOK OF APPLIED CRYPTOGRAPHY, 1996, von Menezes, Oorschot, Vanstone, CRC Press LLC, USA XP002198439, zu finden.

[0014] Um die Sicherheit der kryptographischen Schlüssel-Hash-Funktion zu verbessern, wurde eine vernestete MAC-Funktion (NMAC) entwickelt, die als

$$NMAC_k(x) = F_{k1}(F_{k2}(x))$$

definiert wird, wobei die kryptographische Hash-Funktion F zuerst mit dem Geheimschlüssel k_2 anstatt mit IV versehen wird, und die Nachricht x iterativ auf die Ausgabe von F_{k2} (x) hash-codiert wird. M. Bellare, R. Canetti und H. Krawczyk, Keying Hash Functions for Message Authentication, Advances in Cryptology – Crypto '96 16th Annual International Cryptology Conference. Die vernesteten Hash-Konstruktionen NMAC und HMAC sind in Proceedings of the Annual International Cryptology Conference (Crypto), Berlin, Springer, DE (18-08-1196), CONF 16,1–15, Santa Barbara, Aug. 18–22, 1996, beschrieben. Die Ausgabe $F_{k2}(x)$ wird auf eine Blockgröße gemäß dem Füllzeichensystem von F aufgefüllt, und dann wird das Ergebnis von $F_{k2}(x)$ mit dem Geheimschlüssel k_1 versehen und mit einer äußeren Hash-Funktion F, wie in **Fig. 7** dargestellt, hash-codiert. Somit hat der NMAC-Schlüssel k zwei Teile $k = (k_1, k_2)$. Der folgende Lehrsatz über die Sicherheit von NMAC, bezogen auf die Sicherheit der zugrunde liegenden krpytographischen Hash-Funktion, ist in M. Bellare, R. Canetti und H. Krawczyk, Keying Hash Functions for Message Authentication, wie oben angegeben, bewiesen.

[0015] Lehrsatz 1: In t Schritten und q Abfragen, wenn die Schlüssel-Komprimierungsfunktion f ein $\in_f$ sicherer MAC und die iterierte Schlüssel-Hash-Funktion F $\in_F$ schwach kollisionsresistent ist, dann ist die NMAC-Funktion ein $(\in_f + \in_F)$ sicherer MAC.

[0016] Die NMAC-Konstruktion ruft die Komprimierungsfunktion mindestens zweimal auf; der innere Aufruf an $F_{k2}(x)$ kostet das Gleiche wie die schlüssellose Hash-Funktion F(x). Somit ist der äußere Aufruf an F_{k1} ein zusätzlicher Aufruf, der über die Erfordernisse für die schlüssellose Hash-Funktion hinausgeht. Der äußere Funktionsaufruf ist im Grunde genommen ein Aufruf an die Schlüssel-Komprimierungsfunktion f_{k1}, da die Ausgabe der Größe t von $F_{k2}(x)$ in die Eingabe der Größe b für die Komprimierungsfunktion passt. Für große, aus mehreren Blöcken bestehende x sind die Kosten des zusätzlichen äußeren Komprimierungsaufrufs nicht signifikant. Für kleine Nachrichten x jedoch kann die zusätzliche äußere Komprimierungsfunktion, in Prozenten ausgedrückt, im Vergleich zu der schlüssellosen Hash-Funktion zu einer signifikant hohen Ineffizienz führen. In Tabelle 1 ist die Ineffizienz für kleine x für die SHA-1-Hash-Funktion dargestellt. Die Anzahl der von der zugrunde liegenden Hash-Funktion und dem NMAC benötigten Komprimierungsaufrufe werden für verschiedene kleine x, die um jeweils 30 Byte zunehmen, miteinander verglichen. Die Ineffizienz von NMAC mit Bezug auf die zugrunde liegende Hash-Funktion ist ebenfalls in der Tabelle angegeben.

x in Stufen von 240 Bits	Anzahl der f in F(x)	Anzahl der f in NMAC	Proz. Ineffizienz
240	1	2	100%
480	2	3	50%
720	2	3	50%
960	3	4	33%
1200	3	4	33%
1440	3	4	33%
1680	4	5	25%
1920	4	5	25%
2160	5	6	20%
2400	5	6	20%

Tabelle 1: Vergleich zwischen der Anzahl der Komprimierungsaufrufe für kurze Nachrichten unterschiedlicher Größen.

[0017] Wie ersichtlich ist, kann der Mehraufwand für kurze Nachrichten groß sein. Beispielsweise ist der Mehraufwand für Nachrichten, die in einen Block passen, 100 %, weil NMAC zwei Komprimierungsfunktionsaufrufe benötigt, während bei der zugrunde liegenden kryptographischen Hash-Funktion ein Komprimierungsaufruf ausreicht.

[0018] HMAC ist eine praktische Variante von NMAC für diejenigen Implementierungen, die keinen Zugriff auf die Komprimierungsfunktion f haben, sondern nur die kryptographische Hash-Funktion F mit der Nachricht aufrufen können. Für diese Implementierungen kann der Schlüssel nicht in die Verkettungsvariable gesetzt werden, und die Funktion F wird mit dem festen und bekannten IV aufgerufen, der in der anfänglichen Komprimierungsfunktion benutzt wurde. Die HMAC-Funktion wird definiert als

$HMAC_k(x) = F(k \oplus opad, F(k \oplus ipad, x))$

worin ein Schlüssel k benutzt wird und k die Auffüllung von k mit Nullen darstellt, um den Größe-b-Block der iterierten Hash-Funktion fertig zu stellen. Der Wert k wird bitweise ausschließlich mit opad geORt, und das Ergebnis wird zu der Nachricht x verkettet. Die Hash-Funktion F wird mit der gesamten verketteten Nachricht aufgerufen. Wie in Fig. 8 dargestellt, wird nach der ersten Iteration der Komprimierungsfunktion f der Schlüssel k2 als k2 = f(k ⊕ipad) produziert. Nachdem die Hash-Funktion F fertiggestellt ist, wird der resultierende Wert F(k ⊕ipad,X) produziert. Die Hash-Funktion F wird nochmals mit einer Nachricht aufgerufen, die den Wert von k ⊕opad – eine bitweise ausschließliche Operation mit k und opad – aufweist. Nach der ersten Iteration innerhalb des zweiten Aufrufs der Hash-Funktion F wird der Schlüssel k1 aus der Komprimierungsfunktion f (IV, k ⊕opad) erhalten. Die Werte ipad und opad sind feste Konstanten, wie in M. Bellare, R. Canetti und H. Krawczyk, Keying Hash Functions for Message Authentication, In Proc. CRYPTO 96, Lecture Notes in Computer Science, Springer-Verlag 1996 beschrieben. Die zweite Iteration innerhalb des zweiten Aufrufs der Hash-Funktion benutzt die Komprimierungsfunktion f(k1,F(k ⊕ipad,X)), um die HMAC-Funktion F (k ⊕opad, F (k ⊕ipad,x)) zu erzeugen.

[0019] Wenn für $k_1 = f(k \oplus opad)$ und $k_2 = f(k \oplus ipad)$ definiert wird, ergibt sich $HMAC_k(x)$ zu $NMAC_{(k1,k2)}(x)$. HMAC ist der Internet-Standard für Nachrichtenauthentifizierung. Wie dargestellt, bezieht sich der Sicherheitsbeweis von HMAC auf NMAC und nimmt an, dass die zugrunde liegende kryptographische Hash-Funktion (schwach) kollisionsresistent und die zugrunde liegende Komprimierungsfunktion ein sicherer MAC ist, wenn beide mit einem entsprechenden Schlüssel versehen sind. HMAC ist effizient für lange Nachrichten, bei kurzen Nachrichten führt die vernestete Konstruktion jedoch zu einer signifikanten Ineffizienz. Wenn zum Beispiel eine Nachricht, die kürzer als ein Block ist, mit MAC kodiert werden soll, ohne Zugriff auf die Komprimierungsfunktion zu haben, benötigt HMAC vier Aufrufe für die Komprimierungsfunktion. In Fällen, wo Zugriff auf die Komprimierungsfunktion erlaubt ist, können k1 und k2 vorweg berechnet und in die Verkettungsvariable der Kom-

primierungsfunktion eingefügt werden, so dass zwei Aufrufe an die Komprimierungsfunktion erforderlich sind. Diese Ineffizienz kann für manche Anwendungen, wie Nachrichtenauthentifizierung von Signalisierungsnachrichten, wo die einzelnen Nachrichten alle in einen oder zwei Blöcke passen, besonders hoch sein. Auch im TCP/IP-Verkehr ist bekannt, dass eine große Anzahl von Paketen (z.B. Bestätigung) eine Größe von ungefähr 40 Bytes haben, die in einen Block der meisten kryptographischen Hash-Funktionen passen. Wir schlagen eine Verbesserung vor, die es gestattet, sowohl kurze als auch lange Nachrichten effizienter als HMAC zu authentifizieren, und außerdem auch Sicherheitsbeweise bereitstellt.

KURZDARSTELLUNG DER ERFINDUNG

[0020] Ein erfindungsgemäßes Verfahren und System sind in den unabhängigen Ansprüchen definiert. Bevorzugte Ausführungsformen sind in den abhängigen Ansprüchen definiert.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0021] Andere Aspekte und Vorteile der vorliegenden Erfindung ergeben sich aus dem Lesen der folgenden ausführlichen Beschreibung unter Bezugnahme auf die Zeichnungen, in denen:

[0022] Fig. 1 ein allgemeines Diagramm eines drahtlosen Kommunikationssystems veranschaulicht, für welches das MAC-Erzeugungssystem gemäß den Prinzipien der vorliegenden Erfindung verwendet werden kann;

[0023] Fig. 2 ein allgemeines Diagramm ist, welches darstellt, wie ein MAC zur Authentifizierung von Nachrichten verwendet wird, die von einer drahtlosen Einheit an ein drahtloses Kommunikationssystem gesendet werden;

[0024] Fig. 3 ein allgemeines Diagramm ist, welches darstellt, wie ein MAC zur Authentifizierung von Nachrichten verwendet wird, die von einem drahtlosen Kommunikationssystem an eine drahtlose Einheit gesendet werden;

[0025] Fig. 4 ein Blockdiagramm einer Komprimierungsfunktion f ist;

[0026] Fig. 5 ein Blockdiagramm ist, welches die iterierte Konstruktion einer Hash-Funktion F für eine gegebene Komprimierungsfunktion f darstellt;

[0027] Fig. 6 ein Blockdiagramm ist, welches eine Schlüssel-Hash-Funktion darstellt;

[0028] Fig. 7 ein Blockdiagramm ist, welches eine vernestete Hash-Funktion (NMAC) darstellt;

[0029] Fig. 8 ein Blockdiagramm ist, welches eine Variante einer NMAC-Funktion darstellt, die als HMAC bekannt ist;

[0030] Fig. 9 ein Blockdiagramm eines Einzelblockfalls im Nachrichtenauthentifizierungssystem gemäß den Prinzipien der vorliegenden Erfindung darstellt;

[0031] Fig. 10 ein Blockdiagramm eines Mehrfachblockfalls im Nachrichtenauthentifizierungssystem gemäß den Prinzipien der vorliegenden Erfindung darstellt;

[0032] Fig. 11a und Fig. 11b Blockdiagramme einer ENMAC-Ausführungsform des Nachrichtenauthentifizierungssystems gemäß den Prinzipien der vorliegenden Erfindung darstellen;

[0033] Fig. 12 ein Blockdiagramm einer ENMAC-Ausführungsform des Nachrichtenauthentifizierungssystems gemäß den Prinzipien der vorliegenden Erfindung darstellt;

[0034] Fig. 13a und Fig. 13b Blockdiagramme einer EHMAC-Ausführungsform des Nachrichtenauthentifizierungssystems gemäß den Prinzipien der vorliegenden Erfindung darstellen;

[0035] Fig. 14a und Fig. 14b Blockdiagramme einer SMAC-Ausführungsform des Nachrichtenauthentifizierungssystems gemäß den Prinzipien der vorliegenden Erfindung darstellen; und

[0036] Fig. 15 ein Blockdiagramm für eine SMAC-Ausführungsform des Nachrichtenauthentifizierungssys-

tems gemäß den Prinzipien der vorliegenden Erfindung darstellt.

AUSFÜHRLICHE BESCHREIBUNG

[0037] Im Folgenden sind illustrierte Ausführungsformen eines MAC-Konstruktionssystems und Verfahrens gemäß den Prinzipien der vorliegenden Erfindung zur Verarbeitung von Nachrichten beliebiger Länge beschrieben, die eine verbesserte Effizienz bereitstellen. In der folgenden Beschreibung umfasst der Ausdruck Hash-Funktion eine Komprimierungsfunktion f und eine iterierte Hash-Funktion F. Eine Hash-Funktion kann schlüssellos oder mit einem Schlüssel versehen sein, wobei F_k eine iterierte Schlüssel-Hash-Funktion und f_k eine Schlüssel-Komprimierungsfunktion darstellt. Es wird daran erinnert, dass $f_k(x)$ die Schlüssel-Komprimierungsfunktion ist, die eine Eingangsblockgröße von b Bits und eine Ausgangsgröße von t Bits aufweist, und deren Verkettungsvariablengröße und somit Schlüsselgröße ebenfalls t Bits ist. Gemäß eines Aspekts der vorliegenden Erfindung benutzt der MAC-Generator, je nach Größe der Nachricht, verschiedene Hash-Funktionsanordnungen, um den MAC zu erzeugen. Zum Beispiel könnte der MAC-Generator eine einzelne Iteration einer Schlüsselkomprimierungsfunktion als Hash-Funktion vornehmen, wenn die Nachricht x (und irgendwelche zusätzlich benötigten Bits) in einen Eingangsblock der Komprimierungsfunktion f passt. Für Nachrichten, die nicht in den Eingangsblock passen, benutzt der MAC-Generator vernestete Hash-Funktionen. Wie in **Fig. 9** dargestellt, wird eine Nachricht x in die Komprimierungsfunktion f zusammen mit irgendwelchen benötigten Füllzeichen, Nachrichtenlängefeldern, Blockanzeigefeldern oder anderen Feldern, die an die Nachricht x angehängt werden, eingegeben. Wenn die Nachricht x (und irgendwelche zusätzlich benötigten Bits) in den Eingangsblock für die Komprimierungsfunktion passt, wird eine einzelne Iteration der Schlüssel-Komprimierungsfunktion f 90 vorgenommen, wobei die Nachricht x und ein Schlüssel k dazu benutzt werden, einen MAC $f_k(x)$ für die Nachricht x zu erzeugen.

[0038] Andernfalls wird gemäß **Fig. 10**, wenn die Nachricht x (und irgendwelche zusätzlich benötigten Bits) nicht in einen Eingangsblock der Komprimierungsfunktion f passt, der Nachrichtenblock x in Teile, wie Teil **1** und Teil **2**, eingeteilt. Teile des Nachrichtenblocks können überlappende oder nicht überlappende Sätze der Bits sein, die zusammen die Nachricht x ergeben. Gemäß eines anderen Aspekts der vorliegenden Erfindung wird in der inneren Hash-Funktion F ein erster Teil und in der äußeren Hash-Funktion ein zweiter Teil benutzt, welcher als Komprimierungsfunktion f_{cv1} dargestellt ist. Zum Beispiel wird Teil **2** für die innere Hash-Funktion F bereitgestellt, wo Aufrufe an die, oder Iterationen der, Komprimierungsfunktion **100a** bis **100n** (falls benötigt) mit Blöcken Teil $\mathbf{2_1}$ und Teil $\mathbf{2_n}$ des Teils **2** einschließlich irgendwelcher angehängten Füllzeichen oder Felder vorgenommen werden, worin n>=1 ist. Die anfängliche Iteration oder der Aufruf **100a** an die Komprimierungsfunktion f benutzt eine Verkettungsvariable CV2, die ein Schlüssel oder ein von einem Schlüssel abgeleiteter Schlüssel oder, je nach Ausführungsform, der Standardanfangswert für die Hash-Funktion F sein könnte. Das Ergebnis der inneren Hash-Funktion F_{cv2} (Teil **2**) wird zusammen mit Teil **1** der gesamten Nachricht x und einer Verkettungsvariable CV1 der äußeren Hash-Funktion oder Komprimierungsfunktion f (**102**) zugeführt. Die Verkettungsvariable CV1 könnte ein Schlüssel oder ein von einem Schlüssel abgeleiteter Schlüssel oder, je nach Ausführungsform, der Standardanfangswert IV für die Hash-Funktion F sein. Der resultierende Wert f_{cv1} (Teil **1**, F_{cv2}(Teil2))Teil1) kann zur Erzeugung des in der Nachrichtenauthentifizierung benutzten MAC verwendet werden.

[0039] Die obige allgemeine Beschreibung kann zur Bereitstellung einer verbesserten Leistung gegenüber MAC-Erzeugungstechniken nach dem Stand der Technik verwendet werden. Um zum Beispiel eine verbesserte Effizienz gegenüber NMAC für kurze Nachrichten und auch eine etwas größere Effizienz für längere Nachrichten zu ermöglichen, wird folgende MAC-Konstruktion bereitgestellt. Es wird daran erinnert, dass $f_k(x)$ die Komprimierungsfunktion ist, deren Eingangsblockgröße b Bits und deren Ausgangsgröße t Bits ist, und dass die Größe der Verkettungsvariable und somit der Schlüsselgröße ebenfalls t Bits ist. Wie in **Fig. 11a** und **Fig. 11b** dargestellt, sieht eine bestimmte Ausführungsform der Konstruktion für einen MAC gemäß den Prinzipien der vorliegenden Erfindung folgendermaßen aus:

$ENMAC_k(x) = f_{k1}(x,pad,1)$ wenn $|x|<=b-2$ Bits $= f_{k1}(x_{pref}, F_{k2}(x_{suff}), 0)$ oder,

worin im ersten Fall die ersten b-2 Bits im Block zur Aufnahme der Nachricht x verwendet werden. Wenn die Nachricht x die ersten b-2 Bits nicht füllt, werden Füllzeichen benötigt, und der restliche Block, außer dem letzten Bit, wird mit einer obligatorischen 1, möglicherweise gefolgt von Nullen, gefüllt. Falls die Nachricht eine Länge von b-2 Bits hat, wird das b-1te Bit auf 1 gesetzt. In dieser Ausführungsform zeigt das letzte Bit des Blocks an, ob ein einziger Komprimierungsaufruf für ENMAC verwendet wird. Das letzte Bit des Blocks wird im Fall eines einzigen Komprimierungsaufrufs auf 1 und, falls mehrfache Aufrufe oder Iterationen der Komprimierungsfunktion f benötigt werden, auf 0 gesetzt. Im zweiten Fall, wenn die Dinge nicht in einen einzigen Block

passen, wird die Zeichenkette x in zwei Teile oder Segmente x_{pref} und x_{suff} aufgeteilt, worin

$x_{pref} = x_{1...}x_{b-t-1}$, und

$x_{suff} = x_{b-t...}x_{|x|}$.

[0040] Zuerst wird x_{suff} unter Einsatz eines Schlüsselwertes k2 hash-codiert, um das-t-Bit-Ergebnis von $F_{k2}(x_{suff})$ zu produzieren. Dann wird ein äußerer Komprimierungsaufruf unter Einsatz eines Schlüsselwertes k1 durchgeführt, wobei die ersten b-t-1 Bits auf x_{pref} und die nächsten t Bits auf das Ergebnis $F_{k2}(x_{suff})$ gesetzt werden, und wobei das letzte Bit auf Null gesetzt wird.

[0041] Die oben beschriebene ENMAC-Konstruktion kann eine SHA-1-Hash-Funktion als zugrunde liegende kryptographische Hash-Funktion, wie unten mit besonderer Bezugnahme auf Fig. 12 beschrieben, benutzen. Wie in Block **110** dargestellt, bestimmen die Verarbeitungsschaltkreise, die die ENMAC-Konstruktion implementieren, ob die Länge von x,|x| kleiner oder gleich 510 Bits ist. Falls ja, gehen sie zu Schritt **112**, um die 512-Bit-Nutzinformation von $f_{k1}()$ zu formen, indem sie x in die ersten 510 Bits lädt. Dann wird in Block **114** eine „1" an x angehängt, gefolgt von der Auffüllung mit so vielen Nullen wie nötig (möglicherweise keine), um die 511 Bits in Block **116** zu füllen. Wenn |x| kleiner als 510 Bits ist, wird der restliche Platz hinter der 1 mit Nullen aufgefüllt, oder falls |x| 510 Bits ist, wird nicht mit Nullen aufgefüllt, und nur eine einzige 1 in der 511ten Bitposition bei Block **114** angehängt. Bei Block **118** wird das letzte 512te (Blockanzeigebit) auf Eins gesetzt, um anzuzeigen, dass die Nachricht in einen einzigen Block passt. Bei Block **120** wird die Schlüssel-Komprimierungsfunktion $f_{k1}(x,pad,1)$ unter Einsatz von Schlüssel k1 als 160-Bit-Verkettungsvariable und die Nachricht x, Füllbits) und Blockanzeigebit als 512-Bit-Nutzinformation oder Eingangsblock ausgeführt. Anschließend wird das Ergebnis $f_{k1}(x,pad,1)$ ausgegeben und zur Bereitstellung des MAC bei Block **122** verwendet.

[0042] Wenn bei Block **110** die Nachricht x größer als 510 Bits ist, gehen die Verarbeitungsschaltkreise zu Block **124**, wo die Nachricht in zwei Teile x_{pref} und x_{suff} aufgeteilt wird, wobei $x_{pref} = x_1...x_{351}$ und $x_{suff} = x_{352}...x_{|x|}$ ist. Dann führen die Verarbeitungsschaltkreise bei Block **126** die Schlüssel-Hash-Funktion F_{k2} aus, wobei Schlüssel k2 und der Nachrichtenteil x_{suff} mit irgendwelchen zusätzlichen Füllbit(s) und/oder einem oder mehreren Bitfeldern als Nutzinformation zum Einsatz kommt, um das 160-Bit-Ergebnis von $F_{k2}(x_{suff})$ zu erzielen. Bei Block **128** werden die ersten 351 Bits der Nutzinformation der äußeren Komprimierungsfunktion f_{k1} auf x_{pref} und bei Block **130** die nächsten 160 Bits der Nutzinformation auf das in Block **126** berechnete Ergebnis von $F_{k2}(x_{suff})$ eingestellt. Das letzte 512te Bit der Nutzinformation wird bei Block **132** auf 0 gesetzt. Schließlich wird bei Block **134** die äußere Schlüssel-Komprimierungsfunktion f_{k1} auf die bei Block **128** bis **132** gebildete 512-Bit-Nutzinformation angewendet, und das Ergebnis $f_{k1}(x_{pref},F_{k2}(x_{suff}), 0)$ wird bei Block **136** zur Erzeugung eines MAC ausgegeben.

[0043] Tabelle 2 unten vergleicht die Anzahl der Komprimierungsaufrufe, die von der zugrunde liegenden Hash-Funktion, SHA-1 und von ENMAC für kurze Nachrichten verschiedener, in Stufen von 30 Bytes ansteigenden Größen benötigt werden. Eine signifikante Differenz besteht zwischen Tabelle 2 und der vorhergehenden Tabelle 1, in der einfache NMACs verglichen wurden. Für viele der kleinen Nachrichtengrößen hat NMAC die gleiche Effizienz wie die zugrunde liegende Hash-Funktion. Für größere Nachrichten besteht kein großer Unterschied zwischen der Effizienz von NMAC, ENMAC und der der zugrunde liegenden Hash-Funktion. Für Nachrichten mit einer Größe von 480 Bits zeigt der Eintrag in Tabelle 2 überraschenderweise, dass der ENMAC effizienter als die zugrunde liegende Hash-Funktion ist. Der Grund dafür ist, dass die zugrunde liegende SHA-1-Funktion **64** Bits für die Größeninformation reserviert, während ENMAC für Nachrichten unter 510 Bits nur 2 Bits reserviert. Somit sind die Einsparungen, die sich aus dem Einsatz von ENMAC ergeben, signifikant für Nachrichten, die in einen oder einige wenige Blöcke passen.

X in Stufen von 240 Bits	Anzahl f in F(x)	Anzahl f in ENMAC	Proz. Ineffizienz
240	1	1	0%
480	2	1	-50%
720	2	2	0%

960	3	3	0%
1200	3	3	0%
1440	3	4	33%
1680	4	4	0%
1920	4	5	25%
2160	5	5	0%
2400	5	6	20%

Tabelle 2: Vergleich zwischen der Anzahl der Komprimierungsaufrufe für kurze Nachrichten verschiedener Größen.

[0044] Wenn ein anderer Schlüssel k_3 zum Erzeugen von MACs für Nachrichten, die in einen einzigen Block passen, und Schlüssel $k = (k_1, k_2)$ zum Erzeugen von MACs für größere Nachrichten, die NMAC verwenden, benutzt würde, könnte man sagen, dass das System gesichert ist. Dies ist im Wesentlichen das, was hier bezweckt wird, aber anstatt einen anderen Schlüssel zum Erstellen eines anderen MAC zu benutzen, wird das nachlaufende Bit auf 1 gesetzt, wenn die Nachricht in einen Block passt, und im anderen Fall auf 0 gesetzt. Zweitens, während NMAC die Nutzinformation des äußeren Komprimierungsaufrufs mit Nullen auffüllt, setzt ENMAC einen Teil der Nachricht in den äußeren Aufruf.

[0045] Die ENMAC-Sicherheitsergebnisse sind ähnlich denen von NMAC und sollen im Folgenden zu Lehrzwecken beschrieben und bewiesen werden.

[0046] Lehrsatz 2. In t Schritten und q Abfragen, wenn die Schlüssel-Komprimierungsfunktion f ein ε_f sicherer MAC und die iterierte Schlüssel-Hash-Funktion F ε_F schwach kollisionsresistent ist, dann ist die ENMAC-Funktion ein $(\varepsilon_f + \varepsilon_F)$ sicherer MAC.

[0047] Beweis: Es sei angenommen, dass ein Gegner A_E mit einer Wahrscheinlichkeit ε_E erfolgreich gegenüber ENMAC ist, wobei t Zeitschritte und q adaptiv gewählte Abfragen an die ENMAC-Funktion angenommen werden. Dieser Gegner wird dazu benutzt, einen anderen Gegner A_f zu erstellen, der einen MAC fälscht, der für eine vorher nicht abgefragte Nachricht mit einer Schlüssel-Komprimierungsfunktion assoziiert ist. Diese Wahrscheinlichkeit, den MAC zu brechen, ist, an die Ausdrücken ε_E und ε_F gebunden, die beste Wahrscheinlichkeit für einen Gegner, eine Kollision in der Hash-Funktion F t Zeit und q Abfragen zu finden. Die Wahrscheinlichkeit, den MAC auf diese bestimmte Weise unter Einsatz von A_E zu brechen, muss kleiner als die beste Wahrscheinlichkeit sein, den MAC in irgendeiner Weise ε_f zu brechen. Dies kann dazu verwendet werden, ε_E zu binden. Der Algorithmus A_f, der zur Fälschung des Schlüsselkomprimierungs-MAC verwendet wird, ist im Folgenden definiert.

[0048] Wählen Sie irgendein k_2
Für i...q
$A_E \rightarrow x_i$
Wenn $x_i > b - 2$
$A_E \leftarrow f_{k1}(x_i, pad, 1)$
oder
$A_E \leftarrow f_{k1}(1, x_i, pref, F_{k2}(x_{i,suff}), 0$
$A_E \rightarrow (x, y)$
Wenn $x < b - 2$
Ausgabe $(x, pad, 1), y$
oder
Ausgabe $(x_{pref}, Fk_2(x_{suff}), 0)y$

[0049] Angenommen, $\varepsilon_E = \varepsilon_{E1} + \varepsilon_{E+}$, worin ε_{E1} die Wahrscheinlichkeit darstellt, dass ENMAC angegriffen wird, und die von A_E gefälschte ENMAC-Nachricht ungefähr eine Blockgröße, oder genauer ausgedrückt b – 2 Bits, ist. Und angenommen, E_+ ist das Ereignis und ε_{E+} die Wahrscheinlichkeit, dass ENMAC angegriffen wird, und die von A_E gefälschte ENMAC-Nachricht größer als eine Blockgröße ist. Weiterhin ist $\varepsilon_{E+} = \varepsilon_{E+pref\neq} + \varepsilon_{E+pref=}$ worin $\varepsilon_{E+pref\neq}$ die Wahrscheinlichkeit ausdrückt, dass der ENMAC mit einer Mehrfachblock-Nachricht gefälscht wurde, und dass das Präfix der Nachricht nicht gleich dem Präfix irgendeiner der vorher von A_E abgefragten Nachrich-

ten ist. Und $\varepsilon_{E+pref=}$ ist die Wahrscheinlichkeit, dass der ENMAC mit einer Mehrfachblock-Nachricht gefälscht wurde, und dass das Präfix der Nachricht gleich dem Präfix einiger vorher von A_E abgefragten Nachrichten ist. In diesem Fall muss das Suffix der gefälschten Nachricht ein anderes als das Suffix der Nachrichten mit dem gleichen Präfix sein.

P[Fälschung von f-MAC]
=P[f-MAC gefälscht über E_1 + P[f-MAC gefälscht über E_+] (1)

=ε_{E1}+P[f-MAC gefälscht über E_+]
=ε_{e1}+P[f-MAC gefälscht über $E_{+pref\neq}$]
+ P[f-MAC gefälscht über $E_{+pref=}$] (2)

=ε_{E1}+$\varepsilon_{E+pref\neq}$ +P[f-MAC gefälscht über $E_{+pref=}$]
=ε_{E1}+$\varepsilon_{E+pref\neq}$ +P[$E_{+pref=}$ ∩ keine Suffix-Kollision im Satz
bei gleichem Präfix] (3)

=ε_{E1}+$\varepsilon_{E+pref\neq}$ +1–P[$E_{+pref=}$ ∪ Suffix-Kollision im Satz bei
gleichem Präfix] (4)

=ε_{E1}+$\varepsilon_{E+pref\neq}$ +1–P[$E_{+pref=}$] – P [Kollision im Satz] +
P[$E_{+pref=}$ ∩ Kollosion im Satz]
≥=ε_{E1}+$\varepsilon_{E+pref\neq}$ + 1 – P [$\overline{E}$+pref= – P [Kollision in q Satz] (5)

≥ =ε_{E1}+$\varepsilon_{E+pref\neq}$ +1–1 +$\varepsilon_{E+pref=}$ –P [Kollision in q Abfragen] (6)

≥ =ε_{E1}+$\varepsilon_{E+pref\neq}$ +1–1 + $\varepsilon_{E+pref=}$ – ε_F
≥ ε_{E1}+ε_{E+} – ε_F
≥ ε_E – ε_F
(7)

e_f ≥ P[Fälschung von f-MAC über ENMAC-Fälschung] ≥ ε_E – ε_F
(8)

e_f ≥ ε_E – ε_F
deshalb e_E ≤ ε_f – ε_F (9)

[0050] Gleichung 1 bricht die Wahrscheinlichkeit, einen neuen f-Mac zu fälschen, in die Wahrscheinlichkeit auf, einen neuen f-MAC durch Fälschen eines ENMAC MAC, entweder Einzelblock oder Mehrfachblock, zu fälschen. Die Wahrscheinlichkeit, f durch Brechen eines Mehrfachblock-ENMACs aufzubrechen, wird in Gleichung 2 in den Fall, dass kein Präfix gleich irgendeinem anderen Präfix in allen abgefragten Nachrichten ist, und in den Fall, dass unter den abgefragten Nachrichten einige das gleiche Präfix haben, aufgebrochen. In Gleichung 3 wird die Wahrscheinlichkeit, dass der f-MAC über $E_{+pref=}$ gefälscht wurde, mit dem Fall gleichgesetzt, dass die Wahrscheinlichkeit von $E_{+pref=}$ eintritt, und dass keine Kollisionen in der Hash-Funktion des Suffix in den Nachrichten mit dem gleichen Präfix vorkommen. In Gleichung 4 wird Gleichung 3 unter Einsatz des Demorgan-Gesetzes umgeschrieben. In Gleichung 6 wird die Wahrscheinlichkeit einer Kollision in dem Satz mit dem gleichen Präfix durch die Wahrscheinlichkeit der Kollision mit allen g Abfragen ersetzt. Gleichung 9 ist das gewünschte Ergebnis für die Wahrscheinlichkeit, den ENMAC zu fälschen: ε_E ist kleiner als ε_f, die Wahrscheinlichkeit den MAC zu fälschen, plus ε_F, die Wahrscheinlichkeit, eine Kollision zu finden.

[0051] Da in der Praxis Daten oft in Bytes verarbeitet werden, ist es angebracht, den Fall eines Einzelblocks auszuführen, wenn die Länge der Nachricht x kleiner als b-8 Bits, anstatt der oben angegebenen b-2 Bits ist. Im Fall eines Mehrfachblock-ENMAC kann die Bildung von x_{suff}, beginnend an einer Nicht-Wort-Grenze, eine Neuausrichtung aller Worte in x_{suff} verursachen. Dies kann dadurch vermieden werden, dass man, wie im Folgenden dargestellt, aus praktischen Gründen eine andere Variante von ENMAC verwendet, wobei Byte-Größen, anstatt Bits zum Einsatz kommen.

$ENMAC_k(x) = f_{k1}(x,pad,1)$ wenn |x| <= 504 Bits = $f_{k1}(F_{k2}(x_{pref}),x_{suff}, 0)$ oder

worin für SHA-1 als zugrunde liegende kryptographische Hash-Funktion f,

$x_{pref} = x_1...x_{|x|-344}$, und

$x_{suff} = x_{|x|-343}...x_{|x|}$.

[0052] Bei Nachrichten mit Längen bis zu 63 Bytes (504 Bits) und zusätzlich zu irgendwelchen zusätzlichen Füllzeichen, wie 1 gefolgt von Nullen, um die Nachricht auf 504 Bits aufzufüllen, wird das letzte Byte als Blockanzeiger oder „X0000001" reserviert, worin eine 1 eine Einzelblocknachricht bezeichnet, und das X eine „1" nach einer nicht aufgefüllten 504-Bit-Nachricht sein kann. Bei nicht aufgefüllten Nachrichten von weniger als 504 Bits ist das X eine „0". Bei Nachrichten von über 504 Bits wird die Nachricht in die Teile x_{pref} und x_{suff} aufgeteilt. Wobei die Länge von x_{suff} 43 Bytes (344 Bits) und die Länge von x_{pref} = Länge der Nachricht – 344 Bits ist.

[0053] Neben der/den oben beschriebenen Ausführungsformen kann das Nachrichtenauthentifizierungssystem gemäß den Prinzipien der vorliegenden Erfindung Eingangsparameter und/oder Komprimierungs- und/oder Hash-Funktionen oder andere Operationen, Schlüsselwerte auslassen und/oder hinzufügen und/oder Variationen oder Teile des beschriebenen Systems benutzen. Zum Beispiel ist im Folgenden eine in Fig. 13a und Fig. 13b dargestellte Ausführungsform des Nachrichtenauthentifizierungssystems beschrieben, welches als verbessertes HMAC-System benutzt wird.

$EHMAC_k(x)$= F(k ⊕opad,x,1) wenn|x|<= b-a-1-andere Felder = F(k ⊕opad, x_{pref}, F(k ⊕ipad, x_{suff}), 0) oder.

[0054] Im ersten Fall, Fig. 13a, passt die Nachricht x in einen einzigen Block. Dies bedeutet, dass die Nachricht x kleiner als b-1-other fields sein muss, wobei „other fields" einige Bits aufgrund von Auffüllung und/oder Längenanhängesystemen der Hash-Funktion F aufweisen kann. In der Annahme, dass x klein genug ist, wird eine größere Eingabe gebildet, deren erster Teil k ⊕opad, gefolgt von x ist, auf welches wiederum ein auf 1 gesetztes Bit folgt. Diese längere Nachricht wird in die zugrunde liegende Hash-Funktion F eingegeben. Innerhalb von F sehen wir, dass zuerst durch Aufruf der Komprimierungsfunktion f(k ⊕opad) ein Schlüssel k1 erstellt wird, wobei k möglicherweise auf die entsprechende Länge aufzufüllen ist. Das Ergebnis wird als Verkettungsvariable für den nächsten Aufruf an die Komprimierungsfunktion verwendet, deren Nutzinformation (x,1) durch Füllzeichen aufgefüllt und/oder gemäß der Spezifikation der Hash-Funktion F an die Länge angehängt wird.

[0055] In Fig. 13b, wo die Nachricht x zusammen mit zusätzlichen obligatorischen Feldern nicht in einen einzigen Block passt, ist die Zeichenkette x in zwei Teile oder Segmente x_{pref} und x_{suff} aufgeteilt, wobei

$x_{pref} = x_1...x_{b\text{-}t\text{-}1\text{-}andere}$, und

x_{suff} = Rest von x

[0056] Zuerst wird in einer inneren Hash-Funktion **130** ein bitweises exclusive-or zwischen Schlüssel k und ipad durchgeführt, um k2 zu produzieren, welches zusammen mit dem Eingangsblock x_{suff1} als Verkettungsvariable benutzt wird. Die Komprimierungsfunktion f wird aufgerufen, bis Block x_{suffn} mit irgendwelchen Füllzeichen, angehängten Längenfeldern oder anderen Feldern in die letzte Komprimierungsfunktion eingegeben wird, um das Ergebnis der Hash-Funktion für F (k⊕ipad, x_{suff}) zu produzieren, wobei k möglicherweise auf die entsprechende Länge aufzufüllen ist. Bei einer äußeren Hash-Funktion **132** wird der Schlüssel k1 durch Aufrufen einer Komprimierungsfunktion **134** mit dem Wert IV als Verkettungsvariable und k⊕opad als Eingabe bestimmt. Der Wert k1 wird als Verkettungsvariable für eine Komprimierungsfunktion **136** benutzt, wobei die Eingabe auf x_{pref} mit vorlaufendem F(k⊕ipad, x_{suff}) und nachlaufender Null eingestellt ist. Das Ergebnis F(k⊕opad, x_{pref}, F(k⊕ipad, x_{suff}), 0) kann zur Bereitstellung des MAC verwendet werden.

[0057] Fig. 14a und Fig. 14b zeigen wiederum eine andere Ausführungsform des Nachrichtenauthentifizierungs-systems, welches als SMAC-System, wie unten im Zusammenhang einer spezifischen Beispielimplementierung für Bytes beschrieben, benutzt wird.

$$SMAC(x) = f_K(x,pad,1) \quad \text{wenn } |x| <= 63 \text{ bytes}$$

$$= f_K(F(x_{prefix}),x_{suffix},0) \quad \text{wenn } |x| > 63 \text{ bytes}$$

x_{pref}: bytes $x_1 ... x_{|x|-43}$

x_{suff}: bytes $x_{|x|-42} ... x_{|x|}$

[0058] Wie bei den anderen Ausführungsformen besteht SMAC aus zwei Fällen, dem Einzelblockfall (<=63 Bytes) (Fig. 14a) und dem Mehrfachblockfall (>63 Bytes) (Fig. 14b). In beiden Fällen ergeht ein Aufruf an die Schlüssel-Komprimierungsfunktion f, z.B. eine SHA-Funktion. Im Einzelblockfall werden keine weiteren Funktionsaufrufe benötigt. Im Mehrfachblockfall wird jedoch eine schlüssellose Hash-Funktion F **140**, z.B. die Standard SHA1_HASH-Funktion, auf den Anfangsteil der Nachricht x_{pref} angewendet. Dann werden das Hash-Ergebnis und der Rest der Nachricht in einen Eingangs- oder Nutzinformationsblock eingefügt, und es ergeht ein Aufruf an eine Schlüssel-Komprimierungsfunktion f **142**. Weitere Details über das Laden der SHA-1 Komprimierungsfunktion f sind in Tabelle 3 und 4 angegeben.

[0059] Wie dargestellt, wird das letzte 512te Bit der SHA-1-Komprimierungsfunktion als „Einzelblockanzeigebit" benutzt und wird im Einzelblockauf auf 1, und im Mehrfachblockfall auf 0 gesetzt. Da die Nachricht in dieser Ausführungsform in Mehrfachen von Bytes verarbeitet wird, kann keines der restlichen Bits im letzten Byte zur Verarbeitung der Nachricht verwendet werden. Daher wird das letzte (64te) Byte der Komprimierungsfunktion als Ganzes reserviert. Im Mehrfachblockfall werden die Bits 505–511, wie in Tabelle 4 angegeben, ebenfalls auf Null gesetzt. Im Einzelblockfall werden Bits 506–511 auf Null gesetzt; das 505te Bit wird jedoch als extra Füllbit benutzt, dessen Funktion klar wird, sobald das im Einzelblockfall verwendete Füllsystem erklärt wird.

[0060] Nachrichten, die einen Block teilweise füllen, benötigen eine Füllmethode. Im Mehrfachblockfall ist keine Füllmethode zum Füllen der Komprimierungsfunktion erforderlich, da der Block, wie in Tabelle 4 dargestellt, voll ist. Die SHA1_Hash-Funktion benutzt jedoch nicht ihre eigenen Füllzeichen, wenn sie x_{pref} hash-codiert. Um Nachrichten im Einzelblockfall aufzufüllen, wird eine 1 an die Nachricht angehängt, dann werden so lange Nullen angehängt (möglicherweise keine), bis die restlichen Bits den Block füllen, oder genauer gesagt, bis das 505te Bit gefüllt ist. Beispielsweise wird für den Fall, dass die Einzelblock-Nachricht 63 Bytes oder 504 Bits lang ist, eine 1 zum 505ten Bit hinzugefügt. Die restlichen Bits 506–512 werden wie oben beschrieben gefüllt.

[0061] Im Mehrfachblockfall wird die Hash-Funktion F **140** in den Blöcken x_{pref1} bis x_{prefn} auf alle außer den letzten 43 Bytes der Nachricht angewendet, die eine 20-Byte-Übersicht ausgibt. Die letzten 43 Bytes werden nicht in der Hash-Funktion F verarbeitet, so dass sie von der Komprimierungsfunktion f **142** verarbeitet werden können. Der Grund für 43 Bytes ist, dass von den 64 verfügbaren Bytes die ersten 20 Bytes dazu dienen, die Übersicht zu laden, und das letzte Byte, wie in Tabelle 4 aufgezeigt, speziell für die SHA-1-Hash-Funktion und die SHA-1-Komprimierungsfunktion reserviert wird.

1. Byte	2. Byte		62. Byte	63. Byte	speziell gesetztes 64. Byte							
					505	506	507	507	509	510	511	512.Bit
X_1	X_2 or pad	.....	X_{62} or pad	X_{63} or pad	Padbit	0	0	0	0	0	0	1 Einzel-block-Anzeigebit

Tabelle 3: Einzelblock – Laden der SHA-1-Komprimierungs funktion

$Y_i ... Y_{20}$ = SHA–HASH($x_1 .... x_{|x|-43}$)

1. Byte	2. Byte		20. Byte	21. Byte		62. Byte	63. Byte	spez. Gesetztes 64. Byte							
								5 0 5	5 0 6	5 0 7	5 0 7	5 0 9	5 1 0	5 1 1	512. Bit
								0	0	0	0	0	0	0	0 Einzelblock-Anzeigebit
Y_1	Y_2	...	Y_{20}	$X_{\|x\|-42}$	...	$X_{\|x\|-1}$	$X_{\|x\|}$								

Tabelle 4: Mehrfachblock – Laden der SHA1-Komprimierungsfunktion

[0062] Fig. 15 zeigt ein Ablaufdiagramm für die SMAC-Konstruktion. Zu Anfang wird der Schlüssel mit dem IV XOR-bearbeitet und in die Verkettungsvariable der SHA1-Komprimierungsfunktion geladen, wie bei Block **148** dargestellt. Bei Block **150** bestimmen Verarbeitungs-Schaltkreise, ob |x|> 63 Bytes ist. Wenn nicht, gehen die Verarbeitungsschaltkreise wie im Einzelblockfall vor, wobei die Nachricht x bei Block **152** auf die linke Seite des 512-Bit-Blocks der Komprimierungsfunktion f geladen wird. Bei Block **154** hängen die Verarbeitungs-Schaltkreise eine „1" an das nächste Bit. Bei Block **156** wird der Rest des Blocks mit Nullen bis zum letzten 512^{ten} Bit gefüllt, welches in Block **158** auf 1 gesetzt wird. Bei Block **160** wird die Komprimierungsfunktion f aufgerufen, wobei die Verkettungsvariable (K XOR IV) und die Nutzinformation der Blöcke **152–158** zum Einsatz kommt. Bei Block **162** wird der 20-Byte-MAC zurückgegeben.

[0063] Wenn |x| >63 Bytes ist, gehen bei Block **150** die Verarbeitungsschaltkreise wie für den Mehrfachblockfall vor. Bei Block **164** wird die Nachricht x in zwei Stücke aufgeteilt: x_{pref} Bytes $x_1...x_{|x|-43}$ und x_{suff} Bytes $x_{|x|-43}$ ....$x_{|x|}$. Bei Block **166** wird die SHA1_HASH-Funktion mit x_{pref} aufgerufen, und ein 20-Byte-Ergebnis wird produziert. Bei Block **168** wird das 20-Byte-Ergebnis auf die linke Seite des 64-Byte-Blocks der SHA1-Komprimierungsfunktion geladen, und x_{suff} wird zu Bytes 21 bis 63 hinzugefügt. Bei Block **170** wird das letzte 64te Byte auf 0 gesetzt. Schließlich wird bei Block **172** die SHA1-Komprimierungsfunktion aufgerufen, wobei die zu Anfang berechnete Verkettungsvariable (K XOR IV) und die Nutzinformation der Blöcke **168** und **170** zum Einsatz kommen. Der 20-Byte-MAC wird bei Block **162** zurückgegeben.

[0064] SMAC ist NMAC ähnlicher als HMAC, deshalb soll es jetzt mit NMAC anstatt HMAC verglichen werden. NMAC enthält einen inneren Aufruf an die Hash-Funktion F und einen äußeren Aufruf an die Komprimierungsfunktion f. SMAC verhält sich gleich für Nachrichten, die größer als 63 Bytes sind, überspringt jedoch den Hash-Aufruf für kleinere Nachrichten. Für längere Nachrichten verarbeitet SMAC einen Teil der Nachricht im äußeren Komprimierungsaufruf, so dass der vom internen Hash-Funktions-Aufruf verarbeitete Text reduziert wird. NMAC tut dies nicht, sondern füllt stattdessen den Rest der Nutzinformation der äußeren Komprimierungsaufrufe mit Nullen. In NMAC wird die innere Hash-Funktion mit einem Schlüssel versehen, während SMAC den internen Aufruf nicht mit einem Schlüssel versieht. Der interne SMAC-Aufruf kann mit einem Schlüssel versehen werden, was jedoch aus Effizienzgründen in dieser Ausführungsform nicht geschah. Die Sicherheit wird davon, im Grunde genommen, nicht betroffen, weil es für unmöglich gehalten wird, eine Kollision selbst in der schlüssellosen SHA1-HASH-Funktion zu finden.

[0065] Im Folgenden ist der Code, der zur Implementierung von SMAC verwendet werden sollte, angegeben: Ausgaben an intern gespeicherte Daten:

```
MAC                    32 Bits
```

```
/* smac ruft folgende Funktionen auf:*/
     Sha1_comp(vorzeichenloses zch cv[20],
vorzeichenloses zch temp[64],vorzeichenloses zch
adigest[20])
     {/*sha1_comp ist die sha1 Komprimierungsfunktion,
cv ist die 160-Bit-Verkettungsvariable,temp ist die
512-
     Bit-Nutzinformation,und das Ergebnis wird im 160-
Bit-Adigest ausgegeben*/
     ............
}

     SHA1_HASH(vorzeichenloses zch *M, int textlen,
     vorzeichenloses zch adigest[20]
     { /*SHA1_HASH ist die Hash-Funktion, M ist die
Nachricht, textlen ist die Anzahl der Bytes in der
Nachricht
     und das Ergebnis wird im 160-Bit-Adigest
ausgegeben */
     ................
}

smac(int keylen, vorzeichenloses zch *K, int textlen,
vorzeichenloses zch*M,vorzeichenloses zch mac[20])

{int i j:
vorzeichenloses zch cv[20],temp[64];
   /*20 Byte Verkettungsvariable cv auf Standardwert
   IV0 setzen, wie in fips180* definiert*/
     cv[0]=0x67;cv[1]=0x45;cv[2]=0x23;cv[3]=0x01;
     cv[4]=0xef;cv[5]=0xcd;cv[6]=0xab;cv[7]=0x89;
     cv[8]=0x98;cv[9]=0xba;cv[10]=0xdc;cv[11]=0xfe;
     cv[12]=0x10;cv[13]=0x32;cv[14]=0x54;cv[15]=0x76;
     cv[16]=0xc3;cv[17]=0xd2;
     cv[18]=0xe1;cv[19]=0xf0;
   /*XOR Schlüssel auf die Verkettungsvariable*/
     für(i=0;i<keylen;i++)
          cv[i]=cv[i]^K[i];
```

```
   /*für    temp    Komprimierungsblock    nur    Nullen
   eingeben}*/
     für(i=0;i<64;i++) temp[i]=0;
     wenn(textlen<=63) {
     /*Nachricht auf die linke Seite laden*/
     für(i=0;i<textlen;i++)
          temp[i]=M[I;]
     temp[i]=0x80;                /*'1'anhängen,  rest-
liche Bits wurden bereits auf 0 gesetzt*/
     temp[63] =temp[63] | 0x01;        /*512.  Bit  auf
'1' setzen*/
     sha1_comp(cv,temp.mac);
     }
     oder {/*textlen>63*/
     /*SHA1_HASH auf Präfix von M*/
     SHA1_HASH(M,textlen-43,mac);
     für(i=0;i<20;i++)
          temp[i]=mac[i];
     /*Übersicht auf linke Seite kopieren*/
     für(i=20;i=<63;i++)
          temp[i]=M[textlen-43+(i-20)];
                        /*nun Suffix von M. kopieren*/
     temp[63]=0x00;   /*letztes Bit auf Null setzen. */
     sha1_comp(cv,temp,mac);
   }
}
```

[0066] Das MAC-System wurde für den Einsatz bestimmter Hash- oder Komprimierungsfunktionen, wie z.B. für SHA-1, beschrieben, es können aber auch andere Hash-Funktionen oder verwandte kryptographische Funktionen sowie andere oder zusätzliche Funktionen verwendet werden. Darüber hinaus wurden bestimmte Bit- oder Byte-Werte für die Nachricht, für Nutzinformationen, Verkettungsvariablen und Schlüsselwerte beschrieben, diese Zahlen können sich jedoch je nach Ausführungsform ändern. Des Weiteren können die Schlüsselwerte ein Schlüssel sein, der von einem Schlüssel oder einem/mehreren Teilen desselben abgeleitet wurde. Es versteht sich, dass andere Notationen, Bezugszeichen und Kennzeichnungen für die verschiedenen Werte, Eingaben und Architekturblöcke verwendet werden können. Zum Beispiel wird der Ausdruck Komprimierungsfunktion f und Hash-Funktion F benutzt, wo die iterierte Hash-Funktion F unter Einsatz von iterierten oder verketteten Komprimierungsfunktionen f konstruiert wird. Es versteht sich, dass eine Komprimierungsfunktion auch eine Hash-Funktion ist.

[0067] In anderen Ausführungsformen kann die für das Nachrichtenauthentifizierungssystem beschriebene Funktionalität mit Verarbeitungsschaltkreisen in einem Heimatstandortregister (HLR), einer Home-MSC, einem Besucherauthentifizierungszentrum, einem Besucherstandortregister (VLR) und/oder einer Besucher-MSC realisiert werden. Darüber hinaus kann das Nachrichtenauthentifizierungssystem und Teile desselben in einer drahtlosen Einheit, einer Basisstation, einem Basisstationscontroller, MSC, VLR, HLR oder einem anderen Subsystem eines drahtlosen Kommunikationssystems realisiert werden. Je nach Ausführungsform kann der MAC in Assoziation mit der Nachricht gesendet werden, wobei er mit einem am empfangenden Ende erzeugten MAC verglichen und/oder geprüft wird. Durch zusätzliche Funktionalität kann der MAC geändert oder transformiert werden, bevor er in Assoziation mit der Nachricht gesendet wird, und die gleiche Funktionalität kann

für den am empfangenden Ende zu Vergleichs- und/oder Prüfungszwecken (Nachrichtenauthentifizierung) erzeugten MAC realisiert werden. Schließlich könnte der MAC gesendet, und der empfangene MAC sowie der am empfangenden Ende erzeugte MAC durch zusätzliche Funktionalität zur Realisierung der Nachrichtenauthentifizierung geändert oder transformiert werden. Ein Beispiel für zusätzliche Funktionalität wäre der Einsatz der unwichtigsten 32 Bits des MAC für irgendwelche Vergleiche oder Prüffunktionen bei der Realisierung der Nachrichtenauthentifizierung. Der geänderte und/oder transformierte MAC selbst kann als MAC oder TAG bezeichnet werden.

[0068] Zusätzlich, obwohl das Nachrichtenauthentifizierungssystem im Zusammenhang mit dem drahtlosen Kommunikationssystem beschrieben wird, kann es dazu benutzt werden, die Integrität einer Kommunikationsnachricht, die über ein beliebiges Netzwerk oder Kommunikationsmedium von einem Sendepunkt an einen Empfangspunkt geschickt wird, zu prüfen oder sie zu authentisieren. Es versteht sich, dass das System und Teile desselben sowie Teile der beschriebenen Architektur in den Verarbeitungs-Schaltkreisen in der Einheit oder an verschiedenen Standorten des Kommunikationssystems, oder in anwendungsspezifischen integrierten Schaltkreisen, softwaregetriebenen Verarbeitungsschaltkreisen, programmierbaren Logikeinheiten, Firmware-, Hardware- oder anderen Anordnungen diskreter Komponenten, die von einem normalen Fachmann im Zusammenhang mit dieser Offenbarung verstanden würden, implementiert oder mit diesen integriert werden können. Was beschrieben wurde, dient lediglich zur Veranschaulichung der Prinzipien der vorliegenden Erfindung. Der Fachmann wird ohne Weiteres erkennen, dass diese und verschiedene andere Modifikationen, Anordnungen und Methoden auf die vorliegende Erfindung angewendet werden können, ohne die hierin veranschaulichten und beschriebenen beispielhaften Anwendungen genau zu befolgen, und ohne vom Geltungsbereich der vorliegenden Erfindung, wie beansprucht, abzuweichen.

Patentansprüche

1. Verfahren zum Verarbeiten einer Nachricht zur Authentifizierung unter Verwendung einer Komprimierungsfunktion (**90**) und einer vernesteten Hash-Funktion mit einer inneren Hash-Funktion (**100**) und einer äußeren Hash-Funktion (**102**), wodurch die innere Hash-Funktion (**100**) ein Ergebnis für die äußere Hash-Funktion (**102**) produziert, gekennzeichnet durch die folgenden Schritte:
Durchführen einer einzigen Iteration der Komprimierungsfunktion (**90**) unter Verwendung eines Schlüssels (K) und der Nachricht (X) als Eingaben, wenn die Nachricht (X) in einen Eingangsblock der Komprimierungsfunktion (**90**) paßt; und
wenn die Nachricht größer als der Eingangsblock der Komprimierungsfunktion (**90**) ist, Zuführen der Nachricht (X) und eines Schlüssels zu der vernesteten Hash-Funktion, um die Nachricht (X) zu verarbeiten.

2. Verfahren nach Anspruch 1, mit den folgenden Schritten:
Durchführen einer Hash-Funktion als die innere Hash-Funktion (**100**) unter Verwendung eines ersten Teils (Teil **2**) der Nachricht als Eingabe, um das Ergebnis zu erreichen; und
Durchführen einer Schlüssel-Hash-Funktion (**102**) als die äußere Hash-Funktion unter Verwendung des zweiten Teils (Teil **1**) der Nachricht und des Ergebnisses als Eingabe.

3. Verfahren nach Anspruch 2, wobei die Hash-Funktion eine iterierte Hash-Funktion F und die Schlüssel-Hash-Funktion eine Schlüssel-Komprimierungsfunktion f ist.

4. Verfahren nach Anspruch 2, wobei die Hash-Funktion eine iterierte Hash-Funktion F und die Schlüssel-Hash-Funktion eine iterierte Hash-Funktion F ist.

5. Verfahren nach Anspruch 1, mit den folgenden Schritten:
Verwenden eines Ergebnisses der Komprimierungsfunktion, um einen Nachrichtenauthentifizierungscode (TAG) zu erzeugen; und
Senden des Nachrichtenauthentifizierungscodes (TAG) in Assoziation mit der Nachricht (MESSAGE) zur Authentifizierung der Nachricht unter Verwendung des Nachrichtenauthentifizierungscodes (TAG).

6. Verfahren nach Anspruch 1, mit den folgenden Schritten:
Verwenden eines Ergebnisses der Komprimierungsfunktion, um einen Nachrichtenauthentifizierungscode (TAG') zu erzeugen; und
Vergleichen des Nachrichtenauthentifizierungscodes (TAG') mit dem Nachrichtenauthentifizierungscode (TAG) einer empfangenen Nachricht, der mit der Nachricht (MESSAGE) empfangen wird, wodurch die Nachricht authentisch ist, wenn der Nachrichtenauthentifizierungscode (TAG') und der Nachrichtenauthentifizierungscode (TAG) übereinstimmen.

7. Nachrichtenauthentifizierungssystem, das eine Komprimierungsfunktion (**90**) und eine vernestete Hash-Funktion mit einer inneren Hash-Funktion (**100**) und einer äußeren Hash-Funktion (**102**) verwendet, wodurch die innere Hash-Funktion (**100**) ein Ergebnis für die äußere Hash-Funktion (**102**) produziert, wobei das System durch folgendes gekennzeichnet ist:
Verarbeitungsschaltkreise, die so ausgelegt sind, daß sie eine einzige Iteration der Komprimierungsfunktion (**90**) unter Verwendung eines Schlüssels (K) und der Nachricht (X) als Eingaben durchführen, wenn die Nachricht (X) in einen Eingangsblock der Komprimierungsfunktion (**90**) paßt, und die Nachricht (X) und einen Schlüssel der vernesteten Hash-Funktion zuführen, um die Nachricht (X) zu verarbeiten, wenn die Nachricht größer als der Eingangsblock der Komprimierungsfunktion (**90**) ist.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

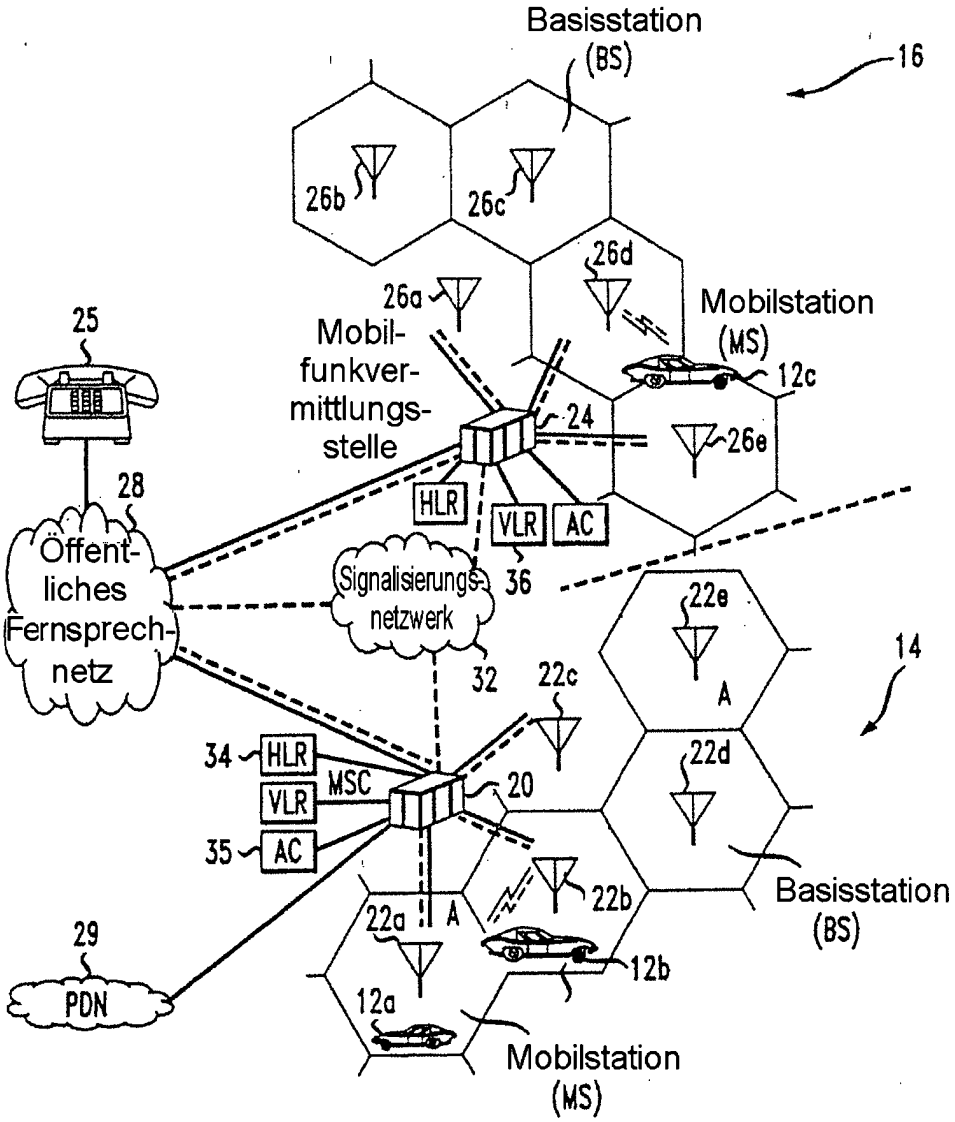


FIG. 2

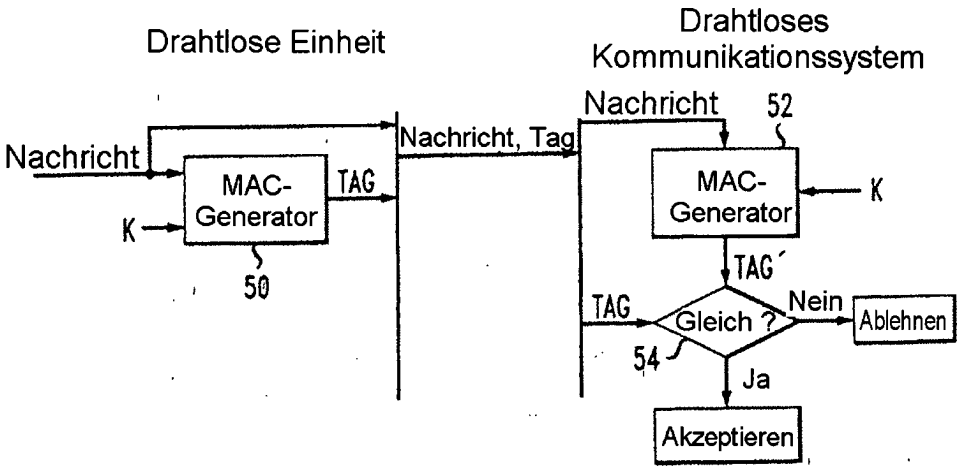

Drahtlose Einheit
Drahtloses Kommunikationssystem
Nachricht
MAC-Generator
TAG
K
50
Nachricht, Tag
Nachricht
52
MAC-Generator
K
TAG´
TAG
Gleich ?
Nein
Ablehnen
54
Ja
Akzeptieren

FIG. 3

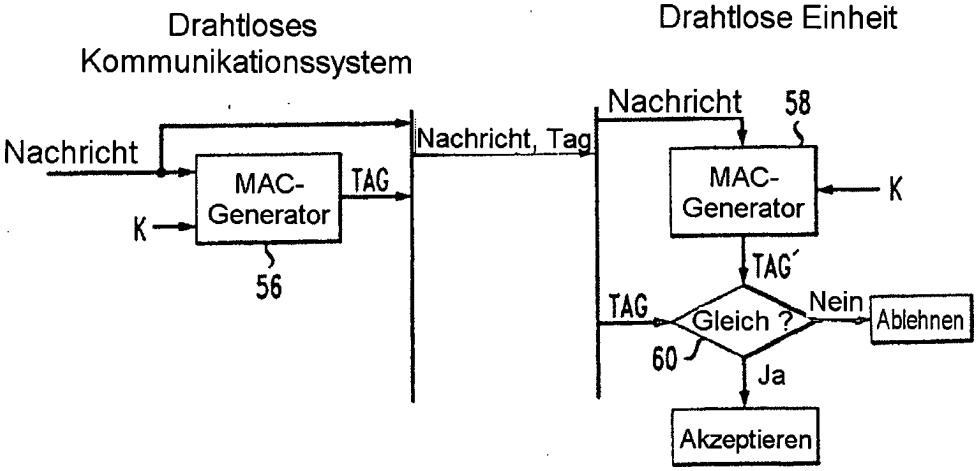

Drahtloses Kommunikationssystem
Drahtlose Einheit
Nachricht
MAC-Generator
TAG
K
56
Nachricht, Tag
Nachricht
58
MAC-Generator
K
TAG´
TAG
Gleich ?
Nein
Ablehnen
60
Ja
Akzeptieren

FIG. 4

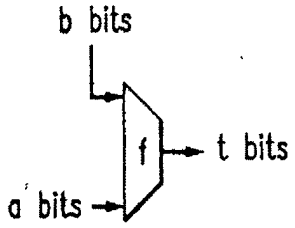


FIG. 5

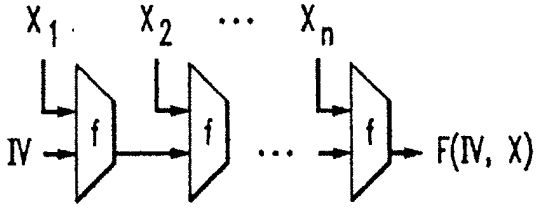


FIG. 6

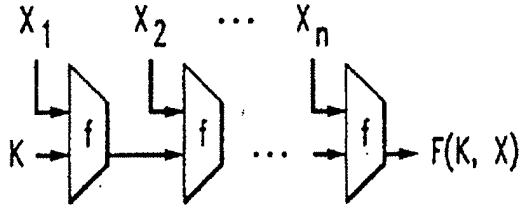


FIG. 7

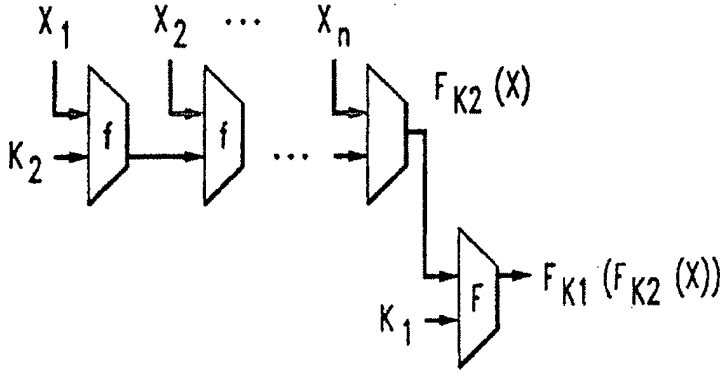


FIG. 8

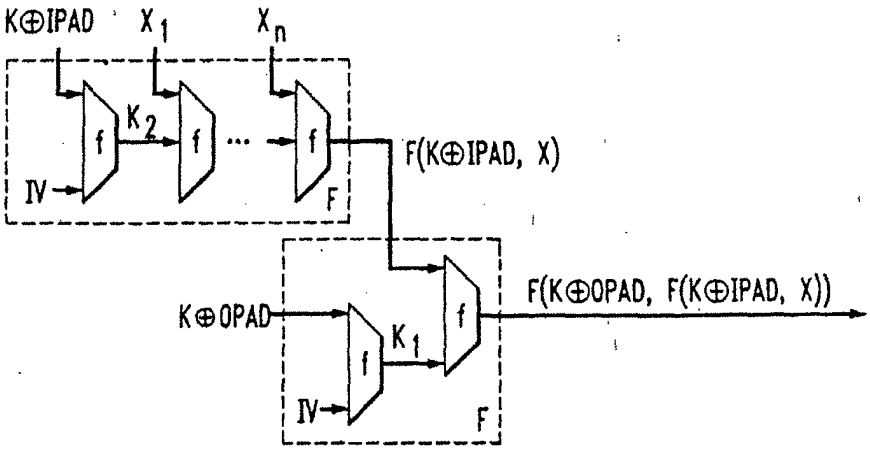

K⊕IPAD
X1
Xn
f
K2
f
...
f
IV
F
F(K⊕IPAD, X)
K⊕OPAD
f
K1
f
IV
F
F(K⊕OPAD, F(K⊕IPAD, X))

FIG. 9

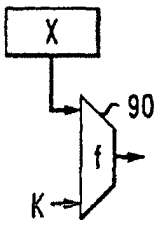

X
90
f
K

FIG. 10

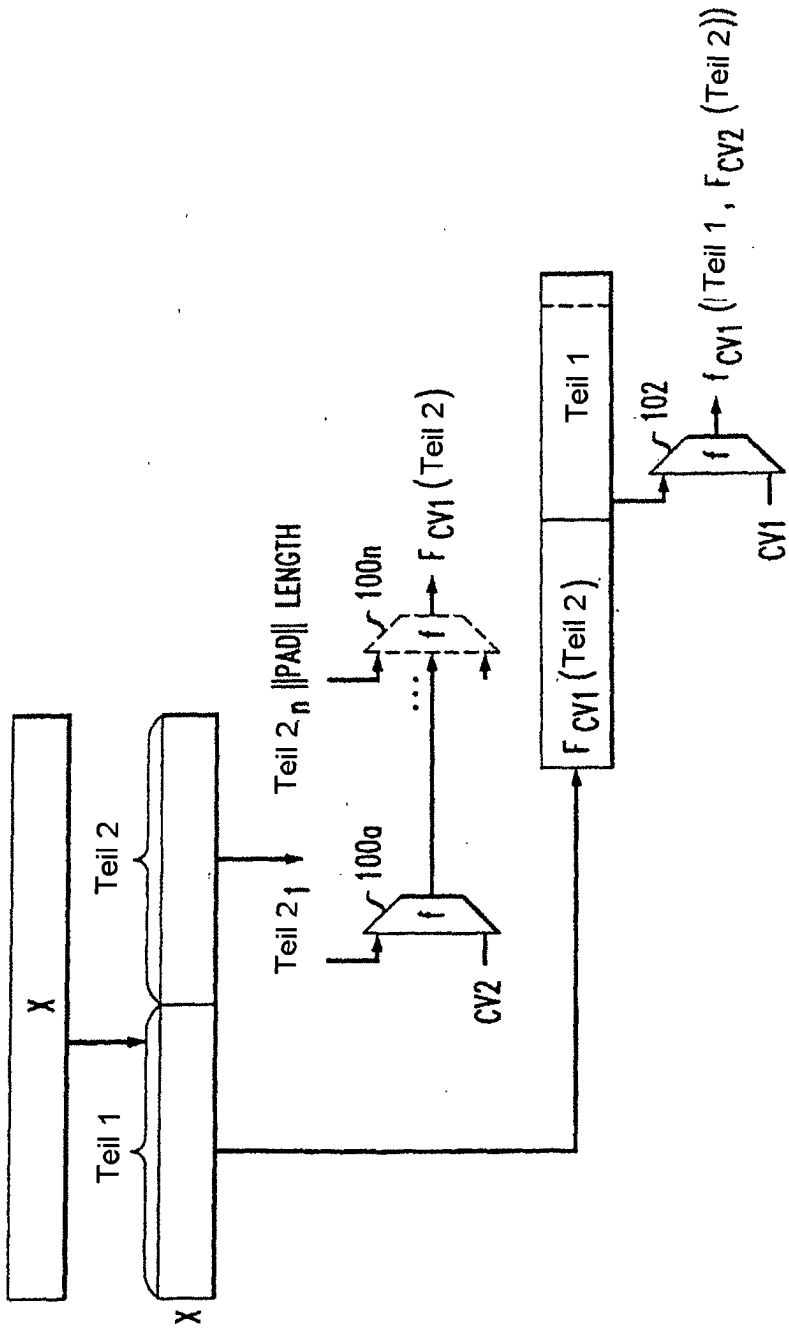

X
Teil 1
Teil 2
X
Teil 2 1
Teil 2 n ||PAD|| LENGTH
100a
100n
CV2
F CV1 (Teil 2)
F CV1 (Teil 2)
Teil 1
102
CV1
f CV1 (f Teil 1 , f CV2 (Teil 2))

FIG. 11a

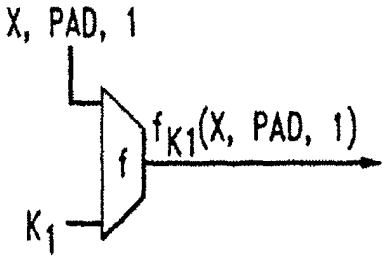


FIG. 11b

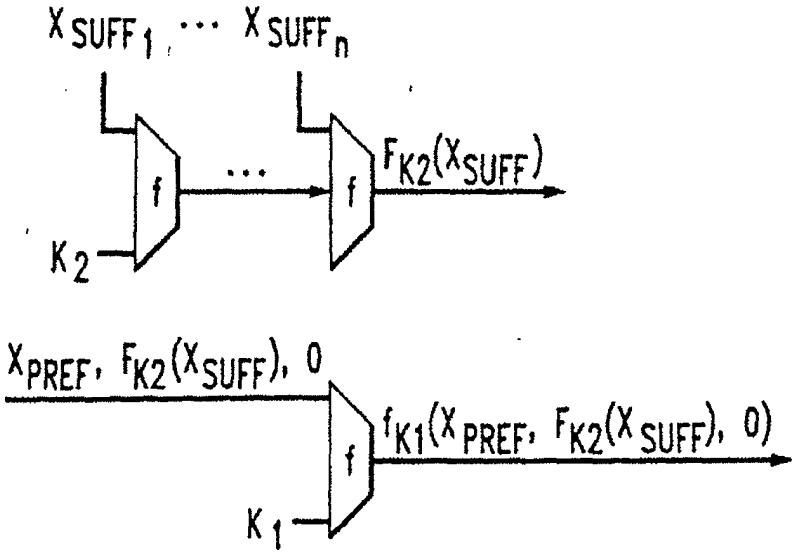


FIG. 12

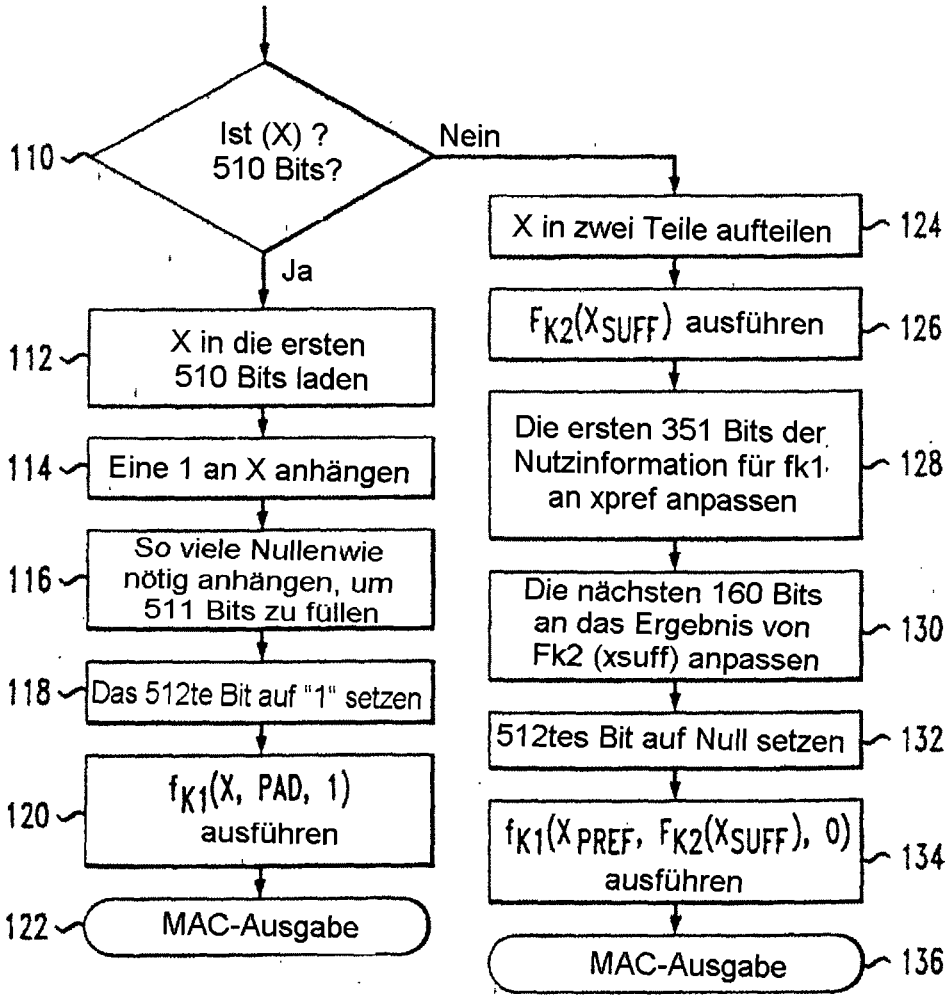


FIG. 13a

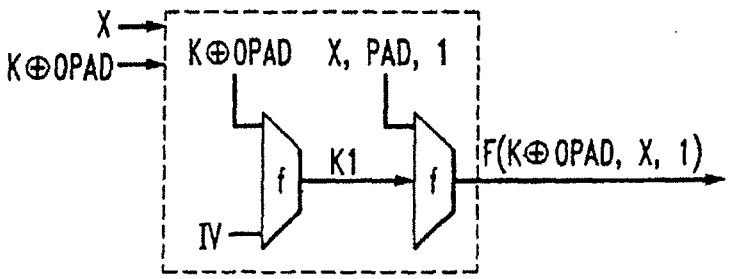


FIG. 13b

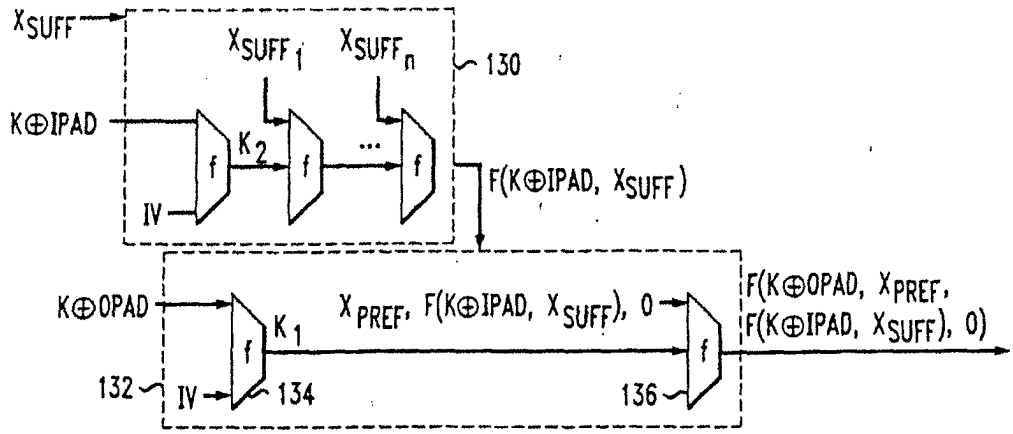


FIG. 14a

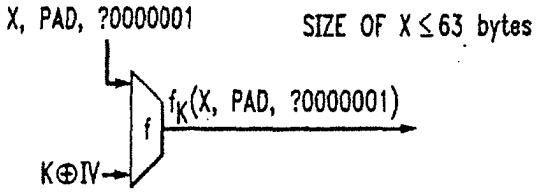


FIG. 14b

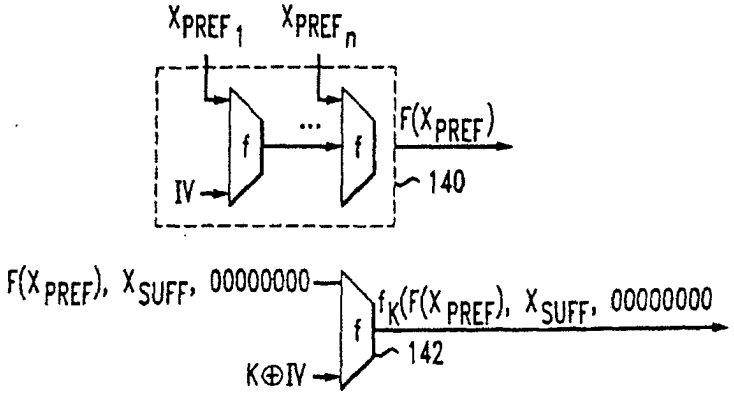
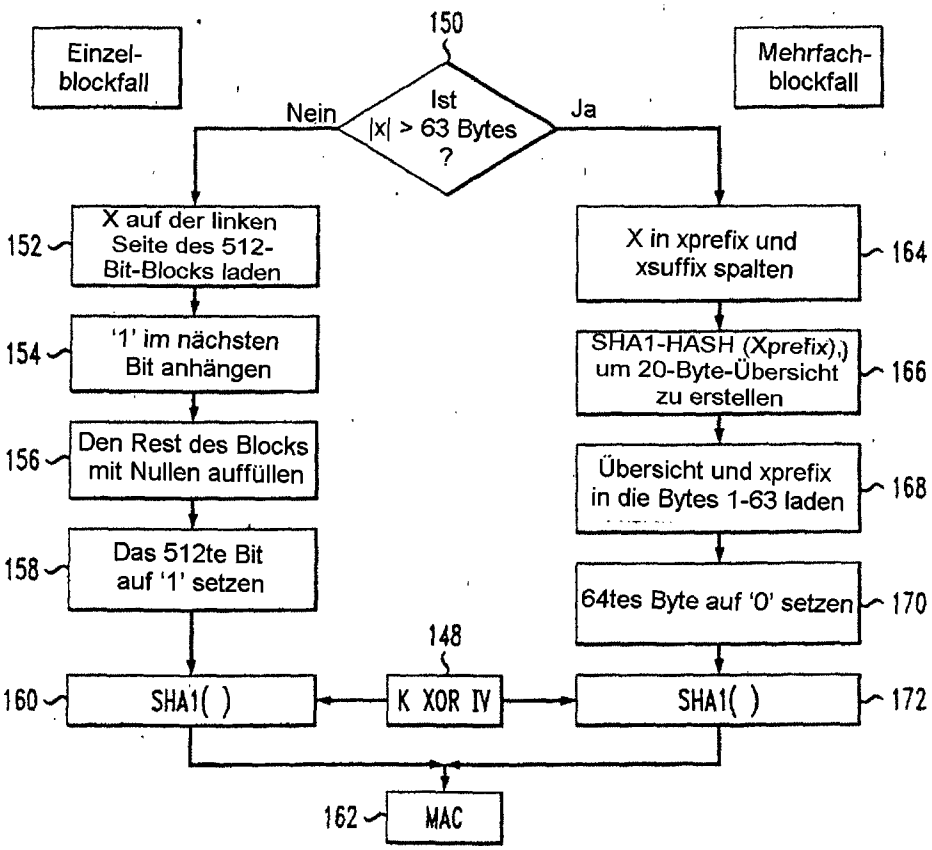


FIG. 15


Einzel-blockfall
150
Ist |x| > 63 Bytes ?
Nein
Ja
Mehrfach-blockfall
152
X auf der linken Seite des 512-Bit-Blocks laden
154
'1' im nächsten Bit anhängen
156
Den Rest des Blocks mit Nullen auffüllen
158
Das 512te Bit auf '1' setzen
X in xprefix und xsuffix spalten
164
SHA1-HASH (Xprefix),) um 20-Byte-Übersicht zu erstellen
166
Übersicht und xprefix in die Bytes 1-63 laden
168
64tes Byte auf '0' setzen
170
148
160
SHA1( )
K XOR IV
SHA1( )
172
162
MAC