(19) Republik Österreich Patentamt

(11) Nummer: AT 004 261 U2

(12) GEBRAUCHSMUSTERSCHRIFT

(21) Anmeldenummer: 865/00

(22) Anmeldetag: 16. 6.1998

(42) Beginn der Schutzdauer: 15. 3.2001
Längste mögliche Dauer: 30. 6.2008

(45) Ausgabetag: 25. 4.2001

(51) Int.Cl.[7] : G06F 15/00
G06F 17/30

(60) Abzweigung aus EP 98934143

(30) Priorität:

17. 6.1997 US 049853 beansprucht.

(73) Gebrauchsmusterinhaber:

PURDUE PHARMA LP
06856 STAMFORD (US).

(54) SICH SELBST ZERSTÖRENDES DOKUMENT ODER E-MAILNACHRICHTENSYSTEM

(57) Sich selbst zerstörendes Dokument oder E-Mailnachrichtensystem, das automatisch Dokumente oder E-Mailnachrichten zu einem vorher festgelegten Zeitpunkt zerstört, indem ein Makro oder ein Virus dem Dokument oder der E-Mailnachricht angefügt wird. Ein Makro wird erzeugt (220) und der Datei, einer E-Mailnachricht (280) oder einem Dokument (270), angefügt (230), wenn sie erzeugt wird. Das Makro enthält ausführbaren Code oder ein ausführbares Programm, das den Computer anweist, die Datei, an die das Virus angefügt ist, zu einer gewünschten Zeit zu überschreiben und/oder zu löschen.

AT 004 261 U2

Die vorliegende Erfindung betrifft das Gebiet der elektronisch erzeugten Dokumente, elektronische Nachrichten ("E-Mail") miteinbezogen, und das Feld der Verwahrung und Löschung von Dokumenten.

E-Mailnachrichtensysteme werden von Unternehmen oder privat weltweit rege genutzt. E-Mailsysteme ermöglichen es einem Benutzer, elektronische Nachrichten zu anderen Benutzern zu senden und elektronische Nachrichten von anderen Benutzern zu erhalten. Ein E-Mailsystem kann derart konfiguriert werden, daß den Benutzern, die an ein Local Area Network (LAN) angeschlossen sind, Nachrichtendienste zur Verfügung gestellt werden, und daß es den Benutzern ermöglicht wird, Nachrichten zu/von anderen Benutzern außerhalb des LAN über ein externes Netzwerk wie das Internet zu senden/zu empfangen. E-Mailsysteme ermöglichen es den Benutzern auch, Nachrichten über das E-Mailsystem zu speichern, zu kopieren und weiterzuleiten.

Die Leichtigkeit, mit der es ein E-Mailsystem den Benutzern erlaubt, Nachrichten zu speichern, zu kopieren und weiterzuleiten, führt zu der ungewollten Konsequenz, daß es schwierig wird, Dokumente zu verwahren.

Typischerweise wird das Verfahren der Dokumentenverwahrung von einem Unternehmen implementiert, um sicherzustellen, daß Dokumente, die von dem Unternehmen erzeugt oder

empfangen wurden, für einen bestimmten Zeitraum verwahrt und dann zerstört werden. Des weiteren können Zeiträume, über die Dokumente verwahrt werden sollen, für verschiedene Dokumente unterschiedlich sein. Eines der Ziele des Verfahrens der Dokumentenverwahrung ist es, ein systematisches Verfahren zum Verwahren und Zerstören von Dokumenten zu liefern, so daß das Unternehmen mit einiger Gewißheit sagen kann, welche Klassen von Dokumenten eines bestimmten Zeitraums noch existieren.

Ein effektives Verfahren der Dokumentenverwahrung ist auch wichtig im Zusammenhang mit Rechtsstreitigkeiten. Hier ist es wichtig, daß ein Unternehmen ein konsistentes Verfahren zur Verwahrung und Zerstörung von Dokumenten hat, und daß das Unternehmen dieses Verfahren implementiert. Zum Beispiel, falls das Verfahren der Dokumentenverwahrung eines Unternehmens festlegt, daß Briefe drei Jahre lang verwahrt werden, und falls es diesem Verfahren konsistent folgt, wird es der Anklage, daß ein bestimmter fünf Jahre alter Brief zerstört wurde, weil er im Rechtsstreit das Unternehmen belastet hätte, weniger verwundbar gegenüber stehen. Des weiteren ist es möglich, daß übriggebliebene Dokumente ein unvollständiges oder inakkurates Bild eines Vorgangs widerspiegeln, der in dieser Zeit ablief, falls dem Verfahren der Dokumentenverwahrung nicht konsistent gefolgt wird und nur einige Dokumente eines bestimmten Zeitraums vernichtet werden.

Systeme, die das Verfahren der Dokumentenverwahrung durchsetzen, sind wohl bekannt. Typischerweise suchen sie die Dateien eines Netzwerks periodisch ab und löschen Dateien, die vor einer bestimmten Zeit erzeugt wurden. Da solche Systeme auf einem Netzwerk laufen, können sie keine Dateien löschen, die auf den Festplatten individueller Computer gespeichert sind. Um diese Dateien ansprechen zu können, muß ein Programm zur Dokumenten-verwahrung auf jedem Computer installiert werden. Allerdings selbst wenn das Programm zur Dokumentenverwahrung auf jedem Computer installiert wird, wird es dem System immer noch unmöglich sein, Dateien, die auf einer Diskette oder einem anderen externen Medium gespeichert sind, zu löschen.

Auch können diese Systeme keine Dateien löschen, die via E-Mail zu anderen Computern außerhalb des Netzwerks geschickt wurden. Demnach werden Dateien, die auf den Heimcomputer oder das Laptop eines Mitarbeiters oder einer dritten Partei transferiert wurden, von einem konventionellen System zur Dokumentenverwahrung nicht gelöscht.

Gemäß einer ersten Ausführungsform der Erfindung wird ein sich selbst zerstörendes Dokumenten- und E-Mailsystem bereitgestellt, das Dokumente oder E-Mails zu einem vorbestimmten Zeitpunkt automatisch zerstört, indem ein ausführbares Modul wie ein "Virus" an das Dokument oder das E-Mail angehängt wird.

Computerviren sind wohl bekannt. Im allgemeinen ist ein Computervirus ein Stück ausführbarer Code oder Programm, das sich an eine Gast-Datei anhängt. Zum Beispiel operieren „angehängte Viren" derart, daß sie sich ans Ende des Gastprogramms anhängen und das Programm zwingen, den Virencode vor Ausführung des Gastprogramms auszuführen. "Voranstehende Viren" hingegen hängen sich an den Anfang eines Gastprogramms. Andere Virenarten befinden sich in anderen Teilen des Gastprogramms. Andere Virenklassen sind als Makroviren bekannt. Diese Viren sind Makros, die in Textdokumenten eingebettet sind, und so konfiguriert werden können, daß sie ausgeführt werden, wenn das Dokument geöffnet, erstellt oder gespeichert wird. Typischerweise wird ein Virus, der in seiner Gastdatei oder in seinem Gastprogramm bleibt und nicht auf andere Dateien überspringt, als Trojanisches Pferd bezeichnet.

In Übereinstimmung mit der ersten Ausführungsform der Erfindung wird ein ausführbares Modul in Form eines Trojanischen Pferdes an eine Datei gehängt (wie etwa eine E-Mail oder ein Dokument), wenn sie erzeugt wird. Das ausführbare Modul enthält einen Teil ausführbaren Code oder ein ausführbares Programm, das den Computer anweist, die Datei, an die es angehängt ist, zu einem bestimmten Zeitpunkt zu überschreiben und/oder zu löschen. Auf diese Weise wird die Lebenszeit des Dokuments oder der E-Mail vom ausführbaren Modul limitiert. Da das ausführbare Modul mit der Datei verbunden ist, wird es mit ihr mitwandern, selbst falls sie kopiert, weitergeschickt oder auf Disketten oder Bändern gesichert werden sollte.

In Übereinstimmung mit einem weiteren Aspekt der ersten Ausführungsform der Erfindung wird das ausführbare Modul jedesmal ausgeführt, wenn die Datei, an die es angeheftet ist, geöffnet wird. Das ausführbare Modul stellt fest, ob die Voraussetzung zur Dateizerstörung gegeben ist. Falls diese Voraussetzung gegeben ist, instruiert das ausführbare Modul den

Computer, die Datei mit aussagelosen Zeichen zu überschreiben und dann zu sichern, zu schließen oder zu löschen.

Auf diese Weise liefert die gegenwärtige Erfindung eine Reihe von Vorzügen gegenüber bekannten Systemen. Zum Beispiel kann das System in Übereinstimmung mit der Erfindung das Verfahren der Dokumentenverwahrung erzwingen, selbst wenn die Datei auf einem externen Datenträger, wie einer Diskette oder einem Computer, der nicht in das Netzwerk eingebunden ist, gespeichert wurde, da das ausführbare Modul an die Datei angehängt ist und ausgeführt wird, wenn die Datei geöffnet wird. Dies liegt daran, daß das ausführbare Modul mit der Datei mitwandert und ausgeführt wird, wann immer und wo immer die Datei geöffnet wird.

In Übereinstimmung mit einer zweiten Ausführungsform der Erfindung wird ein sich selbst zerstörendes E-Mailnachrichtensystem bereitgestellt, das automatisch ein ausführbares Modul an jede erstellte E-Mail oder jeden erstellten E-Mail Anhang anhängt. Das ausführbare Modul enthält einen Teil ausführbaren Code oder ein ausführbares Programm, das den Computer veranlaßt, die Nachricht (und/oder den Anhang), an die das ausführbare Modul angehängt ist, zu einem bestimmten gewünschten Zeitpunkt zu überschreiben und/oder zu löschen. Das ausführbare Modul reist mit der Nachricht mit, wenn die Nachricht an ihre Adresse verschickt wird, da es an ihr angehängt ist. Des weiteren bleibt das ausführbare Modul mit der Nachricht verbunden, selbst wenn die Nachricht kopiert, an eine andere Adresse weitergeleitet oder auf Disketten oder externe Speichermedien gespeichert wird.

In Übereinstimmung mit einer dritten Ausführungsform der Erfindung schließt ein Dokumentensicherheitssystem einen virtuellen Behälter mit ein, in den ein oder mehrere digitale Objekte "abgelegt" werden. In diesem Zusammenhang wird der Begriff "digitales Objekt" umfassend verwendet und schließt Dokumente (wie Tabellen, Kurven/Grafiken, Schriftdokumente, ASCII Textdateien, Bilder und andere Dateien), Programme und alles andere, was mit einem Computer gespeichert werden kann, ein.

Das Dokumenten-Sicherheitssystem enthält einen Behälter-Erzeuger und einen Behälter-Öffner. Der Behälter-Erzeuger und der Behälter-Öffner sind in ein oder mehrere Softwareprogramme implementiert, die von einem Computer ausgeführt werden. Der

Behälter-Öffner ermöglicht es dem Benutzer, ein Limit für die gültige Lebenszeit eines digitalen Objekts im Behälter zu setzen. Diese "Lebenszeitkontrollen" können zum Beispiel ein Ablaufdatum, ein Ablaufdatum und eine Ablaufzeit, eine bestimmte Anzahl, die das Dokument geöffnet werden darf, oder andere Limitierungen enthalten. Wenn der Behälter vom Behälter-Öffner geöffnet wird, überprüft der Behälter-Öffner die Lebenszeitkontrollen. Wenn sie gültig sind, sind die digitalen Objekte im Behälter zum Zugriff durch den Benutzer freigegeben. Wenn allerdings die Lebenszeitkontrollen ungültig sind, wird der Behälter-Öffner das digitale Objekt umgehend zerstören.

Ein Vorteil des Dokumenten-Sicherheitssystems mit Behälter-Öffner und Behälter-Erzeuger besteht darin, daß es in manchen Umgebungen für eine stabilere Durchsetzung der Lebenszeitkontrollen sorgt als ein sich selbst zerstörendes Dokument, das ein ausführbares Modul enthält. Das liegt daran, daß die Fähigkeit gegenwärtiger Textverarbeitungsprogramme und Tabellenkalkulationsprogramme, ein ausführbares Modul auszuführen, von Produkt zu Produkt stark variiert. Zum Beispiel wird sich ein sich selbst zerstörendes Dokument, das ein ausführbares Modul enthält, das von Microsoft Word™ ausgeführt wird, nicht selbst zerstören, wenn es zum Beispiel als einfache ASCII Datei geöffnet wird, die nicht darauf ausgelegt ist, ein ausführbares Modul zu erkennen und auszuführen. Auf der anderen Seite sind die Dokumente mit dem Dokumenten-Sicherheitssystem in einem Behälter gesichert, der vom Behälter-Öffner geöffnet wird, und der Behälter-Öffner kann, wenn er eingesetzt wird, Lebenszeitkontrollen gegen jedes Dokument im Behälter durchsetzen, ganz egal welchen Typs es ist.

Gemäß einem weiteren Aspekt des Dokumenten-Sicherheitssystems und gemäß einer dritten Ausführungsform der gegenwärtigen Erfindung kann jedes digitale Objekt im Behälter unabhängige Lebenszeitkontrollen besitzen. Gemäß diesem Merkmal kann ein einzelnes Objekt im Behälter, wenn es ungültig wird, zerstört werden, während die anderen Objekte intakt bleiben.

Gemäß einer weiteren Ausführungsform des Dokumenten-Sicherheitssystems handelt es sich bei dem digitalen Objekt um ein sich selbst zerstörendes Dokument, das gemäß der ersten oder zweiten Ausführungsform der Erfindung erzeugt wurde.

Gemäß einem weiteren Aspekt dieser Ausführungsform kann der Behälter, sein Inhalt und seine Lebenszeitkontrollen gegen einen Benutzer, der das Sicherheitssystem untergraben möchte, gesichert werden. Diese Sicherheit wird durch den Einsatz von Verschlüsselungstechnologie erreicht. Genauer gesagt wird der Behälter-Erzeuger so konfiguriert, daß er die Objekte im Behälter verschlüsselt und der Behälter-Öffner, daß er sie entschlüsselt. Auf diese Weise ist das Objekt unlesbar, falls ein Benutzer das Objekt öffnet, ohne den Behälter-Öffner zu benutzen.

Gemäß einem weiteren Aspekt der ersten und zweiten Ausführungsform ist das sich selbst zerstörende Dokument oder die E-Mail entweder vom ausführbaren Modul oder einer anderen Anwendung verschlüsselt und das ausführbare Modul so konfiguriert, daß es das Dokument oder die E-Mail nur entschlüsselt, falls die Lebenszeit des Dokuments oder der E-Mail noch nicht abgelaufen ist.

Gemäß einer weiteren Ausführungsform der gegenwärtigen Erfindung wird ein sich selbst zerstörendes Dokument erzeugt, indem mehrere ausführbare Module in das Dokument oder die E-Mail eingebettet werden, wobei jedes Modul von einem anderen Textverarbeitungs- oder E-Mailsystem ausgeführt werden kann. Zum Beispiel kann ein Dokument ein erstes Modul, das von einem ersten System ausgeführt werden kann, und ein zweites Modul, das von einem zweiten System ausgeführt werden kann, enthalten. Das Dokument selbst kann verwandt sein mit beiden Systemen. Gemäß dieser Ausführungsform werden die Lebenszeitkontrollen erzwungen, unabhängig davon, ob es vom ersten oder vom zweiten System geöffnet wurde.

Gemäß einer weiteren Ausführungsform der gegenwärtigen Erfindung wird ein Internet-Handelssystem zur Verfügung gestellt, das die virtuellen Behälter einsetzt. Gemäß dieser Ausführungsform plaziert eine Partei, die ein elektronisch übertragbares Produkt über das Internet verkaufen will, dieses Produkt in einem virtuellen Behälter, wobei sie eine Behälter-Erzeuger-Anwendung benutzt, die das Produkt verschlüsselt und Lebenszeitkontrollen für das Produkt setzt. Ein potentieller Käufer, der das Produkt begutachten möchte, bevor er es kauft, erhält eine Kopie des Behälters zusammen mit der Behälter-Öffner-Anwendung vom Verkäufer. Die Behälter-Öffner-Anwendung ermöglicht es dem potentiellen Käufer, das Produkt anzusehen und auszuprobieren bis zu einem vorherbestimmten Ablaufdatum. In

dieser Ausführungsform erlaubt es der Behälter-Öffner dem Benutzer nicht, das Produkt vom virtuellen Behälter zu entfernen. Gemäß einer bevorzugten Ausführungsform wird das Produkt vom Behälter-Öffner gelöscht, wenn der Behälter nach dem Ablaufdatum geöffnet wird. Gemäß noch einer weiteren Ausführungsform der Erfindung wird der Verkäufer, wenn der Käufer für das Produkt vor Ablauf des Ablaufdatums bezahlt hat, dem Käufer einen einzigartigen Schlüssel übermitteln. Der Behälter-Öffner kann derart konfiguriert werden, daß er in diesem Fall das Produkt aus dem Behälter entläßt.

Die Erfindung wird nachfolgend anhand eines Ausführungsbeispiels unter Bezugnahme auf die Zeichnungen näher erläutert.

Fig. 1 zeigt eine Umgebung gemäß dem heutigen Stand der Technik, in welche die Erfindung implementiert werden kann.

Fig. 2 (a) und 2(b) zeigen E-Mails gemäß der gegenwärtigen Erfindung, die vorne und hinten angehängte Viren enthalten.

Fig. 3 zeigt ein Flußdiagramm zur Veranschaulichung, wie ein sich selbst zerstörendes E-Mail gemäß einer Ausführungsform der gegenwärtigen Erfindung erstellt werden kann.

Fig. 4 zeigt ein Flußdiagramm eines Makro-Virus, der in das sich selbst zerstörende E-Mail aus Fig. 3 eingebettet ist.

Fig. 5(a) bis 5(c) zeigen eine graphische Benutzerschnittstelle für ein sich selbst zerstörendes Dokument gemäß einer Ausführungsform der Erfindung, die für ein Microsoft Excel™ Dokument implementiert wird.

Fig. 6(a) bis 6(c) zeigen eine graphische Benutzerschnittstelle für ein sich selbst zerstörendes Dokument gemäß einer Ausführungsform der Erfindung, die für ein Microsoft Word™ Dokument implementiert wird.

Fig. 7(a) bis 7(e) zeigen eine graphische Benutzerschnittstelle für ein sich selbst zerstörendes Dokument gemäß einer Ausführungsform der Erfindung, die in Microsoft Outlook™ implementiert wird.

Fig. 8(a) zeigt einen virtuellen Behälter eines Dokumentensatzes gemäß einer Ausführungsform der Erfindung.

Fig. 8(b) zeigt einen Vertreter eines virtuellen Behälters gemäß einer Ausführungsform der Erfindung.

Fig. 9(a) und 9(b) zeigen eine weitere Ausführungsform des virtuellen Behälters eines Dokumentensatzes der gegenwärtigen Erfindung.

Fig. 10(a) zeigt ein Kopfzeilen-Format für einen Behälters eines Dokumentensatzes, konfiguriert, um ein einzelnes Dokument zu enthalten.

Fig. 10(b) zeigt ein Kopfzeilen-Format für einen Behälters eines Dokumentensatzes, konfiguriert, um mehrere Dokumente zu enthalten.

Fig. 11 zeigt eine graphische Benutzerschnittstelle für eine Behälter-Erzeuger-Anwendung und eine Behälter-Öffner-Anwendung.

Fig. 12 zeigt ein Flußdiagramm zum Hinzufügen eines Dokuments zum Behälters eines Dokumentensatzes in Figur 10(b).

Fig. 13 zeigt ein Flußdiagramm zum Entnehmen eines Dokuments aus dem Behälters eines Dokumentensatzes in Figur 10(b).

Figur 1 zeigt eine Umgebung gemäß dem heutigen Stand der Technik, in welche die Erfindung implementiert werden kann. Ein lokales Netzwerk 1 (LAN 1) schließt eine Vielzahl von Bürocomputern 10. – 10.6 (im folgenden als Computer 10 bezeichnet) und einen Server 20 ein. Jeder Computer 10 beinhaltet einen primären Datenspeicher 12 (z. B. eine Festplatte) und einen sekundären Datenspeicher 14 (z. B. eine Floppy Disk oder ein CD

ROM-Laufwerk). Der Server 20 besitzt ebenfalls einen primären Netzwerkspeicher 22 (z. B. eine Festplatte) und einen sekundären Datenspeicher 14 (z. B. Band- oder CD ROM-Laufwerk). Die Daten auf den primären oder sekundären Netzwerkspeichermedien 22, 24 können von allen Computern 10 benutzt werden. Im Gegensatz dazu sind die Daten auf den primären und sekundären Speichermedien 12, 14 jedes einzelnen Computers 10 privat und können nur von dem betreffenden Computer benutzt werden. Der Server 20 liefert Internetzugang auf Computer außerhalb des Netzwerks, wie zum Beispiel ein Heimcomputer 40, mittels Datenbahnen 50. Der Heimcomputer 40 beinhaltet ein primäres Speichermedium 42 und ein sekundäres Speichermedium 44. Das LAN 1 unterstützt einen elektronischen Mail-Nachrichtenservice, der es jedem Computer 10 erlaubt, Nachrichten innerhalb des LANs 1 zu anderen Computern 10 zu senden und Nachrichten außerhalb des LANs 1 zu Computern außerhalb des Netzwerks zu senden, wie zum Beispiel Heimcomputer 40. Die Konfiguration, wie sie in Figur 1 gezeigt wird, ist illustrativ für typische LANs, die oft in Unternehmen Anwendung finden. Dem Fachmann ist allerdings klar, daß die gegenwärtige Erfindung in eine ganze Anzahl von Netzwerkkonfigurationen implementiert werden kann. Des weiteren kann das E-Mailnachrichtensystem der gegenwärtigen Erfindung auch auf Applikationen außerhalb eines Netzwerks angewendet werden, bei denen zum Beispiel E-Mailnachrichten von einem alleinstehenden Computer via Modem oder einer anderen Verbindung gesendet und empfangen werden. Des weiteren können die Dokument-Verwahrungs- und Löschungseinrichtungen der gegenwärtigen Erfindung in einen alleinstehenden Computer implementiert werden, mit oder ohne eine Verbindung zu externen Computern.

Einige der Dokument-Verwahrungs- und Zerstörungsprobleme können gemäß der gegenwärtigen Erfindung verringert werden und werden im folgenden unter Bezugnahme auf Figur 1 erläutert.

Zu Illustrationszwecken nehme man an, daß das lokale Netzwerk 1 in einem Unternehmen installiert ist, das ein Dokument-Verwahrungsverfahren adoptiert hat, das fordert, daß Dokumente, die älter sind als zwei Jahre, zerstört werden. Typischerweise wird ein solches Dokument-Verwahrungsverfahren implementiert, indem ein Programm auf dem Server 20 installiert wird, das periodisch die primären Speichermedien 22 des Netzwerks überprüft und Dokumente, die vor über zwei Jahren vor der Überprüfung erstellt wurden, löscht. Ein

Problem, das bei Programmen dieser Art auftritt, ist, daß Kopien der Dokumente, die nicht im Netzwerk gespeichert sind, nicht erfaßt werden.

Zum Beispiel nehme man an, daß Benutzer-1 ein Dokument auf einem Computer 10-1 erzeugt, und es auf dem Netzwerk-Datenspeichermedium 22 speichert. Dann kopiert Benutzer-1 das Dokument eine Woche später auf die Festplatte 12-1 des Computers 10-1 und auf eine Diskette mittels des sekundären Speichermediums 14.1. Des weiteren kann Benutzer-2, da das Dokument auf dem primären Netzwerkspeichermedium 22 gespeichert ist, ohne Kenntnis von Benutzer-1 auf ein primäres Datenspeichermedium 12.2 von Computer 10-2 und dann auf eine andere Diskette mittels des sekundären Speichermediums 14-2 kopieren. Auf diese Weise wurden fünf Kopien desselben Dokuments erstellt, von denen nur eine vom Netzwerk-Dokumenten-Verwahrungsprogramm angesprochen und gelöscht werden kann.

Das oben angesprochene Problem wird weiter verkompliziert, da das LAN aus Figur 1 elektronischen Nachrichtenservice zu Verfügung stellt. Das bedeutet, daß Benutzer-1 ein E-Mail erzeugen kann und es mittels Internet-Server 30 außerhalb des LANs zu Computer 40 senden kann. Des weiteren können kürzlich erzeugte Dokumentdateien als Attachment dem E-Mail angehängt werden. Dies führt zu weiteren Problemen. Zum Beispiel können diese E-Mails und ihre Attachments außerhalb des LANs zu dritten Parteien gesendet werden, die nichts von dem Dokumenten-Verwahrungsverfahren des Unternehmens wissen. Jedenfalls wird der Heimcomputer eines Angestellten, falls das E-Mail dorthin geschickt wurde, nicht vom Netzwerk-Dokumenten-Verwahrungsprogramm angesprochen.

Typischerweise bedingt eine Durchsetzung des Dokumenten-Verwahrungsverfahrens bei Dokumenten, die nicht in Netzwerk-Speichermedien gespeichert sind, daß jeder Angestellte jedes seiner Verzeichnisse seiner individuellen Festplatte und seiner Disketten und anderen sekundären Speichermedien durchsucht. Des weiteren können ältere Dokumente in neuen E-Mails versteckt sein, aufgrund der Fähigkeit eines E-Mailsystems, Dokumente als Attachment an E-Mails anzuhängen, was es schwierig macht, diese Dokumente zu identifizieren und zu löschen. Die Angestellten müssten auch angewiesen werden, ihre Laptops und Heimcomputer nach Unternehmensdateien zu durchsuchen. Selbst falls jeder Angestellte diesen

zeitaufwendigen Verpflichtungen nachkommt, bleibt die Möglichkeit bestehen, daß Dokumente auf den Rechnern dritter Parteien weiter existieren.

Ein weiteres Problem der Implementierung eines Dokumenten-Verwahrungsverfahrens liegt darin, daß, falls ein Dokument von einem Speichermedium gelöscht wird, das Dokument an sich nicht entfernt wird. Statt dessen entfernt der Computer lediglich den Zeiger auf das Dokument, der den Platz im Speichermedium anzeigt. Das Dokument selbst bleibt im Speichermedium gespeichert, bis der Computer es mit einem anderen Dokument überschreibt. Daher müssen Angestellte konsequenterweise, um sicherzustellen, daß ein Dokument tatsächlich gelöscht wird, es mit aussagelosen Zeichen überschreiben und es anschließend löschen, was den Zeitrahmen für die Durchsetzung des Dokumenten-Verwahrungsverfahrens weiter vergrößert.

Ausführbare Module in ein Dokument oder eine E-Mailnachricht integriert

Gemäß der gegenwärtigen Erfindung wird ein sich selbst zerstörendes Dokumentensystem zu Verfügung gestellt, das ein ausführbares Modul an ein Dokument anhängt. Gemäß einer anderen Ausführungsform der Erfindung wird ein sich selbst zerstörendes E-Mailnachrichtensystem zu Verfügung gestellt, das ein ausführbares Modul an eine elektronische Nachricht oder an einen Nachrichtenanhang anhängt, wenn die Nachricht oder der Nachrichtenanhang erstellt wird. Das ausführbare Modul kann an die Nachricht oder das Dokument auf jede bekannte Weise angehängt werden. Zum Beispiel kann das ausführbare Modul, gemäß Figur 2, als voranstehender Virus (Fig. 2A), als nachstehender Virus (Fig. 2B) oder in jeder anderen bekannten Art angehängt werden. Das Modul selbst kann in jeder bekannten Art erstellt werden, indem zum Beispiel Assembly-Sprache, Programmiersprachen höherer Ebene oder Makros eingesetzt werden. Das angehängte ausführbare Modul bewirkt die Selbstzerstörung des Dokuments oder der Nachricht, wenn sich eine vorherbestimmte Situation einstellt. Auf diese Weise erzeugt das Modul eine "Lebenszeitkontrolle", die die Lebenszeit des Dokuments kontrolliert. In der oben angesprochenen Illustration zum Beispiel ist das ausführbare Modul derart programmiert, daß es das Dokument oder die Nachricht, an die es angehängt ist, löscht, wenn das Erzeugungsdatum des Dokuments oder der Nachricht bezüglich dem gegenwärtigen Datum mehr als zwei Jahre zurückliegt. Gemäß einer weiteren Ausführungsform der gegenwärtigen Erfindung überschreibt das ausführbare Modul das

Dokument oder die Nachricht, an die es angehängt ist, mit aussagelosen Zeichen (z. B. "X"), bevor das Dokument oder die Nachricht gelöscht wird.

Gemäß weiteren Ausführungsformen der Erfindung kann das Überschreiben und/oder Löschen des Dokuments oder der Nachricht vom Auftreten anderer oder weiterer Voraussetzungen abhängen. Zum Beispiel kann das System konfiguriert werden, daß die Nachricht oder das Dokument überschrieben und/oder gelöscht wird, wenn das Dokument i) geschlossen oder ii) zum zweiten Mal geöffnet wird, so daß ein Dokument erstellt wird, das nur ein einziges Mal gelesen werden kann, bevor es vernichtet wird. Alternativ kann das System so konfiguriert werden, daß die Nachricht oder das Dokument überschrieben und/oder gelöscht wird, wenn es kopiert, weitergeleitet oder gedruckt werden soll. Des weiteren kann das System konfiguriert werden, daß jeder Versuch, das Dokument oder die Nachricht zu kopieren, unterbunden wird, so daß ein Dokument erzeugt wird, das gelesen, aber nicht gedruckt werden kann. Diese zusätzlichen Lebenszeitkontrollen können auch von einem bestimmten Ereignis abhängen, so daß der Druck- oder Kopierschutz aufgehoben wird, wenn ein Paßwort oder ein Schlüssel eingegeben wird, oder daß der Druck- oder Kopierschutz eingeschaltet wird, wenn eine bestimmte Zeit abgelaufen oder ein anderes Ereignis eingetreten ist.

Gemäß einer anderen Ausführungsform der Erfindung kann das System einem LAN-Administrator ermöglichen, die Überschreib-/Löschvoraussetzung(en) vorherzubestimmen, um allen Systembenutzern und allen Dokumenten und Nachrichten, die von den Systembenutzern erzeugt werden, ein Dokumenten-Verwahrungsverfahren aufzuerlegen. Genauso kann das System konfiguriert werden, daß es jedem Benutzer erlaubt ist, zu wählen, ob ein bestimmtes Dokument oder eine Nachricht eine Überschreib-/Löschvoraussetzung erhalten soll. Des weiteren kann dem Benutzer erlaubt werden, aus einer Vielzahl von Überschreib-/Löschvoraussetzungen zu wählen. Dies kann zum Beispiel durch ein Tool-Menü implementiert werden.

Im folgenden soll eine exemplarische Software zur Implementierung eines sich selbst zerstörenden Dokuments oder zur Bereitstellung eines sich selbst zerstörenden elektronischen Nachrichtensystems beschrieben werden. Obwohl die zur Verdeutlichung aufgeführten Ausführungsbeispiele für Microsoft Word™-, Microsoft Excel™- und Microsoft Outlook™-

Umgebungen ausgelegt sind, wird der Fachmann erkennen, daß die gegenwärtige Erfindung in eine Vielzahl von Umgebungen in einer Vielzahl von Arten implementiert werden kann.

Figuren 3 und 4 zeigen die Implementierung eines sich selbst zerstörenden elektronischen Nachrichtensystems für Microsoft Word™-6.0 Dokumente. In Bezugnahme auf Figur 3 initiiert ein Benutzer in Schritt 200 die Erzeugung eines Dokuments oder einer Nachricht auf, zum Beispiel, einem Bürocomputer 10.1 in Figur 1. In Schritt 220 erzeugt das System ein "AutoOpen" Makro in der "Normal.dot" Datei. Gemäß der Microsoft Word™-Architektur wird ein erzeugtes Makro mit dem Namen AutoOpen jedes Mal, wenn ein Word-Dokument geöffnet wird, ausgeführt. Die Instruktionen, auf die bei Schritt 220 hingewiesen wird und die in Word Basic™-Programmiersprache geschrieben sind, sind als AutoOpen-Makro in der Normal.dot-Template-Datei gespeichert. Normal.dot ist eine Schablone, die vom Word™-Programm als Speicherdatei für globale Makros eingesetzt wird. In Schritt 230 wird die Datei Normal.dot, die das AutoOpen-Makro enthält, auf eine Datei mit Namen "Message.dot" kopiert. Dann wird in Schritt 240 das AutoOpen-Makro aus der Datei Normal.dot gelöscht. In den Schritten 250-260 wird die Datei Message.dot geöffnet und der Benutzer aufgefordert, den Text des Dokuments oder der Nachricht einzufügen. Dann wird in den Schritten 270 die Datei Message.dot, die das Dokument oder die Nachricht als auch eine Kopie der Datei Normal.dot enthält, gesichert. Im Gegenzug enthält die Kopie der Datei Normal.dot das AutoOpen-Makro. In den Schritten 280-290 wird die Datei Message.dot umgenannt in Message.doc und dann als elektronische Mailnachricht oder als Anhang an eine elektronische Mailnachricht verschickt.

Die elektronische Mailnachricht wird zum Beispiel über das Internet zu einem Heimcomputer 40 mittels LAN Server 20 und Internet Servern 30 geschickt. Das AutoOpen-Makro, das in die Message.doc-Datei eingebettet ist, wird ausgeführt, wenn die Datei Message.doc vom Empfänger der Nachricht geöffnet wird. Figur 4 ist ein Flußdiagramm für das AutoOpen-Makro, das in die Message.doc-Datei eingebettet ist. In Schritt 310 gibt die Funktion DateSerial() einen seriellen Wert als die Variable "created" zurück, der für das Erstellungsdatum der Nachricht repräsentativ ist. Das Argument der Funktion DateSerial wird als die Zeit gesetzt, zu der das AutoOpen-Makro erstellt wurde (Schritte 210 und 220 in Figur 3), und wurde in Figuren 3 und 4 willkürlich auf den 10. Juni 1997 festgelegt. In Schritt 320 gibt die Funktion Today() einen seriellen Wert als die Variable "curdate" zurück,

der für das gegenwärtige Datum repräsentativ ist. Die seriellen Werte, die von diesen Funktionen zurückgegeben werden, sind ganze Zahlen zwischen 1 und 802074, wobei die Zahl 1 den 31. Dezember 1899 bedeutet und die Zahl 802074 den 31. Dezember 4095 bedeutet. In Schritt 330 wird die Variable "difference" als der Unterschied zwischen "curdate" und "created" festgelegt. Demnach stimmt die Variable "difference" mit der Anzahl von Tagen zwischen Erstellung der Datei Message.doc und dem gegenwärtigen Datum, an dem das Dokument geöffnet wurde, überein.

Schritte 340 bis 360 enthalten eine "If, Then" Aussage, die entscheidet, ob die Zeit zwischen Erstellungsdatum und gegenwärtigem Datum eine vorher bestimmte Schwelle überschritten hat. Zur Veranschaulichung wird diese Schwelle auf 60 Tage festgelegt. In Schritt 340 wird die Variable "difference" mit dem Wert 60 verglichen. Wenn "difference" größer als 60 ist, d. h. wenn mehr als 60 Tage seit der Erstellung der Datei Message.doc vergangen sind, werden die Schritte 350, 360 und 370 ausgeführt. In Schritt 350 wird eine EditReplace.Find-Funktion ausgeführt. Diese Funktion ersetzt jedes Zeichen der Datei Message.doc durch ein aussageloses Zeichen (das willkürlich als "X" festgelegt wurde). Dann wird die Datei Message.doc, die als eine Serie von "X" umgeschrieben wurde, wobei die Anzahl der "X" mit der Anzahl der Zeichen in dieser Datei übereinstimmt, in Schritt 360 gesichert und geschlossen. Auf diese Weise wurden die Speicherplätze, auf denen die Datei Message.doc gespeichert war, mit aussagelosen Zeichen überschrieben. Falls allerdings in Schritt 340 die Variable "difference" kleiner als oder gleich 60 ist, wird direkt zu Schritt 370 gesprungen und der Empfänger der Nachricht kann die Nachricht frei betrachten und bearbeiten.

Natürlich kann das Makro in Figur 4 modifiziert werden, um weitere Funktionen auszuführen. Zum Beispiel kann das Makro so modifiziert werden, daß die Datei Message.doc gelöscht wird, nachdem sie gesichert wurde, indem die Funktion FileNameFromWindow$() benutzt wird, um den Speicherplatz der Datei Message.doc zu dem Zeitpunkt, zu dem sie geöffnet wurde, zu erlangen, und dann die Funktion Kill benutzt wird, um die Datei zu löschen. Des weiteren kann das Makro dahingehend modifiziert werden, daß es dem Benutzer, der die Datei Message.doc öffnet, erlaubt, das Löschen der Nachricht zu gestatten.

Figuren 5(a) bis 5(c) und Tabelle 1 zeigen ein sich selbst zerstörendes Dokumentensystem unter Bezugnahme auf Microsoft Excel™-Dokumente. In Tabelle 1 wird ein Visual Basic™-

Programm gezeigt, um ein sich selbst zerstörendes Excel-Dokument zu erzeugen, das eine Benutzerschnittstelle enthält, die für eine Microsoft Windows™-Umgebung implementiert ist. Die gezeigte Implementierung nutzt die Routinen "auto_open" und "auto_close", die von Microsoft Excel™ unterstützt werden. Gemäß der gezeigten Ausführungsform sind diese Routinen in der Schablone "book.xlt" im Unterverzeichnis xlstart von Microsoft Excel™ gespeichert. Auf diese Weise wird book.xlt mit den auto_open und auto_close Routinen als Schablone für jedes Microsoft Excel™-Dokument, das erzeugt wird, benutzt. Daher sind diese Routinen selbst im Excel-Dokument angesiedelt.

Jedes Mal, wenn ein Excel-Dokument erzeugt wird, wird das Programm aus Tabelle 1 als Makro in das Dokument eingebettet. Wenn das Dokument geschlossen wird, wird die auto_close Routine implementiert. Nach Lesen der Variablen "lcdata" und "lcmode" aus den einzelnen Dokumenteigenschaften (Tabelle 1, Zeilen 23-26), überprüft das Programm, ob die Lebenszeitkontrollen noch nicht für das Dokument festgelegt wurden (Tabelle1, Zeile 28). Wenn die Lebenszeitkontrollen bereits implementiert sind, wird die auto_close Routine verlassen (Tabelle 1, Zeile 29). Wenn die Lebenszeitkontrollen noch nicht implementiert wurden, fragt das Programm den Benutzer, ob er/sie Lebenszeitkontrollen implementieren möchte (Figur 5a). Falls ja, fordert das Programm den Benutzer auf, ein Verfallsdatum für das Dokument einzugeben (Figur 5(b) und Tabelle 1, Zeilen 35-36). Das Verfallsdatum wird dann als "lcdata" gespeichert und "lcmode" wird auf "1" gesetzt (Tabelle 1, Zeilen 37-38). Wenn der Benutzer keine Lebenszeitkontrollen implementieren möchte, wird "lcmode" auf "0" und "lcdata" auf „“ gesetzt (Tabelle 1, Zeilen 39-41). Die Werte von "lcmode" und "lcdata" werden innerhalb des Excel™-Dokuments als einzelne Dokumenteigenschaften gespeichert.

Wenn ein Excel™-Dokument, das das eingebettete Programm enthält, geöffnet wird, wird das auto_open Makro (Tabelle 1, Zeilen 2-20) ausgeführt. Die Werte von "lcmode" und "lcdata" werden aus den einzelnen Dokumenteigenschaften gelesen (Tabelle1, Zeilen 4-7). Wenn "lcmode" gleich "1" ist (Zeile 9), und das gegenwärtige Datum später ist als das Datum von "lcdata" (Zeile 10), wird der Name des Dokuments in der Variable "fn" (Zeile 12) gespeichert und eine Nachricht, die besagt, daß das Dokument nicht länger gültig ist, im Nachrichtenfenster auf dem Bildschirm des Computers gezeigt (Figur 5(c) und Tabelle 1, Zeile 14) und das Dokument geschlossen und gelöscht (Zeilen 15-16). Zwar überschreibt das Programm in Tabelle 1 das Excel™-Dokument nicht mit aussagelosen Zeichen, bevor es das

Dokument löscht, allerdings kann eine solche Funktion leicht hinzugefügt werden, indem zum Beispiel die Clear Methode der Visual Basic™-Programmiersprache verwendet wird, wie unten beschrieben:

```
Range(„A1“).Select
Range(Selection, Selection.SpecialCells(xlLastCell)).Select
Selection.Clear
ActiveWorkbook.Save
```

Figuren 6a bis 6c und Tabelle 2(a,b) zeigen ein sich selbst zerstörendes Dokumentensystem, illustriert unter Bezugnahme auf Microsoft Word 97™-Dokumente. In den Tabellen 2(a,b) wird ein Visual Basic™-Programm zum Erzeugen eines sich selbst zerstörenden Word 97-Dokuments gezeigt, das eine Benutzerschnittstelle, die für eine Microsoft Windows-Umgebung implementiert ist, beinhaltet. Die Implementierung, die gezeigt wird, verwendet FileSave- und FileSaveAs-Makros, um Lebenszeitkontrollen zu starten und verwendet ein AutoOpen-Makro, um die Lebenszeitkontrollen durchzusetzen. Gemäß der gezeigten Ausführungsform ist das Programm in Tabelle 2 als Schablone in das Dokument implementiert.

Jedes Mal, wenn ein Word 97™-Dokument erzeugt wird, wird das Programm aus Tabelle 2 in das Dokument eingefügt. Das Programm bleibt in einer externen Dokumentenschablone auf der Workstation des Verfassers und wird in das neue Dokument kopiert, indem das AutoNew-Makro verwendet wird, wie in Tabelle 2c gezeigt. Dieses Makro führt die Funktion "InstallSDD" aus, die das Programm aus Tabelle 2 aus der Schablone SDD.dot liest und seinen Inhalt in das neue Dokument kopiert, wobei das Microsoft Word Organizer-Objekt Verwendung findet. Die Verwendung dieser Methode stellt sicher, daß das eingefügte Programm im Dokument eingebunden ist, gemäß dieser Ausführungsform der gegenwärtigen Erfindung.

Wenn das neue Dokument gespeichert wird (FileSave oder FileSaveAs, Tabelle 2b), wird die Variable "lcmode" gelesen. Wenn "lcmode" nicht definiert ist (d. h. das Dokument wurde zum ersten Mal gesichert), wird die Routine „pfImplementLifetimeControls“ (Tabelle 2a) eingeschaltet und das Programm fragt den Benutzer, ob er/sie Lebenszeitkontrollen

implementieren möchte (Figur 6a, Tabelle 2a). Falls ja, verlangt das Programm vom Benutzer, ein Verfallsdatum für das Dokument einzugeben (Figur 6(b) und Tabelle 2a). Das Verfallsdatum wird dann als "lcdata" gespeichert und "lcmode" auf "1" gesetzt (Tabelle 2a). Wenn der Benutzer keine Lebenszeitkontrollen implementieren möchte, wird "lcmode" auf "0" gesetzt und "lcdata" auf „“ (Tabelle 2a). Die Werte von "lcmode" und "lcdata" werden innerhalb des Word 97™-Dokuments als Dokumentvariablen gespeichert.

Wenn ein Word 97™-Dokument, das das eingefügte Programm enthält, geöffnet wird, wird das AutoOpen-Makro (Tabelle 2b) ausgeführt. Die Werte von "lcmode" und "lcdata" werden gelesen, wobei die Funktionen "pfGetLcmode" und "pfGetLcdata" verwendet werden (Tabelle 2a, 2b). Falls "lcmode" gleich "1" ist (Tabelle 2b), und das gegenwärtige Datum später ist als das in "lcdata", wird der Name des Dokuments in der Variablen doc (Zeile 12) gespeichert, die Variable "rng" auf die Länge des Dokuments gesetzt, das Dokument mit aussagelosen Zeichen überschrieben und dann gelöscht (Kill(dlg.name)), und eine Nachricht, die besagt, daß das Dokument nicht länger gültig ist, im Nachrichtenfenster auf dem Bildschirm des Computers gezeigt (Figur 6c und Tabelle 2b).

Figuren 7(a) bis 7(e) und Tabelle 3 zeigen ein sich selbst zerstörendes Dokumentensystem unter Bezugnahme auf Microsoft Outlook™ 98. In Tabelle 3 wird zur Veranschaulichung ein Visual Basic Programm gezeigt, um ein sich selbst zerstörendes Outlook™ 98 E-Mail zu erstellen, das eine Benutzerschnittstelle, die in eine Microsoft Windows-Umgebung implementiert werden kann, enthält. In Figuren 7(a) und 7(b) wird ein Kasten mit der Inschrift "Self-destruct" zur voreingestellten Schablone der E-Mailnachricht eingefügt. Figur 7(a) zeigt den konventionellen Nachrichtkasten 74, der gezeigt wird, wenn ein Benutzer eine E-Mail erstellen möchte. Der Kasten 74 enthält den Text der E-Mail 76 und die Adressen der Empfänger der E-Mail 77. Figur 7(b) zeigt den Selbstzerstörungskasten, der einen Schaltkasten 71 enthält, der anzeigt, ob die Nachricht eine sich selbst zerstörende sein wird, und einen Datumskasten 72, in den das Zerstörungsdatum eingegeben wird. Wie unten beschrieben ist das Zerstörungsdatum in die E-Mail eingebettet und wird gelesen und verarbeitet, wenn die E-Mail geöffnet wird.

Outlook 98 unterstützt Outlook-Zubehör (individuelle Objekte wie Nachrichten, Kalender, Einträge, Aufgaben, usw.), die Visual Basic-Code unterstützen. Eine Auswahl an Ereignissen

wird für jedes Zubehör definiert und Visual Basic-Code kann geschrieben werden, um jedes Ereignis auszuführen. Im Outlook-Objektmodell wird eine E-Mail "MailItem" genannt. Unter Outlook™ ist es einem Programmierer möglich, Anwendungsverhalten zu modifizieren, und Nachrichten-Erstellungs- und Nachrichten-Öffnungsverhalten können modifiziert werden.

Tabelle 3 zeigt ein Visual Basic-Programm, das in eine E-Mail eingefügt ist und immer wenn die E-Mail mit Hilfe der Funktion Item_Open geöffnet wird, ausgeführt wird. Wenn eine E-Mail erzeugt wird und ein Abfragen in den Schalterkasten 71 eingegeben wird und ein Datum in den Datumskasten 72 des "sich selbst zerstören" Kasten 73 eingegeben wird, wird der Wert des Schalterkastens 71 in "Item.UserProperties("DoDestrict")" gespeichert, und das Datum, das in den Datumskasten 72 eingegeben wurde, in "Item.UserProperties("DestructDate")" gespeichert. Diese Werte sind in der E-Mailnachricht zusammen mit dem Programm aus Tabelle 3 eingebettet. Wenn die E-Mail geöffnet wird, wird die eingebettete Programmfunktion Item_Open automatisch ausgeführt. Gemäß Tabelle 3 wird das Programm verlassen und die E-Mail geöffnet, wenn die Eigenschaft von DoDestruct falsch ist (d.h. die Nachricht ist keine sich selbst zerstörende Nachricht). Falls aber die Eigenschaft von DoDestruct wahr ist, überprüft das Programm, ob die Nachricht verfallen ist, indem die Funktion "itemExpired" implementiert wird, die dann die Funktion "daysTilDestruct" implementiert. Die Funktion daysTilDestruct subtrahiert das gegenwärtige Datum von dem Wert von "DestructDate", und gibt den Unterschied in Tagen zurück. Der Wert, der von der Funktion daysTilDestruct zurückgegeben wird, wird in der Variablen "dt" in der Funktion itemExpired gespeichert. Wenn "dt" kleiner oder gleich Null ist, gibt die Funktion "itemExpired" den boolschen Wert True an die Funktion Item_Open zurück, der Text der Nachricht wird überschrieben mit dem Text "This message is no longer available", wie in Figur 7(e) gezeigt, und die Nachricht "This message is no longer available" erscheint in einem Dialogkasten, wie in Figur 7(d) gezeigt. Wenn "dt" größer als 1 ist, gibt die Funktion "itemExpired" den boolschen Wert False zurück, und die Nachricht "This message will destruct in [daysTilDestruct] days" erscheint in einem Dialogkasten, wie in Figur 7(c) gezeigt.

TABELLEN 1 BIS 3

```
Attribute VB_Name = "Module 1"
        Sub auto_open()
```

```
 On Error GoTo ehEnd:
 Dim lcmode As String
 lcmode ActiveWorkbook.CustomDocumentProperties("lcmode")
 Dim lcdata As String
 lcdata = ActiveWorkbook.CustomDocumentProperties("lcdata")
 On Error Resume Next
 If lcmode = "1" Then
  If DateValue(Date) >= DateValue(lcdata) Then
  Dim fn As String
    fn = ActiveWorkbook.Name
    Dim ret
    ret = MsgBox("" + fn +,,, is no longer valid.", vbCritical + vbOKOnly,
,,Purdue/SDD")
    ActiveWorkbook.Close (False)
    Kill (fn)
  End [f
 End If
ehEnd:
End Sub
Sub auto_close()
 On Error GoTo ehImplement:
 Dim lcmode As String
 lcmode = ActiveWorkbook.CustomDocumentProperties("lcmode")
 Dim lcdata As String
 lcdata = ActiveWorkbook.CustomDocumentProperties("lcdata")
 On Error Resume Next
 If lcmode<>""Then
    Exit Sub
 End If
ehImplement:
 Dim ret
 ret = MsgBox("Would you like to implement lifetime controls in this
 workbook?",
```

```
vbQuestion + vbYesNo, „Purdue/SDD")
If ret = vbYes Then
    Icdata = InputBox("Please enter the expiration date for this workbook.",
"Purdue/SDD", Date)
     ret = ActiveWorkbook.CustomDocumentProperties.Add("1cmode", False,
msoPropertyTypeString, "1")
   ret = ActiveWorkbook.CustomDocurnentProperties.Add("lcdata", False,
msoPropertyTypeString, Icdata)
   ElseIf ret = vbNo Then
   ret ActiveWorkbook.CustomDocumenProperties.Add("lcmode", False,
msoPropertyTypeString, "0")
   ret ActiveWorkbook.CustomDocumentproperties.Add("lcdata",False,
msoPropertyTypeString, "")
    End If
End Sub
```

TABELLE 1

```
Attribute VB_Name ="Utilities"
Public Function pfGetLcmode(doc As Document) As String
  On Error GoTo ehUndefined
  Dirn Icmode As String
  lcmode = doc.Variables("Icmode")
  pfGetLcmode = Icmode
  Exit Function
ehUndefined:
  pfGetLcmode =""
 End Function


Public Function pfGetLcdata(doc As Document) As String
```

```
  On Error GoTo ehUndefined
  Dim lcdata As String
  Icdata = doc.Variables("Icdata")
  pfGetLcdata = Icdata
  Exit Function
ehUndefined:
  pfGetLcdata =""
End Function

Public Function pflmplementLifetirneControls(doc As Document) As Boolean
  Dim ret
  ret = MsgBox("Would you like to implement lifetime controls in this document?",
vbQuestion + vbYesNo, „Purdue/SDD")
  If ret = vbYes Then
                ret = doc.Variables.Add("lcmode", "1")
                Dim data As String
                data = InputBox("Please enter the expiration date of this document.",
              „Purdue/SDD", Date)
                ret = doc.Variables.Add("Icdata", data)
              ElseIf ret = vbNo Then
                ret = doc.Variables.Add("Icrnode", "0")
                 End If
                 pfSetLc = True
              End Function
```

TABELLE 2A

```
Sub FileSav()
Attribute FileSave.VB_Description = „Saves the active document or template"
Attribute FileSave.VB_ProcData.VB_Invoke_Func = Norrnal.NewMacros.FileSave"
    Dim lcmode As String
    lcrnode = pfGetLcrnode(ActiveDocurnent)
    If lcmode = ""Then
```

```
            Dim ret As Boolean
            ret = pfImplementLifetimeControls(ActiveDocurnent)
    Else
    End If
    ActiveDocurnent.Saye
End Sub


Sub FileSaveA()
Attribute FileSaveAs.VB_Description = „Saves a copy of the document in a separate file"
Attribute FileSaveAs.VB_ProcData.VB_Invoke_Func =
„Normal.NewMacros.FileSaveAs"
  Dim lcmode As String
  lcmode = pfGetlcmode(ActiveDocument)
  If lcmode = ""Then
    Dim ret As Boolean
    ret = pfImplementLifetimeControls(ActiveDocurnent)
  Else
  Endlf
  Dialogs(wdDialogFileSaveAs).Show
End Sub

Sub AutoOpen()
   Dim doc As Document
   Set doc = ActiveDocument
   Dim lcmode As String
   lcmode = pfGetLcmode(doc)
   Dim lcdata As String
   lcdata = pfGetLcdata(doc)
   If lcmode  = "1"Then
     If DateValue(Date) >= DateValue(lcdata) Then
        Dim rng As Range
        Set rng = doc.Content()
```

```
        ret = rng.Delete()
        doc.Close (wdSavechanges)
        Kill (dlg.Name)
        ret = MsgBox("This document is no longer valid." & Chr(13) & „lt
        has been destroyed.", vbCritical + vbOKOnIy, "Purdue/SDD")
       Exit Sub
     End If
  End If
End Sub
```

TABELLE 2B

```
Sub AutoNew()
    Call lnstallSDD
End Sub

Sub lnstallSDD()
        Dim tPath As String
        tPath = Options.DefaultFilePath(wdUserTemplatesPath)
        tPath = tPath + "\SDD.dot"
        On Error GoTo errHandler
        Application.OrganizerCopy _
        Source:=tPath,_
        Destination:=ActiveDocumentName, _
        Name:="SDDModule",_
        Object:wdorganizerObjectProjectltems
        Exit Sub
errHandler:
        MsgBox „Could not load self-destructing document module."
        Exit Sub
End Sub
```

TABELLE 2C

```
Function 1tem_Open()
   Ifltem.UserProperties ("DoDeststruct").Value = False Then
      Item_Open = True
      Exit Function
   End If
   If itemExpired () Then
      Item.Body = "This message is no longer available."
      Item.Save ()
      MsgBox („This message is no longer available.")
   Else
      MsgBox ("This message will destruct in" & daysTilDestruct () & "days.")
   End If
End Function


Function itemExpired
   dt = daysTilDestruct ()
   If dt<=OThen
      itemExpired = True
   Else
      itemExpired = False
   Endif
End Function

Function daysTilDestruct
      daysTilDestruct = DateDiff("d", Now (), Item.UserProperties
(„DestructDate").Value)
End Function
```

TABELLE 3

Des weiteren können die ausführbaren Module gemäß der sich selbst zerstörenden Dokumente, die oben beschrieben sind, auch Verschlüsselungstechniken verwenden, um

einen Benutzer daran zu hindern, die Lebenszeitkontrollen zu umgehen, indem er zum Beispiel ein sich selbst zerstörendes Dokument mit einer Anwendung öffnet, die das ausführbare Modul nicht ausführen kann. Gemäß einer solchen Ausführungsform wird das ausführbare Modul das Dokument verschlüsseln, wenn es zum ersten Mal gespeichert oder geschlossen wird. Wenn das Dokument das nächste Mal geöffnet wird, wird es das ausführbare Modul nur dann entschlüsseln, wenn die Lebenszeitkontrollen gültig sind. Des weiteren wird es dem Benutzer nicht möglich sein, das Dokument einzusehen, wenn er es mit einer Anwendung öffnen will, die das ausführbare Modul nicht ausführen kann, da es verschlüsselt bleibt. Zum Beispiel, wenn ein Benutzer versucht, ein sich selbst zerstörendes Microsoft Word™-Dokument mit einer Word Perfect™-Anwendung zu öffnen, wird die Word Perfect™-Anwendung das Microsoft Word™-AutoOpen- oder FileOpen-Makro nicht ausführen, und das Dokument wird geöffnet, egal ob die Lebenszeitkontrollen des Dokuments gültig sind oder nicht. Wenn aber das sich selbst zerstörende Microsoft Word™-Dokument vorher von den Makros AutoClose, FileSave oder FileSaveAs verschlüsselt wurde (indem beispielsweise ein exclusiv- oder polyalphabetischer Algorithmus verwendet wird, wie unten beschrieben), bleibt das Dokument verschlüsselt, bis der Benutzer die Datei mit einer Anwendung öffnet, die das Makro ausführt (das den entsprechenden Entschlüsselungsalgorithmus für den exclusiv- oder polyalphabetischer Algorithmus enthält). Die Wahl einer passenden Verschlüsselungstechnik hängt vom gewünschten Sicherheitsgrad ab und von der Fähigkeit des Makros oder Scripts, das verwendet wird, sie zu implementieren.

Virtuelle Behälter

Im folgenden wird ein Dokumenten-Sicherheitssystem gemäß der gegenwärtigen Erfindung beschrieben, das virtuelle Behälter einsetzt.

Gemäß einer ersten Ausführungsform des Dokumenten-Sicherheitssystems, das in Figur 8a gezeigt wird, wird ein Behälter 500 für einen Dokumentensatz verwendet, um ein oder mehrere Dokumente 510 (oder andere digitale Objekte) zu speichern, jedes mit einer unabhängigen Lebenszeitkontrolle 520. Auf jeden Computer, auf dem ein Benutzer die Inhalte des Behälters öffnen und einsehen möchte, wird eine Anwendung installiert. Diese Anwendung wertet die Lebenszeitkontrollen eines Dokuments, bevor es dem Benutzer

gezeigt wird. Falls die Lebenszeitkontrollen gültig sind, wird das Dokument aus dem Behälter geholt und kann vom Benutzer verwendet werden. . Falls die Lebenszeitkontrollen ungültig sind, wird das Dokument vernichtet, indem es beispielsweise mit aussagelosen Zeichen überschrieben wird.

Gemäß einer zweiten Ausführungsform des Dokumenten-Sicherheitssystems zeigt Figur 8(b) einen Proxy-Behälter 500', der dazu verwendet werden kann, ein einzelnes Dokument (oder anderes digitales Objekt) zu speichern. Ein Proxy-Verwalter wird auf jedem Computer, auf dem ein Benutzer ein Dokument im Proxy-Behälter öffnen und darstellen lassen möchte, installiert. Anders als bei einem Behälter für Dokumentensätze ist die Operation eines Proxys für den Benutzer unsichtbar – außer die Lebenszeitkontrollen 520' des Dokuments sind ungültig. Um diese Transparenz zu gewährleisten, wird der Proxy benannt oder auf andere Weise enkodiert, um als normaler Bestandteil des Dokuments zu erscheinen, und der Betriebssystemmechanismus, der Dokumente ihren Anwendungen zuordnet wird derart geändert, daß er stattdessen die Dokumente dem Proxy-Verwalter zuordnet. Der Proxy-Verwalter komplettiert die Zuordnung zwischen Dokument und Anwendung übergangsweise, wenn die Lebenszeitkontrollen gültig sind. Diese Ausführungsform zieht Nutzen aus der Tatsache, daß, wenn auf eine Datei zugegriffen wird, indem zum Beispiel mit der Maus auf eine Datei von einer Windows™ 95-Umgebung aus, von einem Windows 95-Dokumentensymbol in "My Computer", oder vom Dateimanager in Windows 3.1 geklickt wird, das Betriebssytem versucht, die Datei einer Anwendung, wie Microsoft Word™, Microsoft Excel™ oder WordPerfect™ zuzuordnen. In Übereinstimmung mit dem Proxy-Behälter ist die Proxy-Behälteranwendung zwischen Betriebssytem und Anwendung geschaltet, so daß, wenn ein Benutzer den Proxy-Behälter anspricht (der dem Benutzer als ein gewöhnliches Dokument erscheint), das Betriebssystem den Behälter-Öffner veranlaßt, den Behälter automatisch zu öffnen und, wenn die Lebenszeitkontrollen gültig sind, die Anwendung startet und das Dokument im Behälter öffnet. Wenn die Lebenszeitkontrollen ungültig sind, wird das Dokument beispielsweise durch Überschreiben mit aussagelosen Zeichen und/oder Löschen zerstört. Gemäß weiteren Ausführungsformen des Proxy-Behälters kann das System derart konfiguriert werden, daß es das Dokument automatisch an den Proxy-Behälter zurückgibt, wenn die Anwendung geschlossen wird.

Im Folgenden werden mehrere Ausführungsformen des Behälters für Dokumentensätze (DS-Behälter) erläutert. Gemäß einer ersten Ausführungsform des DS-Behälters werden eine Behälter-Öffner- und eine Behälter-Erzeuger-Anwendung zu Verfügung gestellt. Die Behälter-Erzeuger-Anwendung liegt auf dem Computer eines Benutzers, der einen DS-Behälter erstellen möchte, und, wie ihr Name bereits nahe legt, ist dazu in der Lage, einen DS-Behälter zu erzeugen, Lebenszeitkontrollen zu spezifizieren und ein Dokument im DS-Behälter zu plazieren. Die Behälter-Öffner-Anwendung liegt auf dem Computer eines Benutzers, der einen DS-Behälter öffnen möchte, und kann den DS-Behälter öffnen, um die Gültigkeit der Lebenszeitkontrollen zu überprüfen und das Dokument aus dem DS-Behälter zu holen, falls die Lebenszeitkontrollen gültig sind. Wenn die Lebenszeitkontrollen nicht gültig sind, wird das Dokument beispielsweise durch Überschreiben mit aussagelosen Zeichen zerstört. Gemäß einer Ausführungsform enthält der DS-Behälter nur ein Dokument. Gemäß weiteren Ausführungsformen können mehrere Dokumente in einem einzelnen Behälter gespeichert werden, wobei jedes Dokument seine eigene Lebenszeitkontrollen haben kann. In Situationen, in denen es angebracht erscheint, daß der Benutzer einen DS-Behälter nur öffnen, aber nicht erzeugen kann, kann die Behälter-Öffner-Anwendung auf dem Computer installiert werden, aber nicht die Behälter-Erzeuger-Anwendung. Unabhängig von der Anzahl der im DS-Behälter verbleibenden Dokumente, erscheint ein ungeöffneter DS-Behälter einem Benutzer als eine einzige Datei und beinhaltet Kopfzeilen-Information (die zum Beispiel Lebenszeitkontrollen einschließt) und das Dokument oder die Dokumente.

Gemäß einem weiteren Aspekt der DS-Behälter für mehrere Dokumente gemäß einer Ausführungsform der Erfindung kann der DS-Behälter eine Behälter-Kopfzeile enthalten, die Kontrollinformation für den Behälter enthält und die sich auf zwei einzeln gekoppelte Stromlisten bezieht. Die erste gekoppelte Liste enthält aktive Ströme, die zweite enthält inaktive Ströme. Aktive Ströme beziehen sich auf Dokumente mit noch gültigen Lebenszeitkontrollen und inaktive auf Dokumente mit ungültigen Lebenszeitkontrollen oder die vom Behälter entfernt (gelöscht) wurden. Sowohl aktive als auch inaktive Ströme besitzen Kopfzeilen, die die Lebenszeitkontrolle (und vorzugsweise Verschlüsselungsinformation wie unten beschrieben) für die entsprechenden Ströme enthalten. Diese Kopfzeilen der Ströme enthalten jeweils eine Referenz auf den nächsten Strom in ihrer Liste (aktiv oder inaktiv), oder eine Anzeige, daß dieser Strom der letzte in seiner Liste ist. Jeder Strom ist ein angrenzender Block binärer Daten.

Wenn ein Strom ungültig gemacht oder vom Behälter entfernt wird, wird seine Kopfzeile als inaktiv markiert, in die Liste der inaktiven Ströme gesetzt, und der Strom mit aussagelosen Zeichen überschrieben. Wenn dem Behälter ein neuer Strom hinzugefügt wird, kann ein inaktiver Strom reaktiviert und benutzt werden, wenn er lang genug ist, den neuen Strom zu enthalten. Wenn kein inaktiver Strom zu Verfügung steht bzw. lang genug ist, den neuen Strom zu enthalten, kann im Behälter ein neuer Strom erzeugt und aktiviert werden. Wenn ein Behälter viele Änderungen an den Listen aktiver und inaktiver Ströme erfährt, kann er fragmentiert werden. In anderen Worten, wenn Dokumente verfallen und durch aussagelose Zeichen ersetzt werden, können aktive Dokumente zwischen inaktive Dokumente, die nur aussagelose Zeichen enthalten, geraten. In einer bevorzugten Ausführungsform der Erfindung kann die Behälter-Erzeuger-Anwendung diesen Zustand erkennen und den Behälter, falls nötig, komprimieren.

Figuren 9(a) und 9(b) zeigen die aktiver/inaktiver Strom-Ausführungsform der gegenwärtigen Erfindung. Figur 9(a) zeigt einen virtuellen Behälter, der drei Dokumente enthaltend erzeugt wurde. Wie in Figur 9(a) gezeigt, enthält die Erzeuger-Information für die Behälter-Kopfzeile (wie den Namen des Behälters, die standardmäßigen Lebenszeitkontrollen für den Behälter, das Datum, an dem der Behälter erzeugt wurde, das Datum, an dem der Behälter zuletzt modifiziert wurde und das Datum, an dem der Behälter zuletzt betreten wurde), Information über den aktiven Strom (wie ein Zeiger auf das erste Dokument im aktiven Strom und die Anzahl der Dokumente im aktiven Strom), und Information über den inaktiven Strom (wie ein Zeiger auf das erste Dokument im inaktiven Strom und die Anzahl der Dokumente im inaktiven Strom). Jedes Dokument im Behälter enthält eine Dokumenten-Kopfzeile, die Lebenszeitkontrollinformation und einen Pfeil auf das nächste Dokument im betreffenden Strom enthält. Da der Behälter gerade erst erzeugt wurde, sind alle drei Dokumente im aktiven Strom, und die Behälter-Kopfzeile enthält eine Unter-Kopfzeile des aktiven Stroms, die einen Zeiger auf das erste Dokument und eine Anzeige, daß sich drei Dokumente im aktiven Strom befinden, enthält. Die Unter-Kopfzeile des inaktiven Stroms enthält keine Information. Dann wurden in Figur 9(b) zwei neue Dokumente (Dokumente 4 und 5) zum Behälter dazugenommen und Dokument 2 ist verfallen. Während die Unter-Kopfzeile des aktiven Strombehälters immer noch auf Dokument 1 zeigt, zeigt der Zeiger in der Kopfzeile

von Dokument 1 nun auf Dokument 3, anstatt auf Dokument 2. Außerdem zeigt die Unter-Kopfzeile des inaktiven Strombehälters nun auf Dokument 2.

Wie oben ausgeführt, enthält ein virtueller Behälter gemäß der gegenwärtigen Erfindung ein oder zwei Dokumente, die zerstört werden sollten, wenn sie nicht mehr gültig sind (wie von den Lebenszeitkontrollen bestimmt). Generell sollten die Kosten für den Aufwand, diese Kontrollen zu umgehen, höher sein, als der Wert, das gültige Leben der Dokumente zu verlängern.

Gemäß einer bevorzugten Ausführungsform der Erfindung sind die Dokumente im virtuellen Behälter verschlüsselt, um unautorisierten Zugriff auf die Dokumente im Behälter zu verhindern und um einen Benutzer daran zu hindern, die Lebenszeitkontrollen eines Dokuments zu umgehen. Diese Verschlüsselungstechnik kann zum Beispiel bei allen der hier beschriebenen Behälter angewendet werden, die DS-Behälter in Figuren 8(a), 9(a) und 9(b) und die Proxy-Behälter in Figur 8(b) eingeschlossen.

Gemäß einer weiteren Ausführungsform der gegenwärtigen Erfindung ermöglicht es das System dem Autor, aus verschiedenen Sicherheitsebenen auszuwählen, abhängig vom Wert der entsprechenden Dokumente. Im allgemeinen nehmen Aufwand zum Umgehen der Sicherheitseinrichtungen – und Kosten und Komplexität der Sicherheitseinrichtungen – mit steigender Sicherheitsebene zu. Bevor beschrieben wird, wie die Verschlüsselung in virtuellen Behältern eingesetzt werden kann, werden einige Grundprinzipien der Verschlüsselungstechnologie unterbreitet.

Übersicht über Verschlüsselungstechniken

Im allgemeinen bestehen Verschlüsselungstechniken – im folgenden Verschlüsselungssysteme genannt – aus zwei Hauptkomponenten: Algorithmus und Schlüssel. Der Algorithmus ist eine mathematische Funktion, die Daten transformiert. Die Schlüssel "lenken" die exakte Art der Transformation.

Verschlüsselungsalgorithmen können entweder offen oder geschlossen sein. Offene Algorithmen werden von ihren Designern veröffentlicht, von Akademikern verbessert und

durch kommerzielle Anwendungen überprüft. Geschlossene Algorithmen bleiben Eigentum der Designer, die nicht wollen, daß sie von der Öffentlichkeit verstanden oder benutzt werden. Im allgemeinen sind die Verschlüsselungsalgorithmen, die in kommerziellen Produkten verwendet werden, "offene" Algorithmen. Ein Verschlüsselungsalgorithmus, der veröffentlicht und verbessert wurde und nach seiner Implementierung weite Anwendung fand, kann im allgemeinen als sicher angesehen werden, da er sehr wahrscheinlich von großen Forschungsuniversitäten untersucht und von Informatikstudenten in der ganzen Welt attackiert wurde, ohne in Ungnade zu fallen.

Die Stärke eines effektiven Verschlüsselungssystems hängt in der Regel von seinen Schlüsseln ab. Vorausgesetzt, daß der Algorithmus bekannt ist (wie oben erläutert), hängt die Sicherheit des Verschlüsselungssystems von der Erzeugung passender Schlüssel und der vorsichtigen Bewahrung ihrer Geheimhaltung ab. Ein wichtiges Maß ihrer Stärke ist ihre Länge – typischer Weise in Bits gemessen. Ein kurzer Schlüssel bedeutet schnelles Ver- und Entschlüsseln, aber auch ein einfacheres Knacken des Schlüssels. Ein langer Schlüssel bedeutet lange Ver- und Entschlüsselungszeiten, kann aber grundsätzlich unknackbar sein. Diese Erkenntnis ist so überzeugend, daß die U.S.-Regierung die freie und öffentliche Nutzung jedes Verschlüsselungsalgorithmus erlaubt, den Längen der Schlüssel aber klare Reglementierungen auferlegt.

Obwohl eine ganze Reihe von offenen und geschlossenen Verschlüsselungstechniken gemäß der gegenwärtigen Erfindung Anwendung finden können, werden hier nur vier Verschlüsselungstechniken kurz vorgestellt: Symmetrische Algorithmen, Öffentlicher-Schlüssel-Algorithmen, digitale Signaturen und one-way hashes.

Symmetrische Algorithmen bedienen sich im allgemeinen eines einzelnen Schlüssels zum Ver- und Entschlüsseln. Sie sind in der Regel einfach zu verstehen und zu benutzen und schaffen ein relativ hohes Maß an Sicherheit, wenn sie richtig verwendet werden. Jedenfalls müssen sich beide Parteien, bevor sie miteinander kommunizieren, auf einen gemeinsamen Schlüssel einigen, ohne daß Dritte von diesem Schlüssel erfahren. Auch muß der Schlüssel für immer geheim bleiben, oder alte Nachrichten können später entschlüsselt werden. Symmetrische Algorithmen sind insbesondere für flüchtige Nachrichten geeignet, die nach Beendung der Kommunikation nicht gespeichert werden. Wenn die Nachricht nicht

gespeichert wird, gibt es auch keine Gelegenheit, die Nachricht später zu entschlüsseln, selbst wenn der Schlüssel bekannt würde. Ein populärer symmetrische Algorithmus ist der Data Encryption Standard, ein ANSI und ISO Standard, seit über 20 Jahren in Gebrauch.

Öffentlicher-Schlüssel-Algorithmen verwenden zwei Schlüssel – einer öffentlicher Schlüssel, der andere privater Schlüssel genannt. Ein öffentlicher oder privater Schlüssel kann verwendet werden, um eine Nachricht zu entschlüsseln. Nur der private Schlüssel kann eine Nachricht entschlüsseln, die mit dem öffentlichen Schlüssel verschlüsselt wurde und nur der öffentliche Schlüssel kann eine Nachricht entschlüsseln, die mit dem privaten Schlüssel verschlüsselt wurde. Außerdem ist es praktisch nicht möglich, einen zweiten privaten Schlüssel passend zum öffentlichen Schlüssel zu erstellen. Typischerweise wird der öffentliche Schlüssel eines jeden Paares an einem wohlbekannten, vertrauenswürdigen Ort veröffentlicht (zum Beispiel eine Datenbank oder ein Verzeichnis im Internet), während der Besitzer den privaten Schlüssel behält. Ein gewöhnliches Anwendungsbeispiel für die Verwendung eines öffentlichen Schlüssels ist das folgende: Alice möchte Bob eine geheime Nachricht senden. Alice erlangt Bobs öffentlichen Schlüssel von einer vertrauenswürdigen Quelle. Sie verwendet diesen Schlüssel, um ihre Nachricht an Bob zu verschlüsseln. Wenn Bob die verschlüsselte Nachricht erhält, kann er sie mit seinem privaten Schlüssel entschlüsseln. Wenn a) Alice Bobs tatsächlichen öffentlichen Schlüssel (und keine Fälschung) erhalten hat, und b) Bob der einzige Besitzer seines privaten Schlüssels ist (kein anderer hat eine Kopie), dann kann Alice sicher sein, daß nur Bob ihre Nachricht lesen kann. Ein populärer öffentlicher-Schlüssel Algorithmus ist das RSA Cryptosystem, das RSA Data Security, Inc gehört.

Digitale Signaturen dienen, um die Identität des Erstellers der Nachricht zu authentifizieren. Obwohl es viele Möglichkeiten für die Implementierung digitaler Signaturen gibt, wird oft öffentlicher-Schlüssel Verschlüsselung verwendet. Wir können das oben erläuterte Szenario ausweiten. Alice möchte Bob eine private Nachricht senden, und sie möchte, daß er ganz sicher sein kann, daß diese Nachricht auch von ihr ist. Das vorherige Szenario garantiert Bob nicht, daß die Nachricht von Alice ist; es garantiert Alice lediglich, daß kein dritter ihre Nachricht an Bob gelesen hat. Also verschlüsselt Alice ihre Nachricht an Bob mit ihrem privaten Schlüssel, Dann verschlüsselt sie die Nachricht noch einmal mit Bobs öffentlichem Schlüssel, wie oben. Wenn Bob die Nachricht empfängt, entschlüsselt er sie als erstes mit

erstes mit seinem privaten Schlüssel. Dann holt er sich Alices öffentlichen Schlüssel von einer vertrauenswürdigen Quelle, um sicherzustellen, daß die Nachricht auch von ihr ist. Diesen Schlüssel verwendet er, um die Nachricht zu entschlüsseln. Wenn a) Bob sicher ist, daß Alices Schlüssel nicht durch Kopieren oder Fälschen verbreitet wurden (wie oben dargelegt), und b) Alice sicher ist, daß Bobs Schlüssel nicht verbreitet wurden, kann Alice sicher sein, daß nur Bob ihre Nachricht lesen kann und Bob kann sicher sein, daß die Nachricht von Alice kommt. RSA Data Security, Inc. Produziert einen weit verbreiteten digitale-Signatur Algorithmus.

Während digitale Signaturen in der digitalen Welt einen zuverlässigen Weg darstellen, die Identität zu prüfen, gibt es bestimmte Bereiche, in denen es sinnvoll ist, den Verfasser eines Dokumentes zu überprüfen, ohne das gesamte (eventuell große) Dokument zu versenden. Sogenannte one-way hashes bieten eine exzellente Lösung. Sie bestehen aus mathematischen Funktionen, die jede Datenmenge (ein Dokument oder anderes) in einen kleinen Strom binärer Information transferieren. Eine wichtige Eigenschaft der one-way hash Funktionen ist es, daß diese Funktionen keine zwei identischen Ummischungen zweier verschiedener Quellen erzeugen. Es wurden große akademische und kommerzielle Forschungsanstrengungen unternommen, um one-way hashes zu erzeugen, die diese Eigenschaft besitzen. MD 5 ist eine wohl bekannte one-way hash Funktion, die eine 128 Bit Ummischung erzeugt.

Anwendung der Verschlüsselungstechnologie auf virtuelle Behälter

Gemäß einer bevorzugten Ausführungsform der Erfindung ermöglicht es das Dokumentensicherheitssystem der virtuellen Behälter einem Benutzer (und/oder einem Systemadministrator), aus mehreren Sicherheitsebenen auszuwählen. Eine untere Sicherheitsebene beinhaltet keine Verschlüsselung der Dokumente. Die Sicherheit kommt durch die Benutzung eines Behälter-Öffners beim Darstellen der Dokumente zustande. Jeder kommerziell beziehbare Editor für binäre Dateien kann verwendet werden, um den Behälter als einzelnes Dokument zu öffnen und dabei seine Inhalte anzusehen. Genauso können solche Programme dazu verwendet werden, den Behälter als einzelnes Dokument zu öffnen und den darin enthaltenen Text anzusehen, da die meisten Textverarbeitungsprogramme es einem Benutzer ermöglichen, die Dokumente als ASCII Text darzustellen.

Eine zweite Sicherheitsebene kann eine Form von symmetrischer Verschlüsselung sein, "exclusive" oder "polyalphabetic cipher" genannt. Dieser Verschlüsselungsalgorithmus kann leicht in Software implementiert werden und arbeitet sehr schnell. Dieser Verschlüsselungsalgorithmus verwendet einen einzelnen Schlüssel, um die Dokumente zu ver- und zu entschlüsseln. Dieser Schlüssel kann fest in die Software eingebaut sein, kann mit der Software mitgeliefert werden und periodisch vom Benutzer getauscht werden, oder, jedes Mal, wenn er gebraucht wird, von einer vertrauenswürdigen Quelle bezogen werden. Gemäß dieser Methode wird das Dokument mit dem Schlüssel durch ein XOR verschlüsselt (d.h. [Dokument]XOR[Schlüssel] = [verschlüsseltes Dokument]). Zum Beispiel können Behälter-Öffner- und Behälter-Erzeuger-Anwendungen einen einzelnen, fest eingebauten symmetrischen Schlüssel für Ver- und Entschlüsselung besitzen. Der Verfasser eines Dokuments verschlüsselt das Dokument und optional seine Lebenszeitkontrollen, indem er den Behälter-Erzeuger verwendet. Wenn der Verfasser das Dokument verteilen möchte, schiebt er das verschlüsselte Dokument in die Behälter-Öffner-Anwendung. Der Empfänger entschlüsselt Dokument und Lebenszeitkontrollen mit der Behälter-Öffner-Anwendung. Die Behälter-Öffner-Anwendung überprüft die Lebenszeitkontrollen und zeigt, wenn sie gültig sind, das Dokument dem Empfänger. Wenn die Lebenszeitkontrollen nicht gültig sind, zerstört die Behälter-Öffner-Anwendung das Dokument.

Eine höhere Sicherheitsebene verwendet eine zentrale Autorität (CA), um die Sicherheit des Dokuments sicherzustellen. Gemäß einer ersten Ausführungsform des CA-Systems sendet der Autor das Dokument und seine Lebenszeitkontrollen an die CA. Die CA schickt einen einzigartigen Identifizierer für dieses Dokument zurück. Wenn der Autor das Dokument verteilen möchte, schickt er den einzigartigen Identifizierer zum Empfänger. Der Empfänger schickt diesen einzigartigen Identifizierer an die CA, um das Dokument anzusehen. Die CA überprüft die Lebenszeitkontrollen und schickt, wenn sie gültig sind, das Dokument an den Empfänger. Der Empfänger erhält das Dokument und kann es ansehen. Obgleich der Einsatz einer CA die Sicherheit steigert, hat er den Nachteil, daß eine zentrale Autorität eingreifen muß.

Eine weitere Ausführungsform des CA-Systems verwendet "one-way hashes". Gemäß dieser Ausführungsform sendet der Autor das Dokument und seine Lebenszeitkontrollen an die CA.

Die CA mischt die Lebenszeitkontrollen um und kombiniert die umgemischten Lebenszeitkontrollen, die original Lebenszeitkontrollen und die Dokumente im Behälter. Die CA verschlüsselt diesen Behälter mit ihrem privaten Schlüssel und sendet das Paket zum Autor zurück. Wenn der Autor das Dokument verteilen möchte, sendet er den CA-signierten Behälter an den Empfänger. Der Empfänger öffnet den Behälter mit dem öffentlichen Schlüssel der CA und überprüft die Lebenszeitkontrollen. Wenn sie ungültig sind, wird der Behälter zerstört. Wenn sie gültig sind, mischt er sie um und vergleicht die Ummischung mit der im Behälter gespeicherten. Wenn die Ummischungen nicht übereinstimmen, wird der Behälter zerstört. Wenn die Ummischungen übereinstimmen und die Lebenszeitkontrollen gültig sind, wird das Dokument dargestellt.

Eine weitere Ausführungsform des CA-Systems verwendet "one-way hashes" und einen symmetrischen Schlüssel. Gemäß dieser Ausführungsform sendet der Autor das Dokument und seine Lebenszeitkontrollen an die CA. Die CA erzeugt einen einzigartigen symmetrischen Schlüssel (im folgenden DK genannt) und verschlüsselt das Dokument. Der DK und einzigartiger Identifizierer (im folgenden DKID genannt) werden in einer Datenbank aufgezeichnet. Die CA mischt die Lebenszeitkontrollen um und kombiniert die umgemischten Lebenszeitkontrollen, die original Lebenszeitkontrollen, die verschlüsselten Dokumente und den DKID in einem erzeugten Behälter. Sie signiert das Paket mit ihrem privaten Schlüssel und sendet es dem Autor zurück. Wenn der Autor das Dokument verteilen möchte, sendet er den CA-signierten Behälter an den Empfänger. Der Empfänger öffnet den Behälter mit dem öffentlichen Schlüssel der CA. Er sendet die originalen Lebenszeitkontrollen, die umgemischten Lebenszeitkontrollen und den DKID an die CA. Die CA überprüft die Lebenszeitkontrollen. Wenn sie ungültig sind, stoppt die CA. Wenn sie gültig sind, mischt sie sie um und vergleicht die Ummischung mit der Ummischung, die sie vom Empfänger erhalten hat. Wenn diese Ummischungen nicht übereinstimmen, stoppt die CA. Wenn sie übereinstimmen und die Lebenszeitkontrollen gültig sind, schickt die CA den DK, der sich auf den DKID bezieht, zurück an den Empfänger. Der Empfänger entschlüsselt das Dokument mit dem DK und das Dokument kann dargestellt werden.

Ein Problem dieser Ausführungsform ist, daß der Empfänger das System umgehen kann, indem er neue Lebenszeitkontrollen erzeugt, sie ummischt und diese umgemischten neuen Lebenszeitkontrollen an die CA zusammen mit dem originalen DKID des Behälters sendet.

Dieses Problem kann gemäß einer weiteren Ausführungsform umgangen werden, indem die CA einen verschlüsselten Unterbehälter der originalen Lebenszeitkontrollen, der umgemischten Lebenszeitkontrollen und des DKID erstellt. Die Schlüssel zum Verschlüsseln (zum Beispiel symmetrische Schlüssel oder öffentliche und private Schlüssel) sind nur der CA bekannt. Der Empfänger (der keinen Zugriff auf die Schlüssel im Unterbehälter hat) schickt den Unterbehälter an die CA zurück. Wenn dann die CA den Unterbehälter entschlüsselt, kann sie sicher sein, daß Lebenszeitkontrollen, umgemischte Lebenszeitkontrollen und der DKID, die sie aus dem Unterbehälter erhalten hat, von der CA erzeugt und verschlüsselt wurden.

Eine Implementierung des virtuellen Behälters in der Programmiersprache Java™

Nun wird eine anschauliche Implementierung eines DS-Behälters, der ein Dokument enthält, und eine "exclusive-or polyalphabetische cipher" verwendet, beschrieben, wobei die Behälter-Erzeuger- und Behälter-Öffner-Anwendungen in der Programmiersprache Java™ programmiert sind. Behälter-Öffner und Behälter-Erzeuger werden implementiert, indem Daten- und Dateistromklassen, die Teil des standardmäßigen I/O-Pakets der Programmiersprache Java™ sind, verwendet werden. Jedes Objekt einer bestimmten Stromklasse bezieht sich auf einen Satz zusammenhängender Speicher-Bits.

Ein DS-Behälter beinhaltet gemäß dieser Ausführungsform eine Behälter-Kopfzeile, die Kontrollinformation für den Behälter enthält, und eine Dokumentenkopfzeile für die Dokumente im Behälter. Während diese Ausführungsform des DS-Behälters darauf ausgelegt ist, nur ein einzelnes Dokument zu enthalten, können weitere Ausführungsformen eine unbegrenzte Zahl an Dokumenten enthalten. Gemäß einer bevorzugten Ausführungsform des Einzeldokumenten DS-Behälters zeigt Figur 10a die Behälter-Kopfzeile, die den Namen des Behälters CNAME 701, das Datum DM 705, an dem der Behälter zuletzt modifiziert wurde, das Datum DLA 704, an dem auf den Behälter zuletzt zugegriffen wurde, und die Anzahl an Dokumenten 706 (entweder 0 oder 1) im Behälter enthält. Der Behälter enthält des weiteren ein Dokument 710. Das Dokument hat eine Kopfzeile, die den Namen 707 des Dokuments, die Lebenszeitkontrollen 708 des Dokuments und seine Länge 709 enthält. Gemäß dieser Ausführungsform ermöglichen es die Felder DC 703, DM 704 und DLA 705 dem Benutzer, das Datum, an dem der Behälter erzeugt wurde, das Datum, an dem er zuletzt modifiziert

wurde und das Datum, an dem auf ihn zuletzt zugegriffen wurde, zu bestimmen, wie unten mit Bezugnahme auf die "View Methode" in Figur 5(a) erläutert wird. Natürlich können die Felder "Behälter erzeugt", "zuletzt modifiziert" und "letzter Zugriff" von der Behälter-Kopfzeile ausgelassen werden, falls ihre Funktionalität nicht gewünscht ist.

In Tabelle 4(a) wird ein DS-Behälter erzeugt, indem die Methode "public static int create(String envelopefilename)" aufgerufen wird, die als ihr Argument den String "envelopefilename" enthält. Wenn "envelopefilename" ein bestehender Dateiname ist, oder wenn sein Wert Null ist, wird eine Fehlermeldung zurückgegeben, andernfalls wird das Programm abgearbeitet, um einen neuen Behälter zu erzeugen. Es wird das Objekt "fos" im FileOutputStream definiert, das den Zieldateinamen "envelopefilename" enthält. Es wird ein entsprechendes Objekt "dos" im DataOutputStream definiert, das die Zieldaten, die sich auf das Objekt "dos" ("envelopefilename") beziehen, enthält. Das Programm erzeugt dann die Kopfzeile, die im DataOutputStream Objekt "dos" gespeichert wird. Auf diese Weise wird ein Objekt "eh" in der Klasse SdeEnvelopeHeader definiert und das Feld "d_name" der Klasse SdaEnvelopeHeader auf den String "envelopefilename" gesetzt und die Felder "d-created", "d_last_modified" und "d_last_accessed" auf das gegenwärtige Datum gesetzt. Dann wird die Methode "writeTo" hinzugezogen (Tabelle 4(c)), die den Behälternamen, das Erzeugungsdatum des Behälters, das Datum der letzten Änderung, das Datum des letzten Zugriffs und die Anzahl der Dokumente im Behälter (momentan 0) in das Objekt "dos" schreibt, wobei die Behälterkopfzeile für neu erzeugten Behälter erzeugt wird. Auf diese Weise erscheint der Behälter als eine einzelne Datei, die den Namen, der im Feld "envelopename" erscheint, trägt.

In Tabelle 4(b) wird die Methode addDocument verwendet, um dem Behälter ein Dokument hinzuzufügen. Die Methode addDocument nimmt "envelopefilename", "documentfilename" und "expires" als Parameter. Wenn einer dieser Parameter Null ist, wird eine Fehlermeldung zurückgegeben. Fall kein Behälter mit Namen "envelopefilename" existiert, wird die "create"-Methode verwendet, um einen Behälter zu erzeugen. Falls kein Dokument mit Namen "documentfilename" existiert, wird eine Fehlermeldung zurückgegeben. Andernfalls wird ein Objekt "fis" des FileInputStream für "envelopefilename" definiert und ein Objekt "dis" von DataInputStream für die Daten, die mit "envelopefilename" korrespondieren (DataInputStream(fis)). Aus diese Weise wird der Behälter mit Namen "envelopename" in

"fis" und "dis" geöffnet. Dann wird ein temporärer Behälter mit Namen "[envelopefilename].tmp" erzeugt. So wird ein Objekt "fos" des FileOutputStreams für "[envelopefilename].tmp" und ein Objekt "dos" des DataOutputStreams für die Daten, die mit "envelopefilename" (DataInputStream(fos)) korrespondieren, definiert.

Um den "envelope header" auf den neuesten Stand zu bringen, wird ein Objekt "eh" des SdeEnvelopeHeader erzeugt und die Inhalte von "dis" (das ist die Behälterkopfzeile für "envelopefilename") in "eh" durch die Methode "readForm" (Tabelle 4(c)) eingelesen. Unter Bezugnahme auf Tabelle 4(b) werden die Objekte "d_last_modified" und "d_total_documents" der Klasse SdeEnvelopeHeader auf den neuesten Stand gebracht und der auf den neuesten Stand gebrachte "envelope header" (eh) in das Objekt "dos" des DataOutputStream geschrieben.

Um die Dokumentenkopfzeile auf den neuesten Stand zu bringen, wird das Objekt "file_doc" aus File auf den Pfad von "documentfilename" gesetzt, und die Variable "length" wird gleich der Länge der Datei in "file_doc" gesetzt. Das Objekt "fis_doc" von FileInputStream wird so gesetzt, daß es die Datei "documentfilename" hält. Dann wird ein Objekt "dh" von SdeDocumentHeader erzeugt, und das "d_name" Feld der Klasse SdeDocumentHeader auf den String "documentfilename" gesetzt, das "d_expired" Feld auf den Wert des Parameters "expires" gesetzt, und das "d_total_bytes" Feld auf den Wert der Variable "length" gesetzt. Dann wird das Verfahren "writeTo" herangezogen (Tabelle 4(d)), das den Namen des Dokuments, das Verfallsdatum und die Länge des Dokuments in das Objekt dos schreibt, wobei die Dokumentenkopfzeile für das Dokument erzeugt wird.

Dann wird das Dokument in die Variable "ch" gelesen (int ch=fis_doc.read()), und, indem eine "exclusive-or"-Funktion auf "ch" ausgeführt wird, verschlüsselt. Dann wird das verschlüsselte Objekt in das Objekt "dos" geschrieben. Auf diese Weise wurden der auf den neuesten Stand gebrachte "envelope header", die Dokumentenkopfzeile und das verschlüsselte Dokument in die zusammenhängenden Speicherplätze geschrieben, indem die DataOutputStream Funktion verwendet wurde, und ein DS-Behälter, der das verschlüsselte Dokument enthält, wurde erzeugt. Dann wird der originale Behälter (envelopefilename) gelöscht und der auf den neuesten Stand gebrachte Behälter ("[envelopefilename].tmp") wird umbenannt in "envelopefilename". Auf diese Weise erscheint der Behälter als eine einzelne

Datei mit dem Namen, der im CNAME 71 Feld im Behälter enthalten ist, und beinhaltet die Behälterkopfzeile, die Dokumentenkopfzeile und das Dokument.

Im Folgenden wird eine Implementierung des DS-Behälters beschrieben, gemäß Tabellen 5(a) und 5(b). Tabelle 5(a) zeigt ein Programm zum Darstellen der Behälterkopfzeile eines DS-Behälters, erzeugt gemäß Tabellen 4(a) bis 4(c). Die Methode „view" beinhaltet zwei Parameter, den String "envelopename" und den PrintStream "ps". Falls weder "envelopename" noch "documentfilename" "Null" sind, und falls ein Behälter mit Namen "envelopename" existiert, wird ein Objekt "fis" des FileInputStream für "envelopefilename" und ein Objekt "dis" des DataInputStream für die Daten, die sich auf "envelopefilename" beziehen (DataInputStream(fis)), definiert. Auf diese Weise wird der Behälter "envelopename" in "fis" und "dis" geöffnet. Dann wird ein Objekt "eh" das SdeEnvelopeHeader erzeugt und die Inhalte von "dis" (das ist die Behälterkopfzeile für "envelopefilename") in "eh" durch die Methode "readFrom" eingelesen (Tabelle 4(c)). Zum Schluß wird die Information bezüglich der Behälterkopfzeile gedruckt, indem die Methode "println" verwendet wird.

Tabelle 5(b) zeigt ein Programm zum Öffnen eines DS-Behälters, erzeugt gemäß Tabellen 4(a) bis 4(d). Die Methode "extractDocument" beinhaltet zwei Parameter, die Strings "envelopefilename" und "documentfilename". Falls weder "envelopefilename" noch "documentfilename" "Null" sind, und falls ein Behälter mit Namen "envelopefilename" existiert, wird ein Objekt "fis" des FileInputStream für "envelopefilename" und ein Objekt "dis" des DataInputStream für die Daten, die sich auf "envelopefilename" beziehen (DataInputStream(fis)), definiert. Auf diese Weise wird der Behälter "envelopefilename" in "fis" und "dis" geöffnet. Dann wird ein Objekt "eh" das SdeEnvelopeHeader erzeugt und die Inhalte von "dis" (das ist die Behälterkopfzeile für "envelopefilename") in "eh" durch die Methode "readFrom" eingelesen (Tabelle 4(c)). Das Feld "d_name" wird dann mit "documentname" verglichen, und wenn sie übereinstimmen, überprüft das Programm das Verfallsdatum des Dokuments "documentname". Unter Verwendung der Datumsmethode "before" wird die Variable "valid" auf "False" gesetzt, wenn das Datum im Feld "d_expires" vor dem gegenwärtigen Datum ist. Falls nicht, wird der Pfad des Dokuments in ein Objekt "file_doc" der Klasse Files gesetzt.

Vorausgesetzt, daß das Dokument "documentname" existiert und das Dokument nicht verfallen ist, wird das Objekt "file_doc" auf einen neuen Pfad gesetzt: "[documentfilename].x", und das Objekt "fos_doc" des FileOutputStream wird auf FileOutputStream(file_doc) gesetzt. Auf diese Weise wird eine vorläufige Datei "fos_doc" der Klasse FileOutputStream erzeugt. Dann wird das Dokument Byte für Byte in die Variable "ch" eingelesen. Jedes Byte wird durch Anwendung der XOR Funktion auf "ch" entschlüsselt und in "fos_doc" geschrieben. Wenn dieses Dokument nicht existiert, wird "fos_doc" auf den Wert Null gesetzt und das Programm zeigt an, daß die Datei nicht gefunden wurde. Wenn das Dokument existiert, aber abgelaufen ist, wird die Variable "valid" auf FALSE gesetzt. Dies verhindert, daß "fos_doc" mit dem Dokument assoziiert wird, daß das Dokument entschlüsselt wird und veranlaßt das Programm anzuzeigen, daß das Dokument abgelaufen ist. Um das Dokument zu überschreiben um es zu zerstören, können die folgenden Anweisungen in Tabelle 5b eingefügt werden:

```
If(valid=False)
{
for (long 1=0, 1<dh.d_total_bytes; 1++)
{
    int ch=0
    fos_doc.write(ch)
}
}
```

Figuren 11(a) und 11(b) zeigen eine einfache graphische Benutzerschnittstelle, die dazu verwendet werden kann, sowohl den DS-Behälter eines einzelnen Dokuments, wie oben bezüglich Tabellen 4 und 5 beschrieben, und den DS-Behälter für mehrere Dokumente, wie unten bezüglich Figuren 12 und 13 beschrieben, zu implementieren. Um einen DS-Behälter zu erzeugen, drückt ein Benutzer den "Neu"-Knopf 1510. Es wird dann eine Dialogbox erscheinen (nicht dargestellt), die den Benutzer auffordert, einen Behälternamen einzugeben. Sobald der Behältername eingegeben ist, wird ein DS-Behälter erzeugt und der Behältername wird im Banner 1500 erscheinen. In Figuren 11(a) und 11(b) ist der Behältername "demo.env" und der Behälter ist als Datei "demo.env" im Verzeichnis C:\ gespeichert. Um dem DS-Behälter ein neues Dokument hinzuzufügen, drückt ein Benutzer auf den "Hinzufügen"-

Knopf 1520. Es wird dann ein Dateimenü erscheinen (nicht dargestellt), das es dem Benutzer erlaubt, entweder die zur Verfügung stehenden Verzeichnisse nach dem gewünschten Dokument zu durchsuchen, oder Pfad und Dokumentennamen direkt einzugeben. Sobald das Dokument gewählt wurde und ein Verfallsdatum in Kasten 1540 eingegeben wurde, wird das Dokument dem Behälter hinzugefügt. Sobald das Dokument dem Behälter erfolgreich hinzugefügt wurde, werden Dokumentenname, Dokumentenlänge und Verfallsdatum des Dokuments in Kasten 1550 dargestellt. Figur 11(b) zeigt, daß, falls der Behälter "demo.env" nach dem Verfallsdatum des Dokuments "demo.txt" geöffnet wird, das Dokument automatisch mit aussagelosen Zeichen überschrieben wird und eine Nachricht in Kasten 1550 erscheint, die anzeigt, daß das Dokument abgelaufen ist. Um ein Dokument aus dem Behälter zu extrahieren, wird das Dokument durch Drücken auf den Dokumentenkasten 1550 und anschließendem Drücken auf den Kasten "Extrahieren" 1530 ausgewählt. Das Dokument wird dann aus dem Behälter extrahiert und im zugewiesenen Verzeichnis auf dem Computer gespeichert. Wenn ein Dokument abgelaufen ist, wird es das System nicht erlauben, daß das Dokument in Kasten 1550 ausgewählt wird.

Des weiteren kann, gemäß einer weiteren Ausführungsform des DS-Behälters, der Behälter expandiert und komprimiert werden, um mehrere Dokumente und andere digitale Objekte zu enthalten. Neue Dokumente (oder andere digitale Objekte) können dem Behälter hinzugefügt werden, im Behälter existierende Dokumente können auf den neuesten Stand gebracht oder gelöscht werden, und mehrere Behälter können verschmolzen werden. Jedes einzelnen Dokument im Behälter kann unabhängige Lebenszeitkontrollen und verschiedene Sicherheitsstufen besitzen.

Um diese Funktionalität zu gewährleisten, wird der DS-Behälter als ein virtuelles Dateisystem in einer einzigen Datei behandelt. Ein virtuelles Dateisystem ist eine Speichereinheit, die der externen Welt als einzelne Datei erscheint, wobei aber seine interne Repräsentation in Wahrheit viele Dateien und ihre Kontrollinformation verwaltet. Obwohl virtuelle Dateisysteme normalerweise hierarchische Verzeichnisstrukturen beinhalten, wird für den DS-Behälter vorzugsweise eine flache – einzel-Datei – Struktur verwendet. Gemäß der bevorzugten Ausführungsform der Erfindung, unterstützt der DS-Behälter die folgenden Operationen: i) Erzeugen eines neuen DS-Behälters; ii) Hinzufügen eines neuen Dokuments mit unabhängigen Lebenszeitkontrollen und Sicherheitsstufe zum DS-Behälter; iii) auf den

neuesten Stand Bringen des Inhalts eines existierenden Dokuments im DS-Behälter, ohne Lebenszeitkontrollen und Sicherheitsstufe zu ändern; iv) auf den neuesten Stand Bringen der Lebenszeitkontrollen oder der Sicherheitsstufe eines existierenden Dokuments im DS-Behälter; v) Löschen eines existierenden Dokuments aus dem DS-Behälter; vi) ungültig Erklären eines existierenden Dokuments im DS-Behälter auf seine Lebenszeitkontrollen und Löschen dieses Dokuments aus dem DS-Behälter; und vii) Zerstören des DS-Behälters.

Gemäß dieser Ausführungsform beinhaltet ein DS-Behälter eine Behälterkopfzeile, die die Kontrollinformation für den Behälter enthält, und eine Dokumentenkopfzeile für jedes Dokument im Behälter.

Figur 10(b) zeigt gemäß einer bevorzugten Implementierung dieser Ausführungsform die Behälterkopfzeile, die den Namen CNAME 801 des Behälters, die voreingestellten Lebenszeitkontrollen DLC 802 des Behälters, das Datum DC 803, an dem der Behälter erzeugt wurde, das Datum DM 804, an dem der Behälter zuletzt modifiziert wurde, das Datum DLA 805, an dem auf den Behälter zuletzt zugegriffen wurde, und die Anzahl 806 an Dokumenten im Behälter enthält. Figur 10(b) zeigt einen DS-Behälter, der zwei Dokumente 904.1 und 904.2 enthält. Jedes Dokument hat eine Kopfzeile, die den Namen des Dokuments (901.1 oder 901.2) enthält, die Lebenszeitkontrollen für das Dokument (902.1 oder 902.2), und die Länge des Dokuments (903.1 oder 903.2). Gemäß dieser Ausführungsform wird DLC 802, die im "envelope header" enthalten ist, verwendet als voreingestellte Lebenszeitkontrolle, wenn Dokumente dem Behälter hinzugefügt werden. Die DC 803, DM 804 und DLA 805 Felder ermöglichen es dem Benutzer, das Datum, an dem der Behälter erstellt wurde, das Datum, an dem er zuletzt modifiziert wurde und das Datum, an dem auf ihn zuletzt zugegriffen wurde, festzustellen, indem eine Routine ähnlich der "view-Methode" in Tabelle 5(a) implementiert wird. Natürlich können voreingestellte Lebenszeitkontrolle, die Behälter erzeugt-, zuletzt modifiziert- und zuletzt darauf zugegriffen-Felder aus der Behälterkopfzeile ausgelassen werden, wenn diese Funktionalität nicht gewünscht wird.

Da der DS-Behälter aus Figur 10(b) auf ähnliche Weise wie der DS-Behälter aus Figur 10(a) und Tabellen 4(a) bis 5(b) implementiert wird, wir hier keine detaillierte Beschreibung, wie diese Ausführungsform programmiert werden kann, gegeben.

Das Flußdiagramm aus Figur 12 zeigt, wie der existierende DS-Behälter geöffnet wird (Schritt 1010) und die Behälter-Kopfzeile (die Behältername, voreingestellte Lebenszeitkontrollen, Erzeugungsdatum, Datum der letzten Modifikation, Datum des letzten Zugriffs und Anzahl der Dokumente enthält) gelesen wird, um einem existierenden DS-Behälter ein Dokument hinzuzufügen. Dann wird in Schritt 1030 eine Kopfzeile für das neue hinzuzufügende Dokument erstellt, die Dokumentnamen, Lebenszeitkontrollen des Dokuments und Dokumentenlänge enthält. Dann werden Dokumentenkopfzeile und das Dokument selbst an das Ende des Behälters angehängt (Schritte 1040 und 1050), die Werte für die Felder "letzter Zugriff", "letzte Modifikation" und "Gesamtzahl an Dokumenten" auf den neuesten Stand gebracht (Schritt 1060) und die neuen Werte als die neue Behälter-Kopfzeile geschrieben. Gemäß dem Flußdiagramm in Figur 12 wird der ursprüngliche Behälter mit dem auf den neuesten Stand gebrachten Behälter überschrieben. Dies geschieht in Kontrast zur Ausführungsform der Tabellen 4(a) bis 5(b), in denen der auf den neuesten Stand gebrachte Behälter als neue Datei erzeugt wird und der ursprüngliche Behälter nachher gelöscht wird. Natürlich kann die Ausführungsform der Figuren 10(b) und 13 derart modifiziert werden, daß sie der Methode der Tabellen 4(a) bis 5(b) entspricht (d. h. Überschreiben des ursprünglichen Behälters) und umgekehrt.

Das Flußdiagramm aus Figur 13 zeigt, daß der Behälter geöffnet wird (Schritt 1110) und die Behälter-Kopfzeile gelesen wird, um das Dokument aus dem DS-Behälter in Figur 10(b) zu extrahieren. Dann wird die Dokumentenkopfzeile für das erste Dokument im Behälter in Schritt 1120 aus dem Behälter gelesen. In Schritt 1130 wird der Name des zu extrahierenden Dokuments mit dem Namen im Feld dname 901 verglichen, und wenn beide übereinstimmen, fährt das Programm mit Schritt 1140 fort. Wenn sie nicht übereinstimmen, fährt das Programm mit Schritt 1150 fort. Vorausgesetzt, der gegenwärtige Dokumentenname im Feld dname ist derselbe, wie der des erwünschten Dokuments, überprüft das Programm in Schritt 1140, ob die Lebenszeitkontrollen für das Dokument gültig sind (z. B. ob das gegenwärtige Datum später ist als das Verfallsdatum). Wenn die Lebenszeitkontrollen gültig sind, wird das gegenwärtige Dokument extrahiert. Wenn die Lebenszeitkontrollen nicht gültig sind, wird das Dokument gelöscht. Unten wird ein veranschaulichendes Codesegment zum Löschen des Dokuments angegeben, worin die Funktion getTotalBytes() die Dokumentlänge aus der Dokument-Kopfzeile des gegenwärtigen Dokuments erhält.

```
Public void nullifyDocumentContent(SdeDocumentHeader dh)
        throws IOException
    {
        long bytesToNullify = dh.getTotalBytes();
        for (long l=0; l<bytesToNullify; l++)
        {
            write(0);
        }
    }
```

Wenn das gegenwärtige Dokument nicht das gewünschte ist, überspringt das Programm in Schritt 1150 den Inhalt des gegenwärtigen Dokuments zur nächsten Dokument-Kopfzeile, oder zum Ende des "envelope". Verdeutlichender Code zur Implementierung dieser Funktion ist der folgende:

```
Public void skipDocumentContent(SdeDocumentHeader dh)
        throws IOException
    {
        long bytesToSkip = dh.getTotalBytes()
        for (long l = 0; I<bytesToSkip; l++)
        {
            int ch = read();
        }
    }
```

Auf diese Weise liest das Programm bis hinter das Ende des gegenwärtigen Dokuments, so daß es bereit ist, die Dokument-Kopfzeile des nächsten Dokuments im Behälter zu lesen, wenn das Programm zu Schritt 1120 zurückkehrt.

Nachdem das Dokument extrahiert wurde, wird das Feld in der Behälter-Kopfzeile, auf das zuletzt zugegriffen wurde, auf den neuesten Stand gebracht (Schritt 1160), und die Behälter-Kopfzeilenfelder 801 bis 806 werden mit der auf den neuesten Stand gebrachten Behälter-Kopfzeile überschrieben (Schritt 1170) und der Behälter geschlossen (Schritt 1180).

TABELLEN 4 BIS 5(B)

```
public static int create(String envelopefilename)
{
if( envelopefilename == null)
{
                    return(CREATE_FA1L_BAD_PARAM);
}
if ( envelopeExists(envelopefilename))
{
                    return(CREATE_FA1L_BAD_PARAM);
}
try
{
  FileOutputStream fos = new FileOutputStream(envelopefilename);
  DataOutputStream dosnew DataOutputStream(fos);
  SdeEnvelopeHeader eh = new SdeEnvelopeHeader();
  eh.d_name          = envelopefilename;
  eh.d_created       = new Date();
  eh.d_last_modified = new Date();
  eh.d_last_accessed = new Date();
  eh.writeTo(dos);
  dos.flush();
  fos.close();
```

TABELLE 4A

```
public static int addDocument(
   String envelopefilename,
   String documentfilename,
   Date expires
)
```

```
    if( envelopefilename == null ||
        documentfllename == null ||
        expires == null)
{
        retum(ADD_FAIL_BAD_PARAM);
}
    if ( enveIopeExists(envelopefilename) == false)
{
                        create(envelopefilename);
}
    if ( documentExists(documenttfilename) == false)
{
                        return(ADD_FAIL_BAD_PARAM);
}
try
        FileInputStream fis = new FileInputStream(envelopefilename);
        DataInputStream dis = new DataInputStream(fis);
        FileOutputStream fos = new FileOutputStream(envelopefilename + ".tmp");
        DataOutputStream dos = new DataoutputStream(fos);
        SdeEnvelopeHeader eh = new SdeEnvelopeHeader();
        eh.readFrom(dis);
        eh.d_last_modifled = new Date();
        eh.d_total_documents++;
        eh.writeTo(dos);
        File file_doc = new File(documentfilename);
        long length = file_doc.length();
        FileInputStream fis_doc = new FileInputStream(file_doc);
        SdeDocumentHeader dh = new SdeDocumentHeader();
        dh.d_name = documentfilename;
        dh.d_expires = expires;
        dh.d_totaI_bytes = length;
        dh.writeTo(dos);
        while(true)
```

```
    {
                int ch = fis_doc.read();
                if(ch == -1)
                                break;
                ch ^= 65;
                dos.write(ch);
    }
    fis_doc.close();
    dos.flush();
    fos.close();
    fis.close();
    File file_orig = new File(envelopefilename);
    File_orig.delete();
    File file_new = new File(envelopefilename + ".tmp");
    file_new.renameTo(file_orig);
}
catch ( IOException ioe)
{
                return(ADD_FAIL_IOEXCEPTION);
}
return(ADD_OK);
}
```

TABELLE 4B

```
class SdeEnvelopeHeader
{
            public String       d_name              = null;
            public Date         d_created           = null;
            public Date         d_last_modified     = null;
            public Date         d_last_accessed     =null;
            public long         d_totaldocuments    =0;
            public boolean writeTo(DataQutputStream dos)
```

```
                throws IOException
        {


        dos.writeUTF(d_name);
                dos.writeLong(d_created.getTime());
                dos.writeLong(d_last_modified.getTime());
                dos.writeLong(d_last_accessed.getTime());
                dos.writeLong(d_total_documents);
                retum(true);
        }
    public boolean readFrom(DataInputStream dis)
        throws IOException

        {

        d_name = dis.readUTF();
        long c = dis.readLong();
        d_createcd = new Date(c);
        long lm = dis.readLong();
        d_last_modified = new Date(lm);
        long la = dis.readLong();
        d_last_accessed = new Date(la);
        d_total_documents = dis.readLong();
        retum(true);
    }
}
```

TABELLE 4C

```
class SdeDocumentHeader
{

        public String                   d_name              = null;
        public Date                     d_expires           = null;
        public long                     d_total_bytes       =0;
        public boolean writeTo(DataOutputStream dos)
                throws lOException
        {
                                dos.writeUTF(d_name);
                                dos.writeLong(d_expires.getTime());
                                dos.writeLong(d_total_bytes);
                                return(true);
        }
 public boolean readFrom(DataInputStream dis)
          throws lOException
        {

                                d_name = dis.readUTF();
                                long e = dis.readLong();
                                d_expires = new Date(e);
                                d_total_bytes = disreadLong();
                                return(true);
          }
 }
```

TABELLE 4D

```
   public static int VIEW_OK =0;
   public static int VIEW_FAIL_BAD_PARAM =1;
   public static int VIEW_FAIL_IOEXCEPION =2;
```

```
public static int view(String envelopefilename, PrintStream ps)
{
                    if( envelopefilename == null || ps == null)
                    {
                            return(VIEW_FAIL_BAD_PARAM);
                    }
                    if ( envelopeExists(envelopefilename) == false)
                    {
                            return(EXTRACT_FAIL_BAD_PARAM);
                    }
                    try
                    {
                    FileInputStream fis = new FileInputStream(envelopefilename);
                    DataInputStream dis = new DataInputStream(fis);
                    SdeEnvelopeHeader eh = new SdeEnvelopeHeader();
                    eh.readFrom(dis);
                    ps.println("Envelope: "'+ eh.d_name +'".");
                    ps.println("Created:  "+ eh.d_created.toString());
                    ps.println("Last modified: "+ eh.d_last_modified.toString());
                    ps.println("Last accessed: "+ eh.d_last_accessed.toString());
                    ps.println("Contains: "+ eh.d_total_documents +"
        document(s).");
                fis.close();
                    }
                    catch ( IOException ioe)
                    {
                            return(VIEW_FAIL_IOEXCEPTION);
                    }
                    return(VIEW_OK);
```

TABELLE 5A

```
public static int extractDocument(String envelopefilename, String documentfilename)
```

```
{
        if ( envelopefilenarne == null || documentfilename == null)
        {
    return(EXTRACT_FAIL_BAD_PARAM);
        }
    if ( envelopeExists(envelopefilenarne) == false)
    {
        return(EXTRACT_FAIL_BAD_PARAM);
        }
        boolean valid          = true;
        int     invalid_reason = EXTRACT_FAIL_UNKNOWN;
    try
        {
            FileinputStream fis = new
 FileInputStream(envelopefilename);
            DataInputStream dis = new DataInputStream(fis);
            SdeEnvelopeHeader eh = new SdeEnvelopeHeader();
            eh.readFrom(dis);
          SdeDocurnentReader dh = new SdeDocumentHeader();
            dh.readFrom(dis);
       if ( dh.d_namc.equalsIgnoreCase(documentfilename) =
    false)
            {

        valid = false;
        invalid_reason
EXTRACT_FAIL_NOT_FOUND
        }
            if( dh.d_expires.before(new Date()))
            {
                valid = false;
                invalid_reason = EXTRACT_FAIL_INVALID;
            }
```

```
        File file_doc =  new File(documentfilename);
        if( file_doc.exists())
        {
            file_doc  new FiIe(documentfilename + = ".x");
        }
        FileOutputStream fos_doc = null;
        if( valid)
        {
                fos_doc = new FileOutputStream(file_doc);
        }
        for (long l = 0; 1 < dh.d_total_bytes; 1++)
        {
                int ch = disread();
                if( valid)
{
          ch ^= 65;
                        fos_doc.write(ch);
                 }
        }
         if(valid)
      {
                fos_doc.close();
      }
      fis.close();
 }
 catch ( IOException ioe)
 {
      return(EXTRACT_FAIL_IOEXCEPTION);
}
if( valid)
{
        return(EXTRACT_OK);
```

```
        }
        else
        {
                return(invalid_reason);

        }
}
```

TABELLE 5B

Es sollte erwähnt werden, daß, obgleich oben dargestellten Ausführungsformen für ihre Annäherung an die Idee der virtuellen Behälter Java verwenden, und zwar mit einer relativ einfachen graphischen Benutzerschnittstelle, einer einfachen Verschlüsselungstechnik und Lebenszeitkontrollen, die ein Verfallsdatum beinhalten, die Erfindung eine große Zahl an weiteren Implementierungen einschließt, die einfacher oder auch komplexer sein können. So kann die Benutzerschnittstelle eine komplexere GUI oder eine einfache Kommandozeilen-Schnittstelle ohne Graphik beinhalten. Genauso können die virtuellen Behälter implementiert werden, indem Serialisationsmechanismus von Java oder eine völlig andere Programmiersprache wie C^{++} verwendet wird. Auch können die Lebenszeitkontrollen modifiziert werden, so daß sie eine Funktion der Anzahl an Extraktionen des Dokuments oder der Anzahl an Modifizierungen des Dokuments darstellen. Auch können die Lebenszeitkontrollen verwendet werden, um das Dokument nach einer spezifizierten Zeitperiode in ein "read-only", also ein nur lesbares, Dokument umzuwandeln, oder um zu verhindern, daß das Dokument kopiert oder gedruckt wird.

Internethandel-Anwendung von virtuellen Behältern

Gemäß einer anderen Ausführungsform der Erfindung wird ein Internethandelsystem zur Verfügung gestellt, das die virtuellen Behälter verwendet. Gemäß dieser Ausführungsform legt eine Partei, die ein elektronisch übersendbares Produkt über das Internet verkaufen möchte, das Produkt in einen virtuellen Behälter, indem sie eine Behälter-Erzeuger-

Anwendung verwendet, die das Produkt verschlüsselt und Lebenszeitkontrollen für das Produkt setzt. So kann es sein, daß der Verkäufer es wünscht, daß ein potentieller Käufer das Produkt für eine begrenzte Probezeit ansehen oder auf andere Weise verwenden kann, und daß das Produkt nach dieser Probezeit zerstört wird, falls es nicht gekauft wurde. Ein potentieller Käufer des Produkts, der, bevor er das Produkt kauft, es ausprobieren möchte, erhält von Verkäufer eine Kopie des Behälters zusammen mit einer Behälter-Öffner-Anwendung. Die Behälter-Öffner-Anwendung gestattet es dem Käufer, das Produkt anzuschauen oder zu benutzen, während es im Behälter gewartet wird. Die Behälter-Öffner-Anwendung kann so konfiguriert werden, daß es dem Benutzer nicht erlaubt ist, das Produkt (oder Teile davon) auszudrucken, zu kopieren oder zu modifizieren. Wenn der Käufer versucht, das Produkt ohne die Behälter-Öffner-Anwendung anzuschauen, wird sie/er unfähig sein, es zu entschlüsseln. Auf jeden Fall wird jeder Versuch, den Behälter mit dem Behälter-Öffner zu öffnen, nachdem die Lebenszeitkontrollen ungültig wurden (z. B. nach einer bestimmten, durch den Verkäufer voreingestellten Zeitdauer), zu einer Zerstörung des Produkts führen. Es soll erwähnt werden, daß das Produkt jede Form von elektronischen Medien, die in digitaler Form übersendet werden können, sein kann, wie zum Beispiel Dokumente, Photographien, Bilder und Programme.

Gemäß einem weiteren Aspekt dieser Ausführungsform kann der Käufer das Produkt vor Ablauf der Lebenszeitkontrollen kaufen, indem sie/er dem Verkäufer ihre/seine Kreditkarteninformation übermittelt. Nach Erhalt und/oder Überprüfung der Zahlungsinformation übermittelt der Verkäufer dem Käufer den Kaufschlüssel. Der Behälter-Öffner ist konfiguriert, diesen Kaufschlüssel zu erkennen (der vorzugsweise einzigartig ist und nur auf dieses, diesem einen Käufer übermittelte Produkt paßt), und es dem Käufer zu gestatten, das Produkt aus dem Behälter zu extrahieren, wenn der Schlüssel gültig ist.

Verwendung von Softwarebestandteilen für virtuelle Behälter und eingebettete ausführbare Module

In jedem der oben dargelegten exemplarischen Ausführungsbeispiele wurden die sich selbst zerstörenden Dokumente in anwendungsspezifische Programme implementiert, so daß zur Implementierung eines sich selbst zerstörenden Word™-Dokuments ein anderes Programm verwendet wurde, als für ein sich selbst zerstörendes Excel™-Dokument. Um diesen sich

wiederholenden (rück-)entwickelnden Prozeß zu vermeiden, kann das sich selbst zerstörende Dokument, oder, in diesem Fall, das virtuelle Behältersystem, als Softwarekomponente implementiert werden. Eine Softwarekomponente ist eine leicht wieder zu verwendende Softwareeinheit, die typischer Weise einen einfachen Service bereitstellt.

Das Computer Objekt Model (COM) stellt den Microsoft Standard für Software-Komponententechnologie dar. Es definiert einen Standard zum Packen von Software für den einfachen Wiedergebrauch. Eine typische COM-Komponente enthält inhaltlich zwei breite Kategorien: die Funktionalität der Komponente, implementiert als Satz von Methoden (Funktionen), und beschreibende Information über die Komponente und ihrer Funktionalität. Dieser Inhalt wird die Schnittstelle der Komponente genannt. Es soll erwähnt werden, daß dieser Begriff anders als der Begriff „Benutzerschnittstelle“ verwendet wird. Eine Benutzerschnittstelle ist typischerweise eine graphische Anordnung von Fenstern, Menüs, Knöpfen und dergleichen, die es dem Benutzer erlauben, mit der Funktionalität einer Anwendung Wechsel zu wirken. Eine "Schnittstelle" allerdings bezieht sich ganz allgemein auf den Eingabepunkt/die Eingabepunkte einer Softwareeinheit (Methoden, Funktionen).

COM-Komponenten können verwendet werden, um Benutzerschnittstellen-Elemente, wie Knöpfe und Listen, zu implementieren. Sie können auch benutzt werden, um Services ohne Benutzerschnittstelle zur Verfügung zu stellen. Zum Beispiel kann eine COM-Komponente mathematische Funktionen zur Verfügung stellen, um Sinus- und Cosinusfunktionen zu berechnen.

Obwohl es feine technische Unterschiede zwischen Komponenten und Objekten gibt, stellen sie zwei sehr ähnliche Konzepte dar. So verwendete dieses Dokument diese Termini austauschbar, gemäß den Sprachgewohnheiten der Presse und Benutzerhandbüchern.

Des weiteren wird der Terminus "Kontrolle" oft verwendet, um eine Komponente zu beschreiben, die keine Benutzerschnittstelle besitzt. Insbesondere verwendet Microsoft den Terminus ActivX-Kontrolle, um eine spezielle Klasse von COM-Komponenten zu beschreiben, die typischerweise Benutzerschnittstellen besitzen. Wenn eine COM-Komponente verwendet wird, um die Funktionalität einer bestehenden Anwendung auszuweiten, spricht man von einer Einbettung der Komponente in das Dokument der

Anwendung. Eine solche Komponente wird oft als eingebettetes Objekt bezeichnet. Diesbezüglich weisen wir darauf hin, daß Word 97™, Excel ™ und Outlook 98™ jeder eine Unterstützung für Visual Basic-Anwendungen in Form einer Bibliothek an COM-Objekten besitzen.

Anwendungsausweitungen, implementiert mit Techniken der eingebetteten Objekte, erfordern im allgemeinen COM-Komponenten, die auf dem Computer des Verfassers des Dokuments installiert sind. Des weiteren muß im allgemeinen jeder Benutzer, der eine Kopie des Dokuments erhält, auch die COM-Komponenten auf seinem Computer installiert haben, um die ausgeweitete Funktionalität zu verwenden.

Grob gesagt kann eine COM-Komponente für einen virtuellen Behälter wie folgt implementiert werden:

1. Spezifikation der Schnittstelle für die Komponente. Die Schnittstelle beschreibt jeden der von der Komponente angebotenen Service. Im Fall einer SDE-Komponente kann der Service CreativeEnvelope, AddDocumentToEnvelope und ExtractDocumentFromEnvelope enthalten.

2. Implementierung jedes einzelnen Service der Komponente unter Verwendung einer Programmiersprache wie Java, C++ oder Visual Basic.

3. Verwendung des Entwicklungswerkzeugs der Programmiersprache, um die COM-Komponente zu erzeugen.

4. Installation der COM-Komponente auf der Workstation des Benutzers. Implementierung der wenigen Skripte zum Aufruf der Services der Komponente.

Allerdings sollte erwähnt werden, daß hier der Microsoft COM-Komponenten-Standard beschrieben wurde, aber auch Softwarestandards, die von anderen Herstellern veröffentlicht werden, verwendet werden können.

Obgleich hier die gegenwärtig bevorzugten Ausführungsformen der Erfindung beschrieben wurden, wird der Fachmann erkennen, daß Änderungen und Modifikationen vorgenommen werden können, ohne vom Grundgedanken der Erfindung abzuweichen. Es liegt in unserer Intention, alle solchen Modifikationen, die in den Rahmen dieser Erfindung fallen, zu beanspruchen.

ANSPRÜCHE

1. Virtuelles Behältersystem, umfassend:

eine CPU (bzw. eine entsprechend funktionelle Hardware-Vorrichtung);

einen an die CPU angebundenen elektronischen Speicher, der das verschlüsselte digitale Objekt, das in einem virtuellen Behälter enthalten ist, und den virtuellen Behälter, der eine Information über eine Lebenszeitkontrolle des digitalen Objekts umfaßt, enthält;

einen Schlüssel;

ein Computerprogramm, das von der CPU (bzw. der Hardware-Vorrichtung) ausgeführt werden kann, um das digitale Objekt zu entschlüsseln, falls die Lebenszeitkontrolle gültig ist, und um die Lebenszeitkontrolle nach dem Erhalt eines Schlüssels zu modifizieren.

2. Virtuelles Behältersystem nach Anspruch 1,
dadurch gekennzeichnet, daß der Schlüssel in einem vom elektronischen Speicher separaten Speicherplatz aufbewahrt wird und dem Programm nicht zur Verfügung steht.

3. Virtuelles Behältersystem nach Anspruch 2,
dadurch gekennzeichnet, daß der Schlüssel dem Programm als Folge einer Anfrage des Benutzers zur Verfügung gestellt wird.

4. Virtuelles Behältersystem nach Anspruch 3,
dadurch gekennzeichnet, daß der Schlüssel ein handelsüblicher Schlüssel ist und die Anfrage eine Bezahlung miteinschließt.

5. Virtuelles Behältersystem nach Anspruch 4,
dadurch gekennzeichnet, daß die Bezahlung in Form von Kreditkarteninformation geschieht.

6. Virtuelles Behältersystem nach Anspruch 1,
dadurch gekennzeichnet, daß die Lebenszeitkontrolle aus einem Verfallsdatum besteht.

7. Virtuelles Behältersystem nach Anspruch 6,
dadurch gekennzeichnet, daß besagte Lebenszeitkontrolle modifiziert wird, indem das Verfallsdatum gelöscht wird, so daß das digitale Objekt nicht verfällt.

8. Virtuelles Behältersystem nach Anspruch 6,
dadurch gekennzeichnet, daß besagte Lebenszeitkontrolle modifiziert wird, indem das Verfallsdatum geändert wird.

9. Virtuelles Behältersystem nach Anspruch 1, weiterhin bestehend aus:
einer Behälter-Erzeuger-Anwendung, die einen virtuellen Behälter erzeugen kann, der einen Kopfzeilenteil, der die Information für die Lebenszeitkontrolle trägt, und einen Teil für ein digitales Objekt, der das digitale Objekt enthält, umfaßt; und
eine Behälter-Öffner-Anwendung, die betätigt werden kann, um zu bestimmen, ob die Lebenszeitkontrolle gültig ist.

10. Virtuelles Behältersystem nach Anspruch 9,
dadurch gekennzeichnet, daß der Behälter-Erzeuger das digitale Objekt verschlüsseln und das verschlüsselte digitale Objekt in den Teil des virtuellen Behälters, der für ein digitales Objekt vorgesehen ist, schreiben kann.

11. Virtuelles Behältersystem nach Anspruch 10,
dadurch gekennzeichnet, daß Behälter-Öffner das verschlüsselte digitale Objekt lesen und das digitale Objekt entschlüsseln kann, wenn die Lebenszeitkontrollen gültig sind.

12. Virtuelles Behältersystem nach Anspruch 9,
dadurch gekennzeichnet, daß es eine Software enthält, die die Inhalte des digitalen Objekts darstellen kann.

13. Virtuelles Behältersystem nach Anspruch 9,
dadurch gekennzeichnet, daß die Behälter-Öffner-Anwendung das digitale Objekt zerstören kann, wenn die Lebenszeitkontrolle nicht gültig ist.

14. Virtuelles Behältersystem nach Anspruch 9,
dadurch gekennzeichnet, daß die Behälter-Öffner-Anwendung das digitale Objekt mit aussagelosen Zeichen überschreiben kann, wenn die Lebenszeitkontrolle nicht gültig ist.

15. Virtuelles Behältersystem nach Anspruch 1,
dadurch gekennzeichnet, daß der virtuelle Behälter auf zusammenhängenden Speicherplätzen im elektronischen Speicher gespeichert ist.

16. Virtuelles Behältersystem nach Anspruch 1,
dadurch gekennzeichnet, daß die Lebenszeitkontrolle aus einem Verfallsdatum und einer Verfallszeit besteht.

17. Virtuelles Behältersystem nach Anspruch 1,
dadurch gekennzeichnet, daß die Lebenszeitkontrolle aus der Anzahl, die das digitale Objekt geöffnet wurde, besteht.

18. Ein System zum Übersenden eines verfallenden digitalen Objekts an einen Empfänger, umfassend Software mit folgenden Funktionen:

Erzeugen eines digitalen Behälters mit einem Kopfzeilenteil und einem Teil für ein digitales Objekt;

Verschlüsseln eines digitalen Objekts;

Schreiben des verschlüsselten digitalen Objekts in den Speicherbereich für das digitale Objekt;

Auswahl einer Lebenszeitkontrolle für das digitale Objekt;

Schreiben der Information für die Lebenszeitkontrolle in den Kopfzeilenteil des virtuellen Behälters;

Übersenden des virtuellen Behälters an einen Empfänger;

sowie entsprechend betätigbare Hardware.

19. System nach Anspruch 18, mit folgenden weiteren Funktionen:

Lesen der Lebenszeitkontrolle;

Entschlüsseln des digitalen Objekts, wenn die Lebenszeitkontrolle gültig ist.

20. System nach Anspruch 19, mit folgenden weiteren Funktionen:

zur Verfügung stellen eines Schlüssels;

Verwenden des Schlüssels, um die Lebenszeitkontrolle zu modifizieren.

21. System nach Anspruch 20, mit folgender weiterer Funktion:

Akzeptieren einer Bezahlung vor zur Verfügung stellen des Schlüssels.

22. System nach Anspruch 21, bei dem die Verwendung des Schlüssels eine Elimination der Lebenszeitkontrolle beinhaltet.

23. System nach Anspruch 21, bei dem die Verwendung des Schlüssels eine Änderung der Lebenszeitkontrolle beinhaltet.

24. System nach Anspruch 19, bei dem das digitale Objekt zerstört wird, wenn die Lebenszeitkontrolle nicht gültig ist.

25. System nach Anspruch 23, bei dem der Zerstörungsschritt das Überschreiben des digitalen Objekts mit aussagelosen Zeichen beinhaltet.

26. System nach Anspruch 18, bei dem der Übersendungsschritt das Übersenden des digitalen Objekts über das Internet beinhaltet.

27. System nach Anspruch 18, bei dem die Software folgende weitere Funktionen beinhaltet:

Erzeugung einer Behälterkopfzeile, die Information bezüglich des Behälters einschließlich des Behälternamens enthält;

Erzeugung einer Kopfzeile für jedes einzelne digitale Objekt aus einer Reihe digitaler Objekte, wobei die Kopfzeile Information bezüglich des digitalen Objekts einschließlich des Namens des digitalen Objekts enthält;

Speicherung jedes digitalen Objekts neben seiner Kopfzeile im virtuellen Behälter.

28. System nach Anspruch 26, bei dem das Schreiben der Information bezüglich der Lebenszeitkontrolle für jedes digitale Objekt das Schreiben der Information für das digitale Objekt in die Behälterkopfzeile beinhaltet.

29. System nach Anspruch 26, bei dem das Schreiben der Information bezüglich der Lebenszeitkontrolle für jedes digitale Objekt das Schreiben der Information in die Kopfzeile für das digitale Objekt beinhaltet.

30. System nach Anspruch 18, bei dem der Erzeugungsschritt folgenden weiteren Schritt beinhaltet:

Erzeugung des virtuellen Behälters auf zusammenhängenden Speicherplätzen in einem elektronischen Speicher.

31. System zum Extrahieren von Information aus einem virtuellen Behälter, das folgende Software-Funktionen beinhaltet:

Lesen von Information bezüglich der Lebenszeitkontrolle aus einem Kopfzeilenteil eines virtuellen Behälters, wobei der virtuelle Behälter den Kopfzeilenteil und den Teil für das digitale Objekt beinhaltet, und wobei der Teil für das digitale Objekt ein verschlüsseltes digitales Objekt beinhaltet;

Überprüfen der Gültigkeit der Lebenszeitkontrolle, basierend auf die Information;

Lesen des digitalen Objekts aus dem Teil für das digitale Objekt und Entschlüsseln des digitalen Objekts, wenn die Lebenszeitkontrolle gültig ist.

32. System zum Übersenden eines digitalen Objekts, das zerstört werden kann, an einen Empfänger, das folgende Software-Funktionen und entsprechend betreibbare Hardware beinhaltet:

Erzeugen eines virtuellen Behälters, der auf zusammenhängenden Speicherplätzen in einem elektronischen Speicher eines Computers gespeichert ist und einen Kopfzeilenteil und einen Teil für ein oder mehrere digitale Objekte beinhaltet;

Auswahl eines digitalen Objekts zur Einfügung in den virtuellen Behälter;

Anwenden einer Verschlüsselungstechnik auf das digitale Objekt, um ein verschlüsseltes digitales Objekt zu erzeugen;

Schreiben des verschlüsselten digitalen Objekts in den Teil für das digitale Objekt;

Auswahl eines Verfallsdatums für das digitale Objekt;

Schreiben von Information bezüglich des Verfallsdatums in den Kopfzeilenteil des virtuellen Behälters;

Übersenden des virtuellen Behälters und der Behälter-Öffner-Anwendung an einen Empfänger, wobei die Behälter-Öffner-Anwendung, wenn sie vom Empfänger aufgerufen wird, die Information bezüglich des Verfallsdatums aus dem Kopfzeilenteil des virtuellen Behälters liest, bestimmt, basierend auf besagter Information, ob das digitale Objekt verfallen ist, den Teil für das digitale Objekt des virtuellen Behälters, falls es verfallen ist, mit aussagelosen Zeichen überschreibt, und das verschlüsselte digitale Objekt aus dem Teil für das digitale Objekt liest und eine Entschlüsselungstechnik auf das digitale Objekt anwendet, wenn das digitale Objekt nicht verfallen ist.

33. System nach Anspruch 32, bei dem der virtuelle Behälter über das Internet gesendet wird.

34. System nach Anspruch 32, bei dem das digitale Objekt ein Dokument ist.

35. System nach Anspruch 32, bei dem das digitale Objekt ein Programm ist.

36. System nach Anspruch 31, bei dem der Teil für das digitale Objekt des virtuellen Behälters mit aussagelosen Zeichen überschrieben wird, wenn die Lebenszeitkontrolle ungültig ist.

37. System nach Anspruch 31, bei dem der virtuelle Behälter in zusammenhängenden Speicherplätzen in einem elektronischen Speicher gespeichert ist.

38. Virtuelles Behältersystem nach Anspruch 1, bei dem die Lebenszeitkontrolle ein Kopieren des digitalen Objekts verhindert.

39. Virtuelles Behältersystem nach Anspruch 1, bei dem die Lebenszeitkontrolle ein Weiterleiten des digitalen Objekts verhindert.

40. System nach Anspruch 18, bei dem die Lebenszeitkontrolle ein Kopieren des digitalen Objekts verhindert.

41. System nach Anspruch 18, bei dem die Lebenszeitkontrolle ein Weiterleiten des digitalen Objekts verhindert.

42. System nach Anspruch 31, bei dem die Lebenszeitkontrolle ein Kopieren des digitalen Objekts verhindert.

43. System nach Anspruch 31, bei dem die Lebenszeitkontrolle ein Weiterleiten des digitalen Objekts verhindert.

44. System nach Anspruch 32, bei dem die Lebenszeitkontrolle ein Kopieren des digitalen Objekts verhindert.

45. System nach Anspruch 32, bei dem die Lebenszeitkontrolle ein Weiterleiten des digitalen Objekts verhindert.

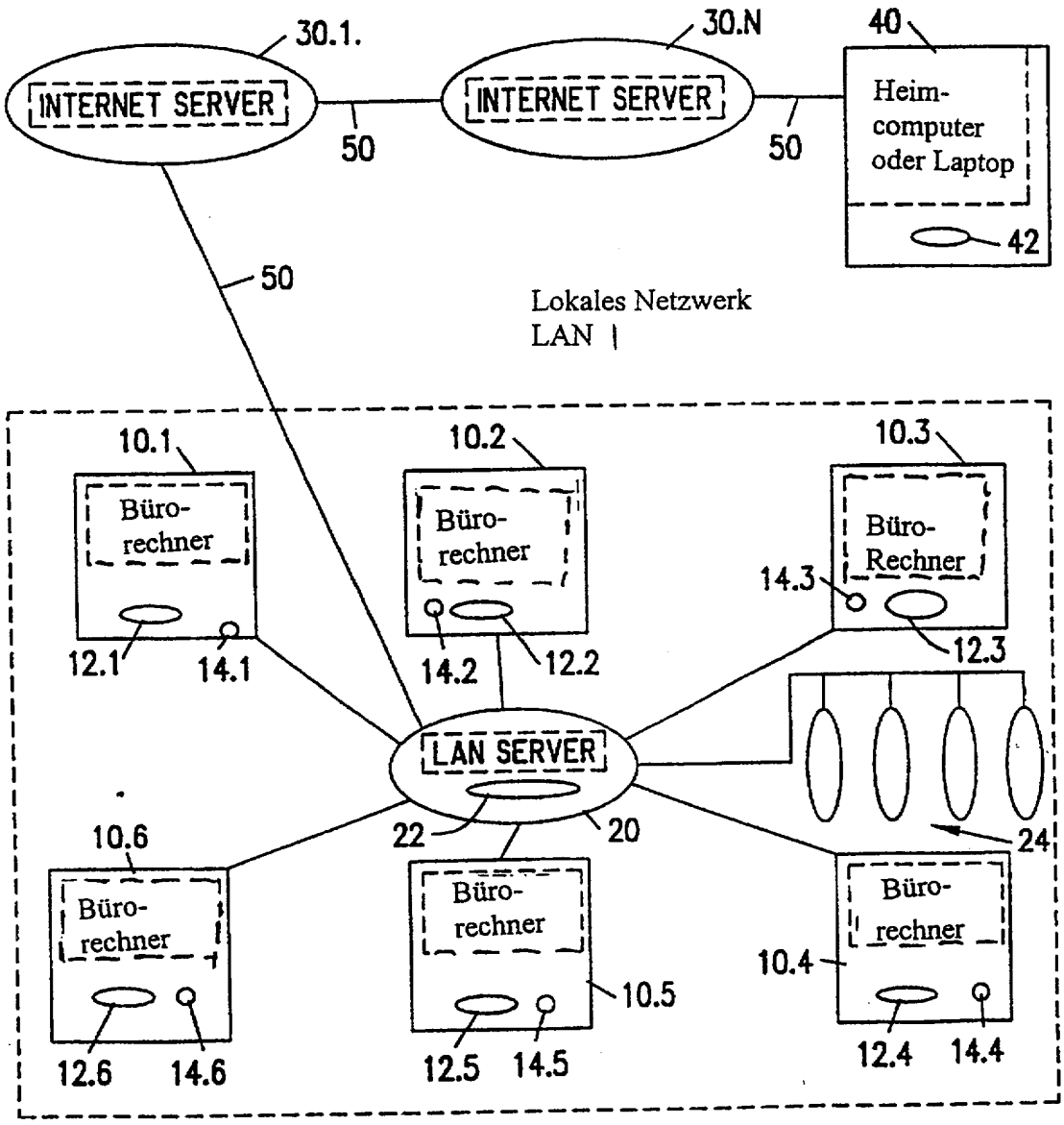

30.1.
INTERNET SERVER
50
30.N
INTERNET SERVER
50
40
Heim-computer oder Laptop
42
50
Lokales Netzwerk
LAN
10.1
Büro-rechner
12.1
14.1
10.2
Büro-rechner
14.2
12.2
10.3
Büro-Rechner
14.3
12.3
LAN SERVER
22
20
24
10.6
Büro-rechner
12.6
14.6
Büro-rechner
10.5
12.5
14.5
Büro-rechner
10.4
12.4
14.4

FIG. 1

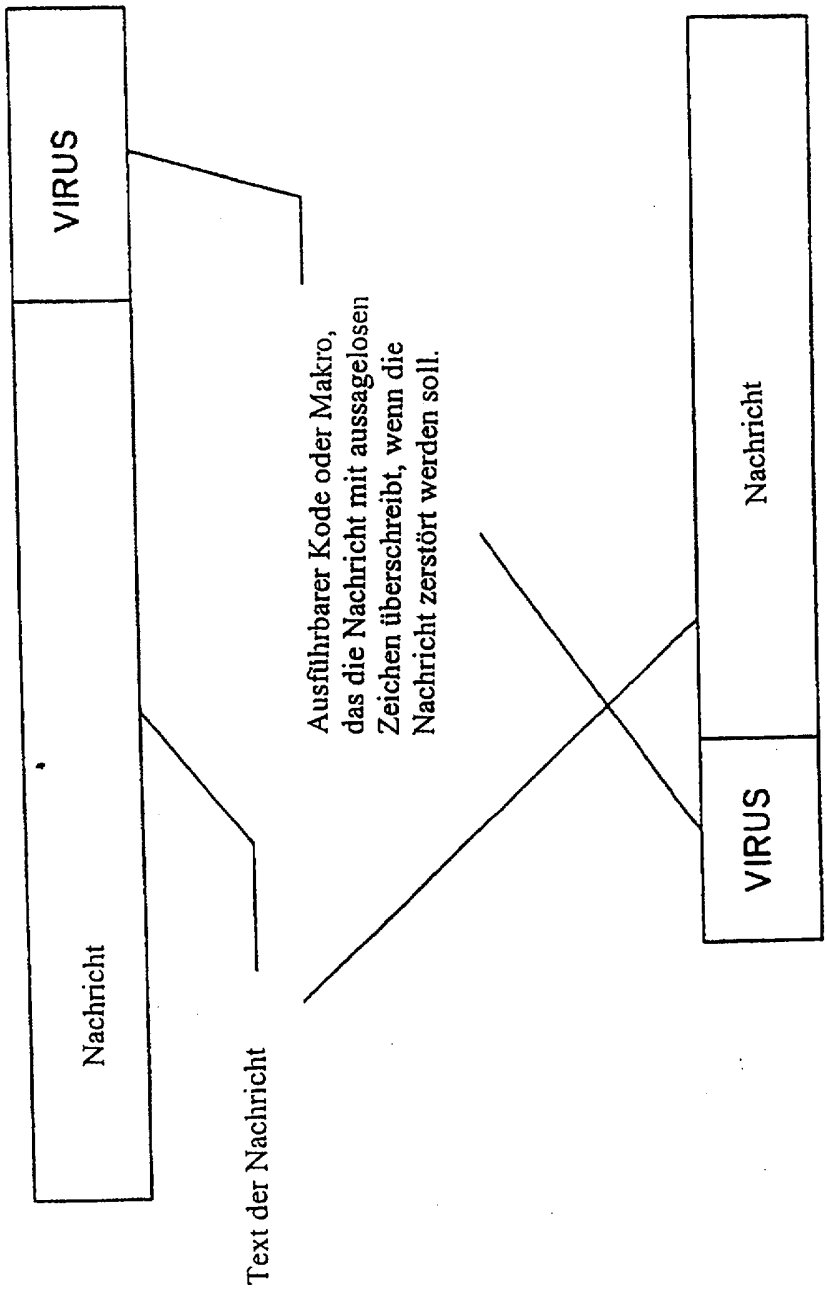

VIRUS
Nachricht
Text der Nachricht
Ausführbarer Kode oder Makro, das die Nachricht mit aussagelosen Zeichen überschreibt, wenn die Nachricht zerstört werden soll.
Nachricht
VIRUS

FIG. 2

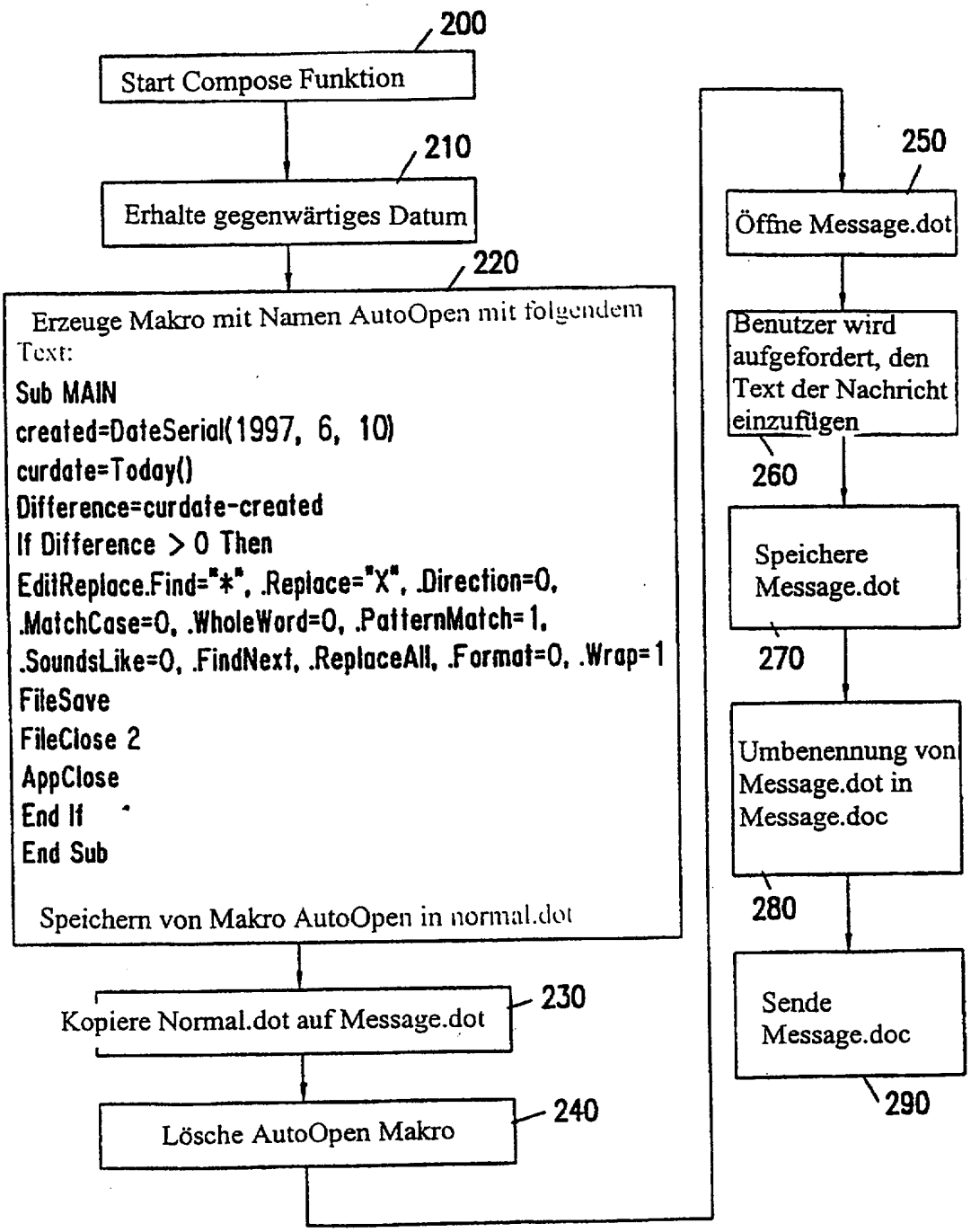

200
Start Compose Funktion
210
Erhalte gegenwärtiges Datum
220
Erzeuge Makro mit Namen AutoOpen mit folgendem Text:
Sub MAIN
created=DateSerial(1997, 6, 10)
curdate=Today()
Difference=curdate-created
If Difference > 0 Then
EditReplace.Find="*", .Replace="X", .Direction=0,
.MatchCase=0, .WholeWord=0, .PatternMatch=1,
.SoundsLike=0, .FindNext, .ReplaceAll, .Format=0, .Wrap=1
FileSave
FileClose 2
AppClose
End If
End Sub
Speichern von Makro AutoOpen in normal.dot
230
Kopiere Normal.dot auf Message.dot
240
Lösche AutoOpen Makro
250
Öffne Message.dot
Benutzer wird aufgefordert, den Text der Nachricht einzufügen
260
Speichere Message.dot
270
Umbenennung von Message.dot in Message.doc
280
Sende Message.doc
290

FIG. 3

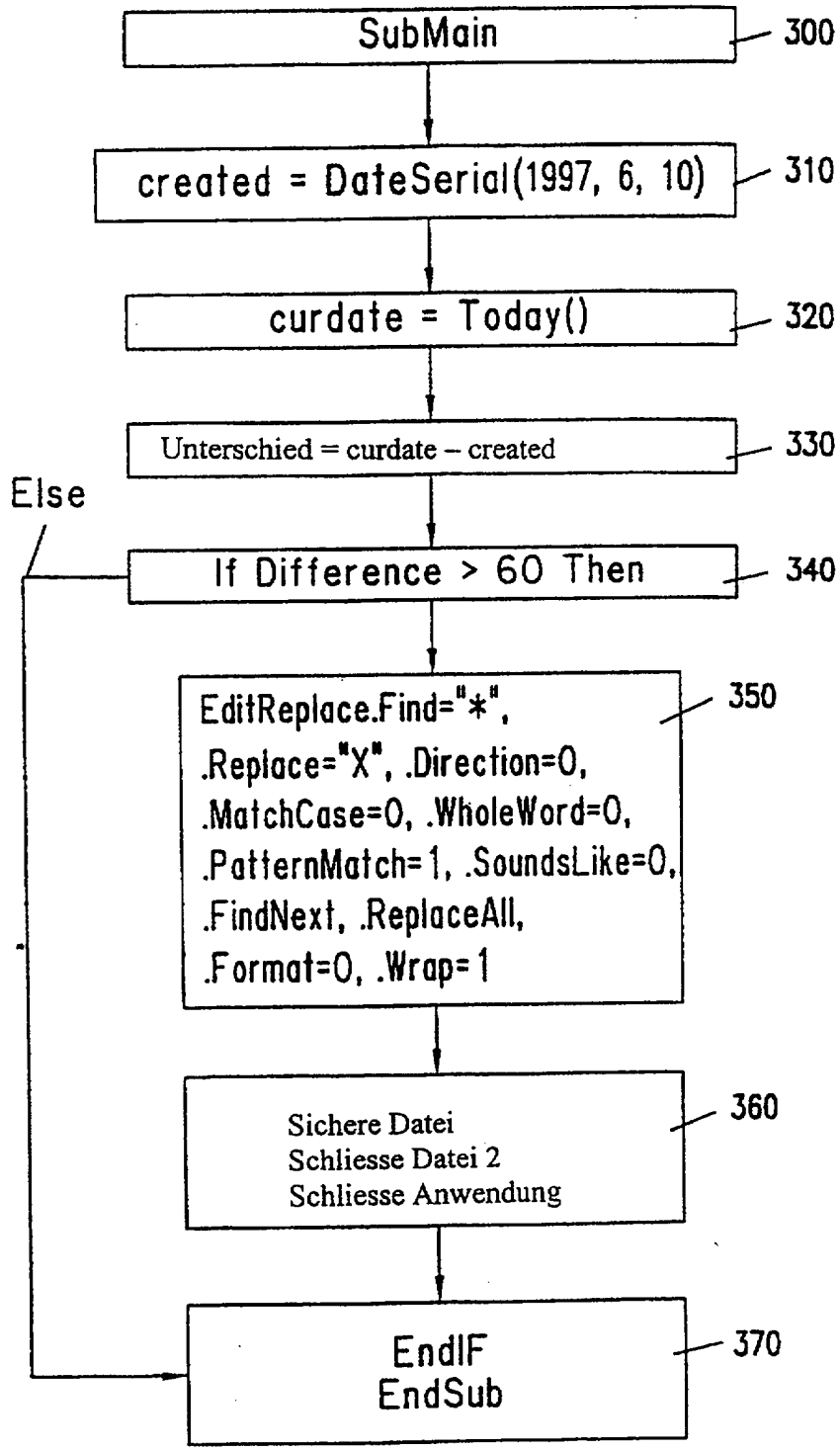
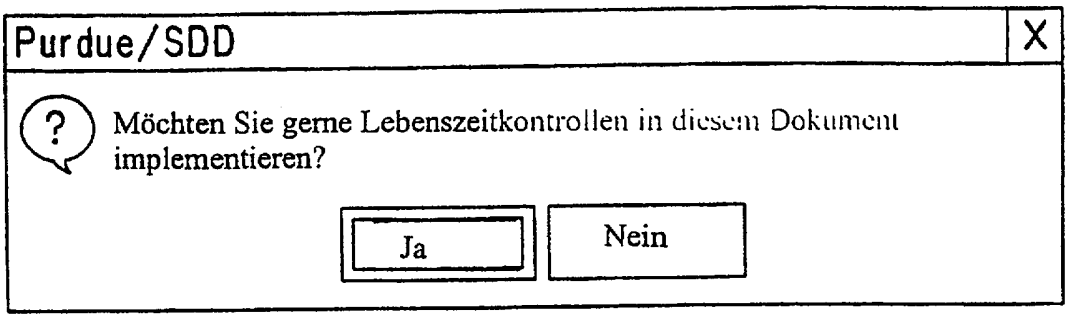
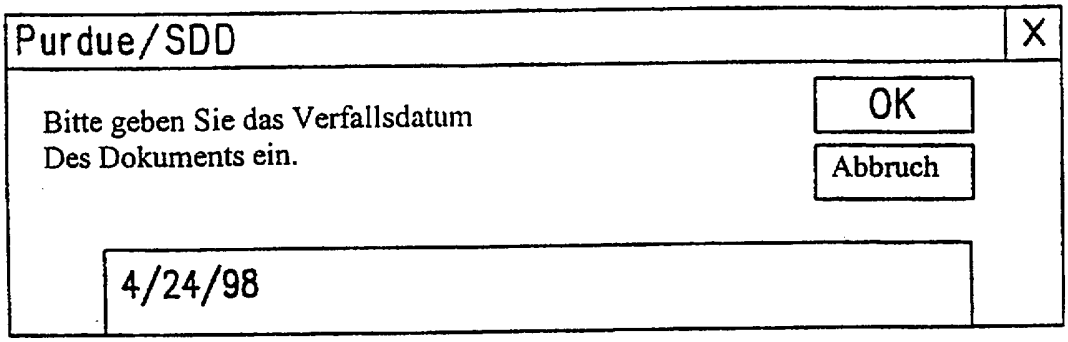

SubMain
300
created = DateSerial(1997, 6, 10)
310
curdate = Today()
320
Unterschied = curdate – created
330
Else
If Difference > 60 Then
340
EditReplace.Find="*",
.Replace="X", .Direction=0,
.MatchCase=0, .WholeWord=0,
.PatternMatch=1, .SoundsLike=0,
.FindNext, .ReplaceAll,
.Format=0, .Wrap=1
350
Sichere Datei
Schliesse Datei 2
Schliesse Anwendung
360
EndIF
EndSub
370

FIG. 4



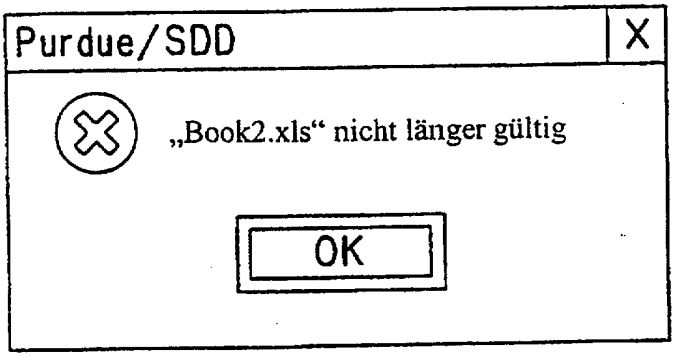
Fordere Benutzer auf, aus dem Dokument ein SDD zu machen

FIG. 5A



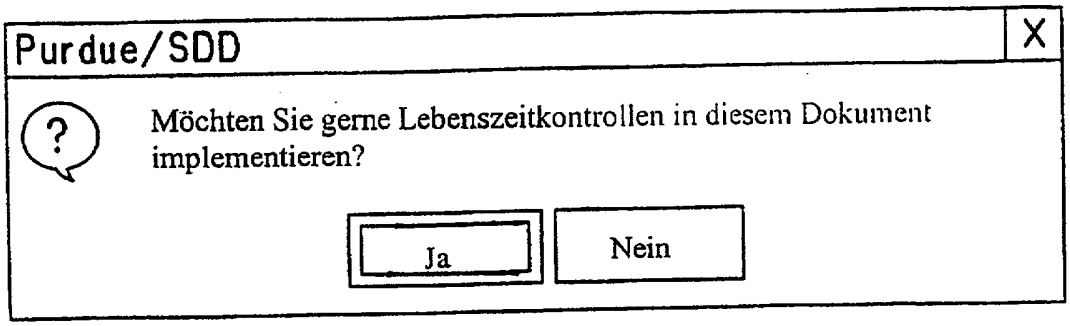
Fordere Benutzer auf, die Zerstörungsparameter einzugeben

FIG. 5B



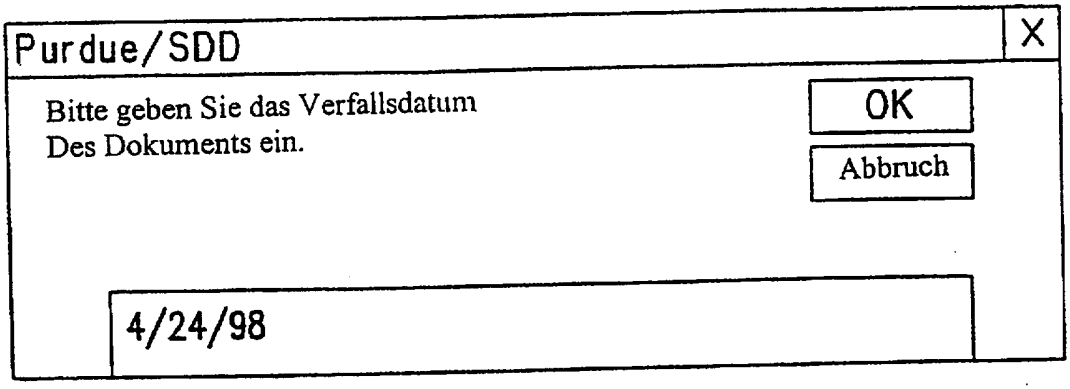
Weise auf folgende Zerstörung hin

FIG. 5C



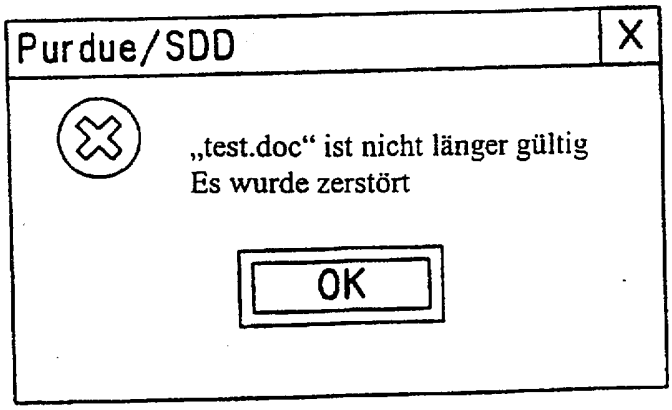
Fordere Benutzer auf, aus dem Dokument ein SDD zu machen

FIG. 6A



Fordere Benutzer auf, die Zerstörungsparameter einzugeben

FIG. 6B



Hinweis auf erfolgte Zerstörung

FIG. 6C

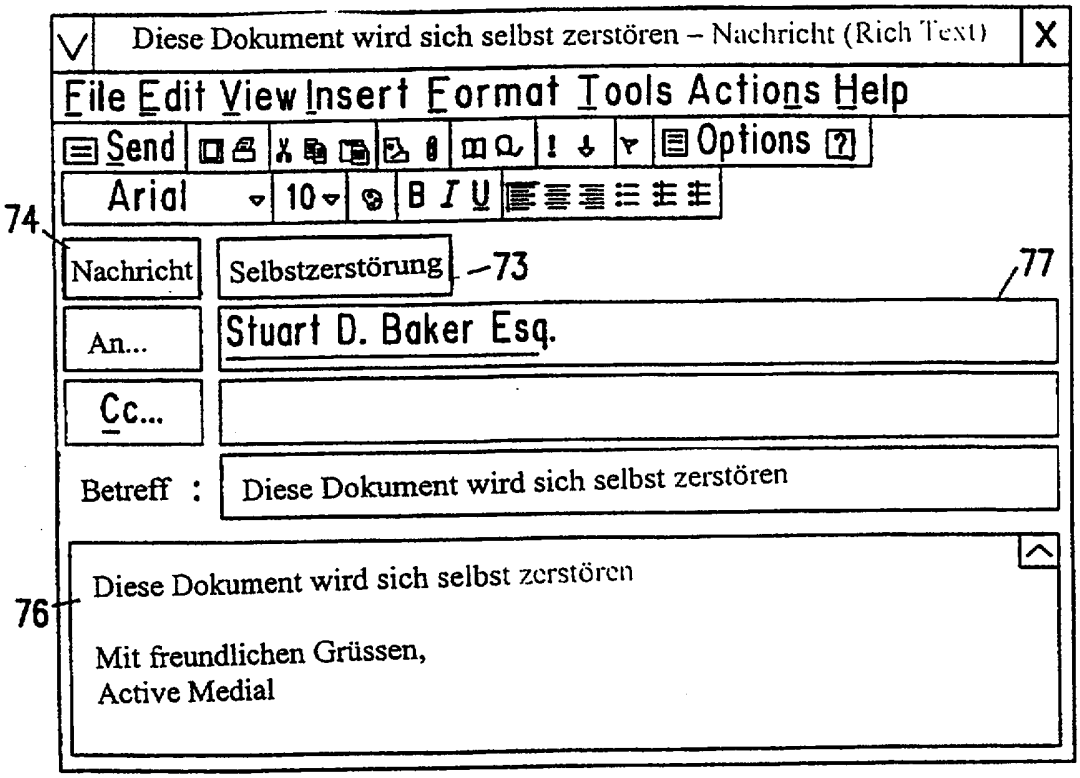

Diese Dokument wird sich selbst zerstören – Nachricht (Rich Text)
File Edit View Insert Format Tools Actions Help
Send
Options
Arial
10
B I U
74
Nachricht
Selbstzerstörung
73
77
An...
Stuart D. Baker Esq.
Cc...
Betreff :
Diese Dokument wird sich selbst zerstören
76
Diese Dokument wird sich selbst zerstören
Mit freundlichen Grüssen,
Active Medial

FIG. 7A

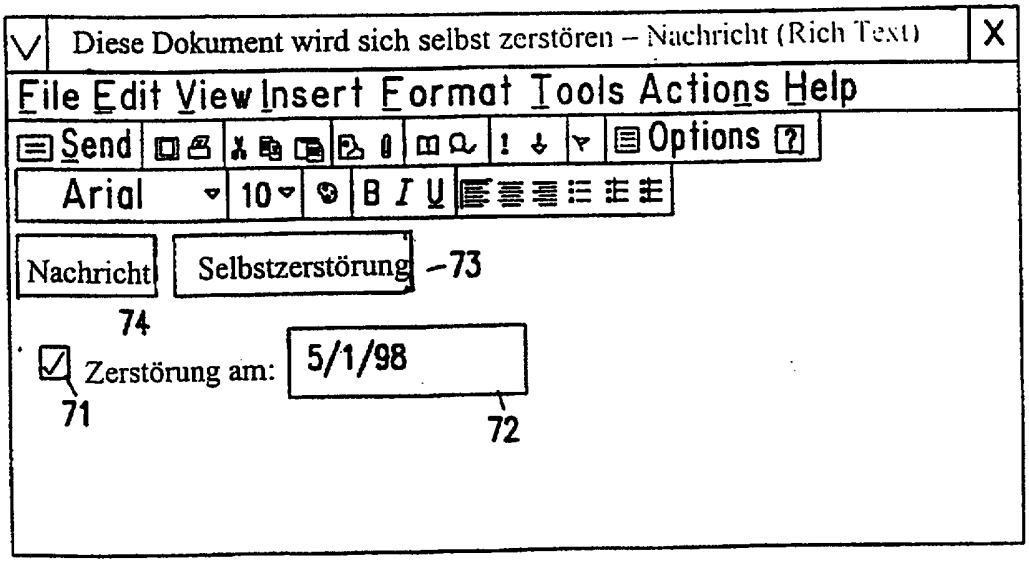
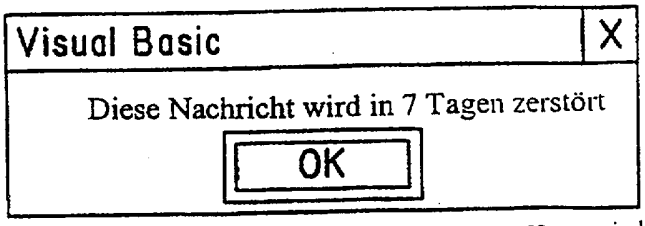
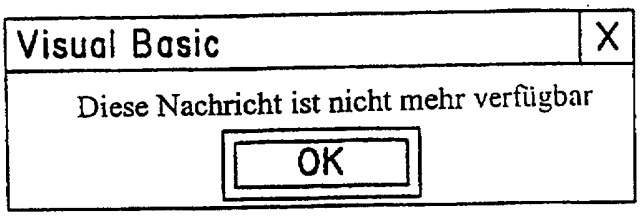

Diese Dokument wird sich selbst zerstören – Nachricht (Rich Text)
File Edit View Insert Format Tools Actions Help
Send
Options
Arial
10
B I U
Nachricht
Selbstzerstörung
73
74
Zerstörung am:
5/1/98
71
72

FIG. 7B



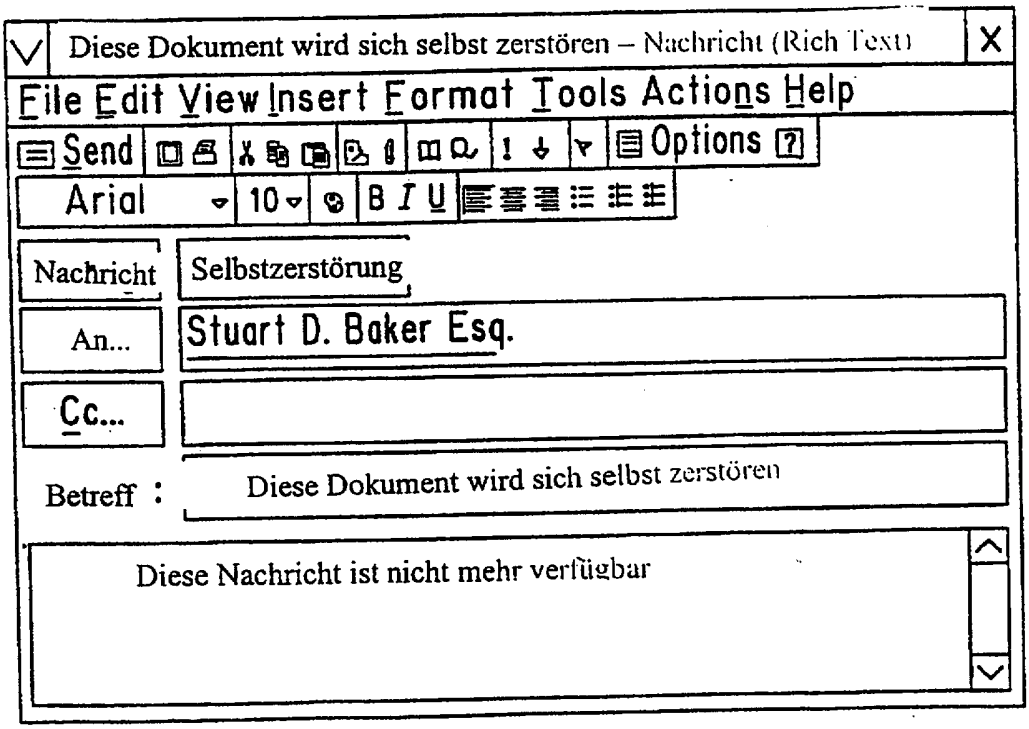
Warnung, wenn Nachricht geöffnet wird

FIG. 7C



Nachricht nach Zerstörung der Nachricht

FIG. 7D



Bestandteile der Nachricht nach ihrer Zerstörung

FIG. 7E

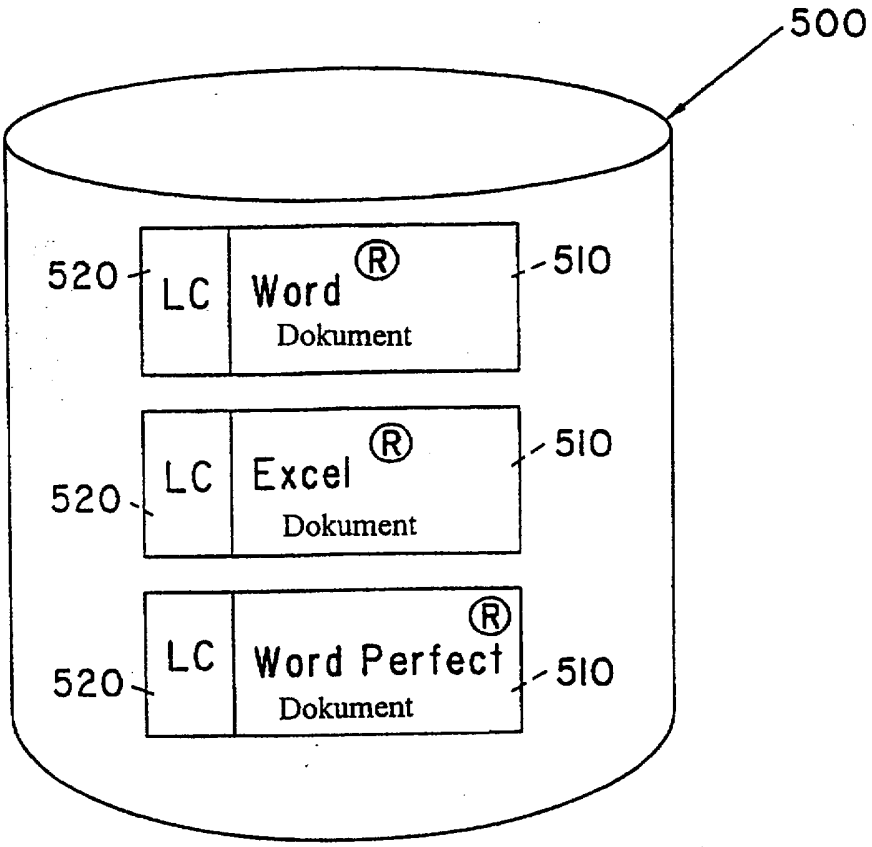

500
520
LC
Word ®
Dokument
510
520
LC
Excel ®
Dokument
510
520
LC
Word Perfect ®
Dokument
510

FIG. 8A

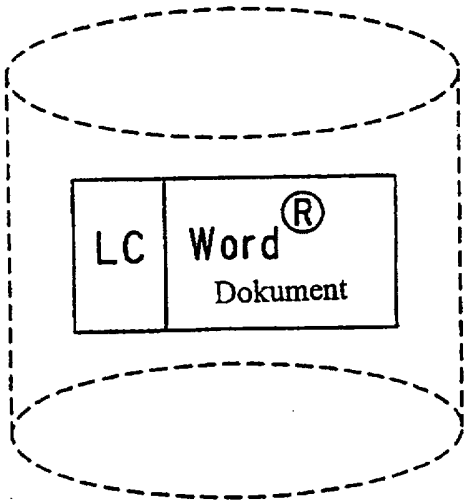

LC
Word ®
Dokument

FIG. 8B

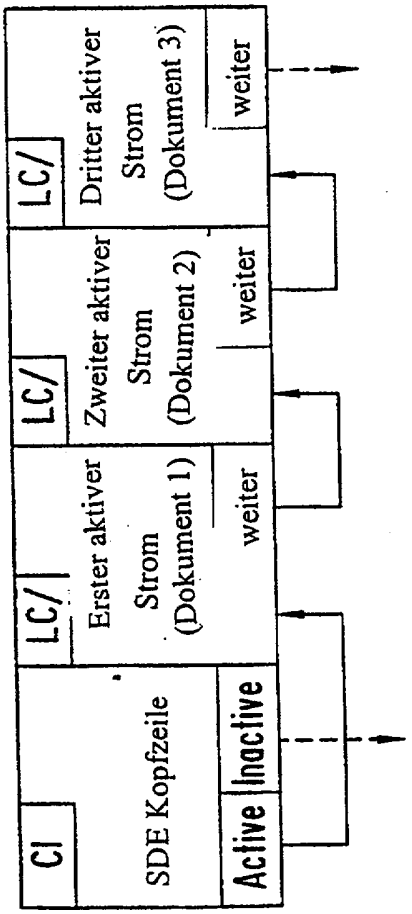

CI
SDE Kopfzeile
Active
Inactive
LC/
Erster aktiver Strom (Dokument 1)
weiter
LC/
Zweiter aktiver Strom (Dokument 2)
weiter
LC/
Dritter aktiver Strom (Dokument 3)
weiter

FIG. 9A

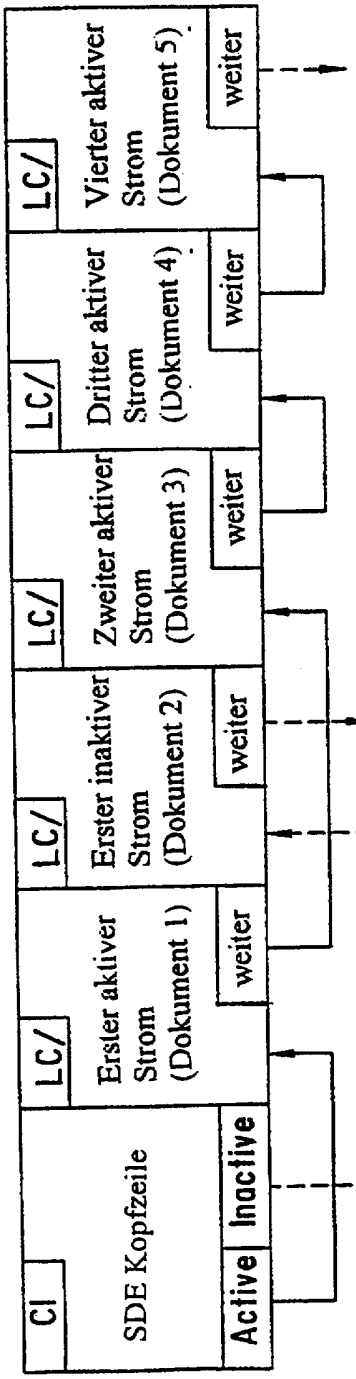

CI
SDE Kopfzeile
Active
Inactive
LC/
Erster aktiver Strom (Dokument 1)
weiter
LC/
Erster inaktiver Strom (Dokument 2)
weiter
LC/
Zweiter aktiver Strom (Dokument 3)
weiter
LC/
Dritter aktiver Strom (Dokument 4)
weiter
LC/
Vierter aktiver Strom (Dokument 5)
weiter

FIG. 9B

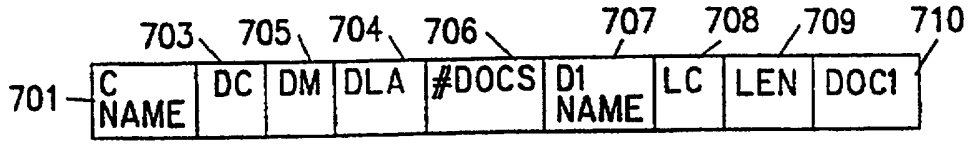


FIG. 10A

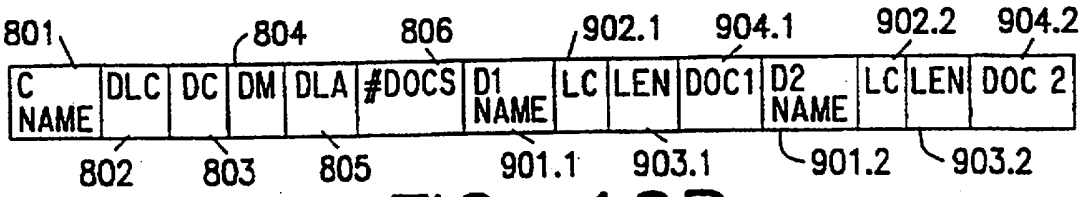


FIG. 10B

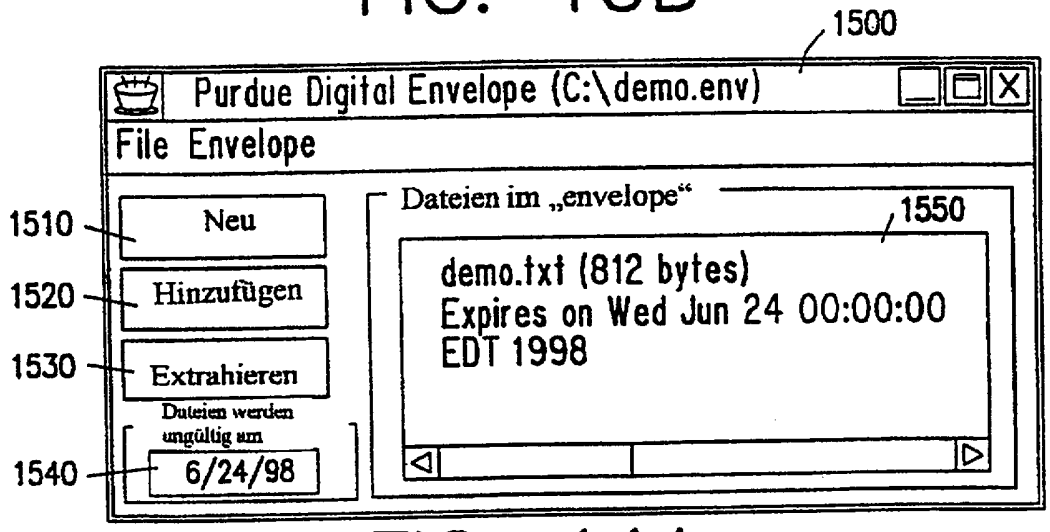


FIG. 11A

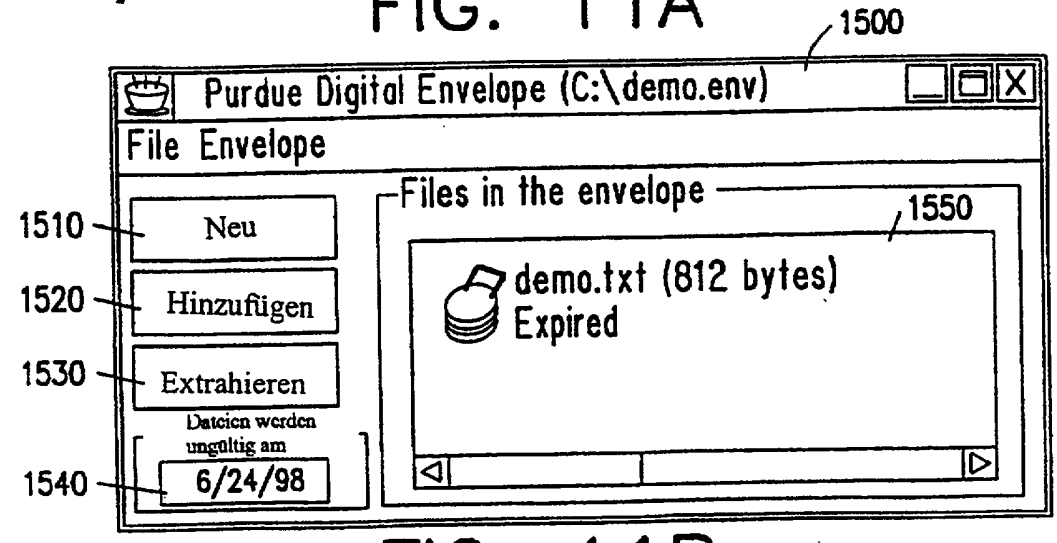


FIG. 11B

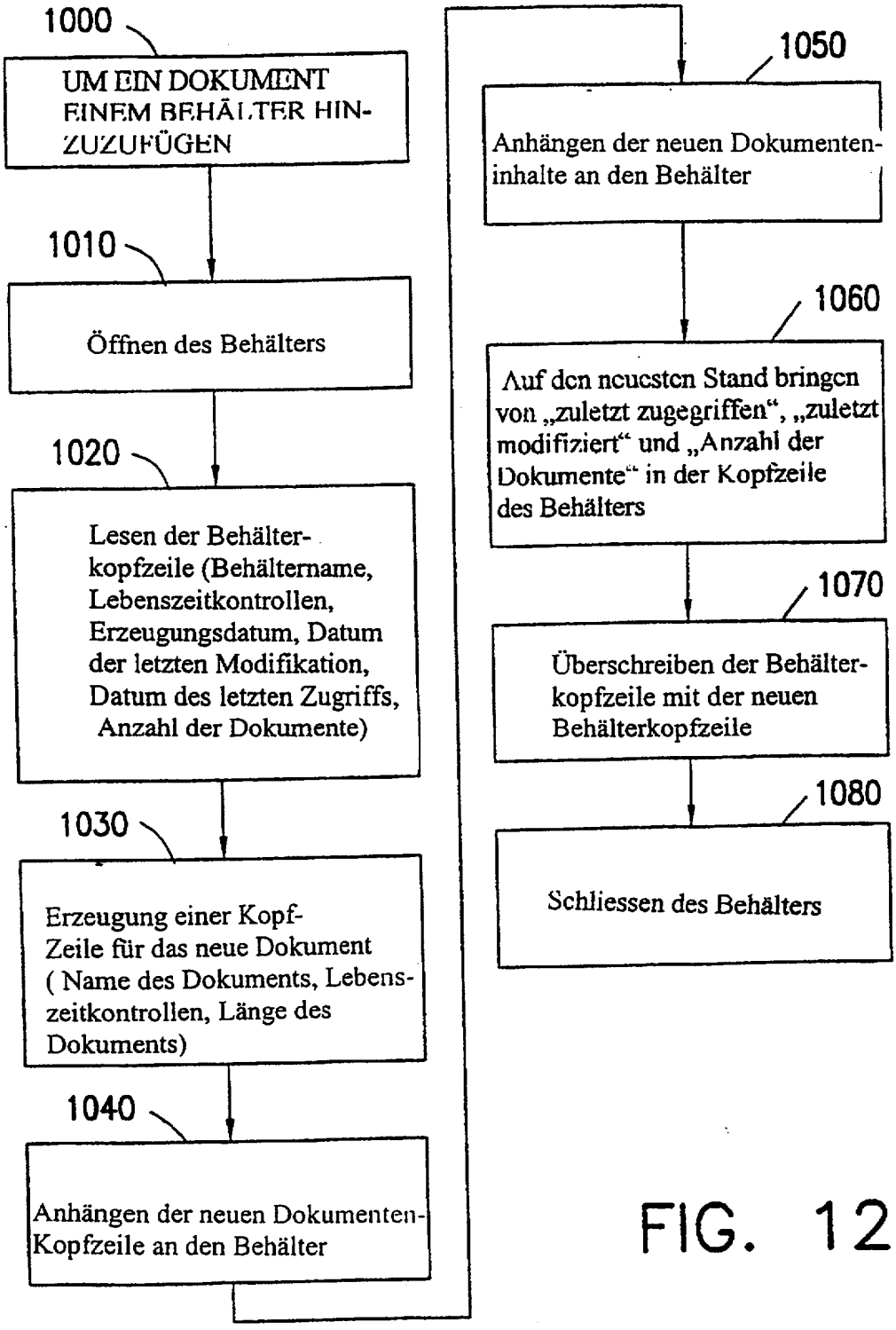

1000
UM EIN DOKUMENT EINEM BEHÄLTER HINZUZUFÜGEN
1010
Öffnen des Behälters
1020
Lesen der Behälterkopfzeile (Behältername, Lebenszeitkontrollen, Erzeugungsdatum, Datum der letzten Modifikation, Datum des letzten Zugriffs, Anzahl der Dokumente)
1030
Erzeugung einer Kopf-Zeile für das neue Dokument (Name des Dokuments, Lebenszeitkontrollen, Länge des Dokuments)
1040
Anhängen der neuen Dokumenten-Kopfzeile an den Behälter
1050
Anhängen der neuen Dokumenteninhalte an den Behälter
1060
Auf den neuesten Stand bringen von „zuletzt zugegriffen", „zuletzt modifiziert" und „Anzahl der Dokumente" in der Kopfzeile des Behälters
1070
Überschreiben der Behälterkopfzeile mit der neuen Behälterkopfzeile
1080
Schliessen des Behälters

FIG. 12

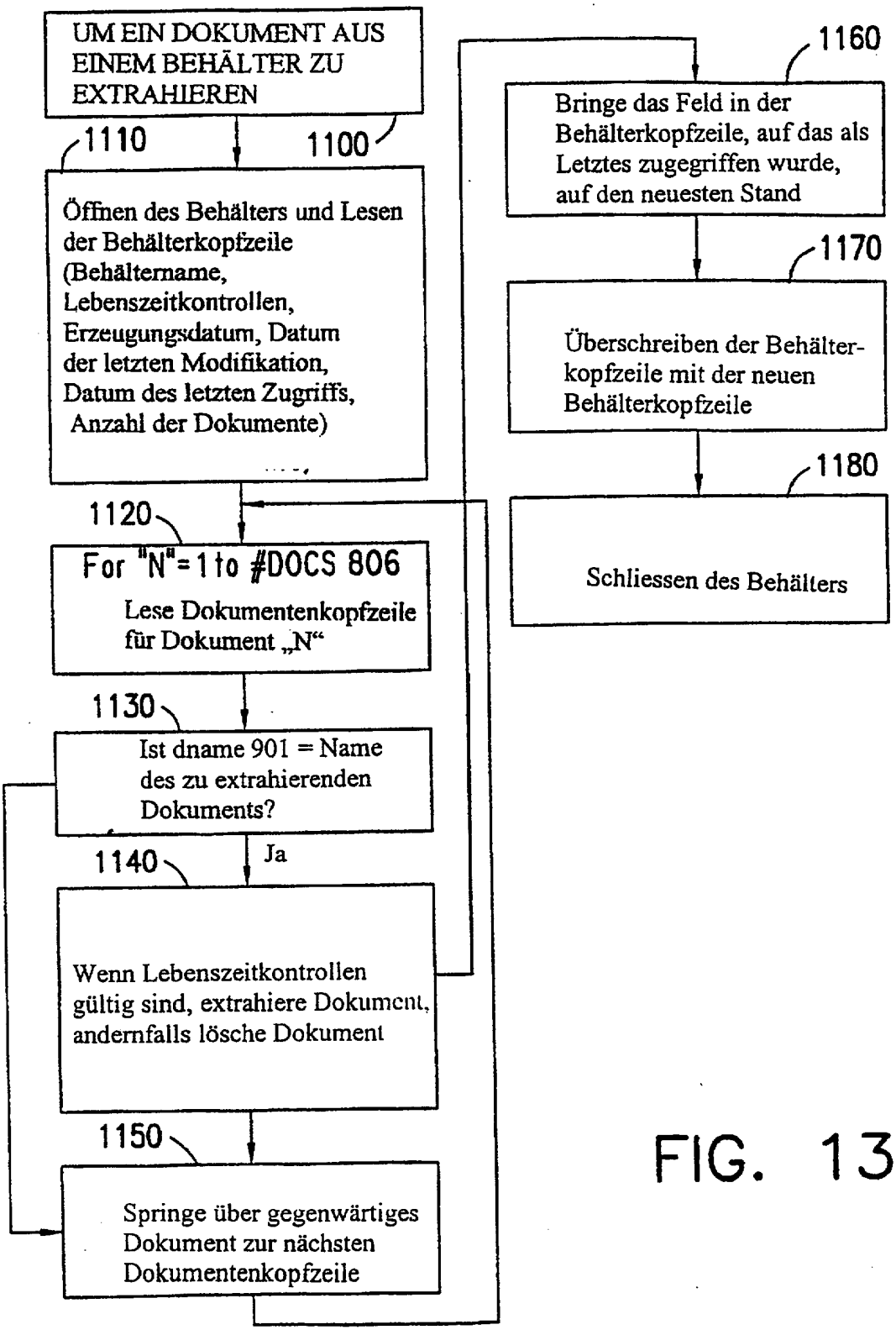

UM EIN DOKUMENT AUS EINEM BEHÄLTER ZU EXTRAHIEREN
1100
1110
Öffnen des Behälters und Lesen der Behälterkopfzeile (Behältername, Lebenszeitkontrollen, Erzeugungsdatum, Datum der letzten Modifikation, Datum des letzten Zugriffs, Anzahl der Dokumente)
1120
For "N" = 1 to #DOCS 806
Lese Dokumentenkopfzeile für Dokument „N"
1130
Ist dname 901 = Name des zu extrahierenden Dokuments?
Ja
1140
Wenn Lebenszeitkontrollen gültig sind, extrahiere Dokument, andernfalls lösche Dokument
1150
Springe über gegenwärtiges Dokument zur nächsten Dokumentenkopfzeile
1160
Bringe das Feld in der Behälterkopfzeile, auf das als Letztes zugegriffen wurde, auf den neuesten Stand
1170
Überschreiben der Behälterkopfzeile mit der neuen Behälterkopfzeile
1180
Schliessen des Behälters

FIG. 13