(19) Deutsches Patent- und Markenamt

(10) **DE 11 2018 001 559 B4** 2023.09.07

(12) # Patentschrift

(21) Deutsches Aktenzeichen: **11 2018 001 559.9**
(86) PCT-Aktenzeichen: **PCT/IB2018/053877**
(87) PCT-Veröffentlichungs-Nr.: **WO 2018/220570**
(86) PCT-Anmeldetag: **31.05.2018**
(87) PCT-Veröffentlichungstag: **06.12.2018**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **05.12.2019**
(45) Veröffentlichungstag der Patenterteilung: **07.09.2023**

(51) Int Cl.: ***H04L 65/00*** (2022.01)
H04L 9/40 (2022.01)
H04L 69/12 (2022.01)
H04L 67/01 (2022.01)
H04L 9/08 (2006.01)
H04L 9/30 (2006.01)
H04L 67/146 (2022.01)
H04L 67/56 (2022.01)

(30) Unionspriorität:
15/611,229 **01.06.2017** **US**

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

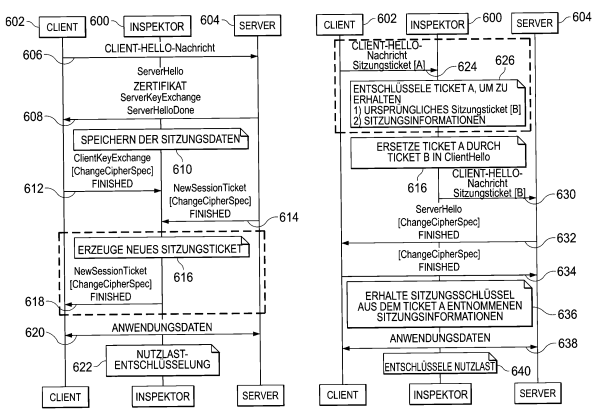
(74) Vertreter:
Richardt Patentanwälte PartG mbB, 65185 Wiesbaden, DE

(72) Erfinder:
Lee, Cheng-ta, Taipei, CN; Hsiung, Wei-Hsiang, Taipei, CN; Suen, Wei-Shiau, Taipei, CN; Wu, Ming-Hsun, Taipei, CN

(56) Ermittelter Stand der Technik:
US **2016 / 0 315 913** **A1**

(54) Bezeichnung: **CACHESPEICHERLOSE SITZUNGSTICKET-UNTERSTÜTZUNG BEI TLS-PRÜFUNG**

(57) Hauptanspruch: Verfahren, das innerhalb eines zwischen einem Transport Layer Security-(TLS-)Client (602) und einem TLS-Server (604) angeordneten Vermittlers (600) wirksam ist und das eine TLS-Prüffunktion während einer TLS-Sitzung bereitstellt, wobei das Verfahren aufweist:
nach dem Empfang (614) eines ursprünglichen Sitzungstickets von dem TLS-Server (604) Erzeugen (616) eines neuen Sitzungstickets, wobei das neue Sitzungsticket ein Wert ist, der abgeleitet wird, indem eine Umsetzungsfunktion auf das ursprüngliche Sitzungsticket und Sitzungskontextinformationen angewendet wird, wobei die Sitzungskontextinformationen eine Cipher-Suite und einen Masterschlüssel beinhalten, die zur Verwendung während der TLS-Sitzung ausgehandelt werden;
Anwenden einer Verschlüsselungsfunktion auf das neue Sitzungsticket;
Ausgeben (618) des verschlüsselten neuen Sitzungstickets an den TLS-Client (602), wodurch die Notwendigkeit, die Sitzungskontextinformationen zu verwalten, von dem Vermittler (600) auf den TLS-Client (602) übertragen wird;
nach dem Empfang (624) des verschlüsselten neuen Sitzungstickets von dem TLS-Client (602) Entschlüsseln (626) des verschlüsselten neuen Sitzungstickets, um das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen einschließlich des Masterschlüssels wiederherzustellen; und
Verwenden des wiederhergestellten ursprünglichen Sitzungstickets und der Sitzungskontextinformationen, um die TLS-Sitzung wieder aufzunehmen.



Beschreibung

HINTERGRUND DER ERFINDUNG

Technisches Gebiet

[0001] Diese Offenbarung betrifft allgemein die Informationssicherheit auf Appliances, die mit einem Netzwerk verbunden sind.

Beschreibung der verwandten Technik

[0002] Sicherheitsbedrohungen nehmen ständig weiter zu. Mit der schnellen Zunahme von innovativen Web-Anwendungen und der zunehmenden gemeinsamen Dateinutzung könnten Aktivitäten, die in der Vergangenheit möglicherweise als harmlos erachtet wurden, potenzielle Einfallstore für Angreifer werden. Herkömmliche Sicherheitsmittel wie beispielsweise Anti-Malware-Software und Firewalls sind leichter zu umgehen geworden. Somit besteht ein erheblicher Bedarf an einem erweiterten proaktiven Bedrohungsschutz, der dazu beitragen kann, umfangreiche Sicherheit gegen neue und entstehende Bedrohungen zu leisten.

[0003] Mit dem Netzwerk verbundene Einheiten ohne Bildschirm („Appliances“) sind in vielen Datenverarbeitungsumgebungen weit verbreitet. Zum Beispiel sind Appliances, die vorsätzlich zur Durchführung von herkömmlichen Funktionen einer serviceorientierten Architektur (SOA) für Middleware gebaut wurden, über bestimmte Computerumgebungen hinweg häufig anzutreffen. SOA-Middleware-Appliances können Bereitstellungen von XML- und Web-Services vereinfachen, zu sichern helfen oder beschleunigen, während sie eine vorhandene SOA-Infrastruktur über ein Unternehmen hinweg erweitern. Mit der Nutzung von für Middleware bestimmter Hardware und eines einfachen Middleware-Stapelspeichers kann die von herkömmlichen Software-Lösungen erfahrene Leistungseinbuße angegangen werden. Ferner sieht der Appliance-Formfaktor eine sichere, verbraucherfreundliche Gehäuseform für die Ausführung von Middleware-SOA-Funktionen vor. Ein besonderer Vorteil, den diese Arten von Einheiten bieten, besteht darin, Backend-Systeme von Verarbeitungsaufgaben zu entlasten. Zu diesem Zweck ist es hinlänglich bekannt, solche Middleware-Einheiten zur Durchführung von rechnerisch aufwendigen Prozessen in Bezug auf die Netzwerksicherheit zu verwenden. Zum Beispiel sind Netzwerk-Intrusion-Prevention-System- (IPS-)Appliances so ausgelegt, dass sie an den Eintrittspunkten in ein Unternehmensnetzwerk sitzen, um geschäftskritische Betriebsmittel wie beispielsweise interne Netzwerke, Server, Endpunkte und Anwendungen vor böswilligen Bedrohungen zu schützen.

[0004] Die Verwendung von Secure Sockets Layer-(SSL)- und/oder Transport Layer Security-(TLS)-basierter Verschlüsselung für Netzübertragungen behindert im Allgemeinen die Fähigkeit, Bedrohungsverkehr aus dem Netzinneren heraus zu erkennen und zu entschärfen. Es wird nunmehr geschätzt, dass über zwei Drittel oder mehr des gesamten Unternehmensnetzverkehrs über SSL/TLS übertragen wird. Das bedeutet, dass Organisationen, die auf Netzübertragungen angewiesen sind, üblicherweise nicht in der Lage sind, die Endpunkte in ihrem Unternehmen, die gegebenenfalls für solche Bedrohungen anfällig sind, (aus dem Netzwerk) zu schützen. Tatsächlich verwendet der überwiegende Teil der SSL/TLS-Übertragungen nur Serverauthentifizierung, d.h., der Server wird über die SSL/TLS-Protokolle dem Client gegenüber authentifiziert, jedoch wird der Client in Bezug auf den Server nicht authentifiziert. Diese Authentifizierungsasymmetrie bietet einem Prozess die Möglichkeit, sich selbst so zwischen Client und Server zu schalten, dass eine Entschlüsselung von Übertragungen und eine Prüfung von deren Inhalten ermöglicht wird. Ein solcher „Man-in-the-Middle“-(MITM-)Prozess kann böswillig sein oder er kann aus legitimen Gründen, wie beispielsweise zur Paketprüfung (zur Erkennung von Bedrohungen), verwendet werden.

[0005] Es ist somit bekannt, einen Transport-(MITM-)Proxy zwischen einem Client und einem Server bereitzustellen, der so konfiguriert sein kann, dass er zwei getrennte SSL/TLS-Sitzungen aufbaut und verwaltet, eine als der Client zum Zielserver und eine weitere als ein Server zu dem einleitenden Client. Der Zwischenproxy erscheint dem Server somit als ein Client und dem Client als der vorgesehene Server. Von dem Client aus eingeleitete Übertragungen und etwaige von dem Server empfangene Antworten stehen dann theoretisch zur Prüfung und anschließenden Aktion durch einen SSL/TLS-Inspektor zur Verfügung.

[0006] Bei der Durchführung einer Man-in-the-Middle-Prüfung einer SSL/TLS-Verbindung ist die Unterstützung durch den SSL/TLS-Inspektor einer TLS-Sitzungswiederaufnahme eine wichtige Voraussetzung. Herkömmlicherweise und wie hinlänglich bekannt ist, ermöglichen TLS-Server Clients eine Sitzungswiederaufnahme zu Clients (ungeachtet des MITM) gemäß einem von zwei (2) bestimmten Verfahren, nämlich Session ID (RFC 4507) und Session Ticket (RFC 5077). Session ID war der erste Mechanismus, der erfunden wurde, um den SSL/TLS-Handshake zu beschleunigen und eine Sitzungswiederaufnahme zu ermöglichen. Bei Session ID werden alle Sitzungsinformationen serverseitig gespeichert. Eine Session ID-Sitzungswiederaufnahme hat mehrere Nachteile, wobei der bedeutendste ein Leistungsengpass aufgrund dessen ist, dass für das Nachschlagen einer bestimmten Sitzungs-ID im

Falle einer großen Anzahl von zwischengespeicherten Sitzungen Zeit und Platz aufgewendet werden muss. Ein weiterer wesentlicher Nachteil von Session ID ist, dass eine Sitzungs-ID nur auf einem Server funktionieren kann, was einen Einsatz sehr schwer skalierbar macht, sofern Sitzungscachespeicher nicht über die Server hinweg synchronisiert werden. Um Unzulänglichkeiten von Session ID anzugehen, wurde Session Ticket entwickelt. Session Ticket ist ein Mechanismus, der es dem TLS-Server ermöglicht, Sitzungen wieder aufzunehmen und das Festhalten eines clientweisen Sitzungszustands zu vermeiden. Zu diesem Zweck kapselt der TLS-Server den Sitzungszustand in ein Ticket ein und leitet es an den Client weiter. Das Ticket wird von dem Serviceanbieter signiert. Der Client kann daraufhin eine Sitzung unter Verwendung des erhaltenen Tickets wieder aufnehmen. Wenn der Server später ein Ticket empfängt und feststellt, dass es von dem Serviceanbieter signiert wurde, berücksichtigt er jede in dem Ticket gespeicherte Einstellung. Session Ticket wird unter Webservern aufgrund seiner Skalierbarkeit und eines geringeren serverseitigen Ressourcenaufwands in großem Umfang eingesetzt.

[0007] Um Session Ticket bei einer TLS-Prüfung zu unterstützen (z.B. durch einen transparenten MITM-Proxy), ist eine Zwischenspeicherung notwendig, um die Zuordnung zwischen Ticket und Sitzungsschlüssel zu verwalten. Der Grund hierfür ist, dass es keine praktische Möglichkeit gibt, das Sitzungsticket zu entschlüsseln, da der Verschlüsselungsmechanismus durch die Anwendung/den Service gesteuert wird und nicht von der SSL/TLS-Spezifikation definiert ist. Das Verwalten des Sitzungscachespeichers in dem Inspektor hat jedoch mehrere Nachteile, darunter eine schlechte Skalierbarkeit aufgrund der Schwierigkeit, den Sitzungscachespeicher zu verteilen, Begrenzungen bei der Größe des Sitzungscachespeichers wegen Massenspeicher- und Hauptspeichereinschränkungen, die dazu führen können, dass dem Proxy Cachespeicher ausgeht, CPU-Begrenzungen, die die Nachschlagezeit im Cachespeicher erschweren und die Art der Hash-Algorithmen begrenzen, die für eine Zuordnung der Sitzungstickets ausgeführt werden können, sowie eine Verwundbarkeit gegen eine Denial-of-Service-Attacke durch einen Angreifer, der Einträge in dem Sitzungscachespeicher absichtlich löschen und dadurch eine Prüfung umgehen könnte.

[0008] Es bleibt, eine TLS-Sitzungswiederaufnahme in einem TLS-Inspektor, der Session-Ticket-Unterstützung bereitstellt, zu ermöglichen, welche aber diese und damit verbundene Unzulänglichkeiten in früheren Methoden überwindet.

[0009] Folglich besteht in der Technik Bedarf, das vorstehend genannte Problem anzugehen.

[0010] Die US 2016 / 0 315 913 A1 offenbart ein zwischengeschaltetes Netzwerkgerät, das eine Anfrage für eine sichere Kommunikationssitzung zwischen einem Endpunkt-Server und einem Endpunkt-Client über das Netzwerkgerät empfängt. Die sichere Sitzung zwischen dem Endpunkt-Server und dem Endpunkt-Client ist in eine erste Sitzung und eine zweite Sitzung unterteilt. Die erste Sitzung findet zwischen dem Endpunkt-Server und dem Netzwerkgerät statt. Die zweite Sitzung findet zwischen dem Netzwerkgerät und dem Endpunkt-Client statt. Das Netzwerkgerät empfängt ein erstes Sitzungsticket vom Endpunkt-Server. Ein Sitzungsstatus eines Proxy-Clients in der ersten Sitzung, einschließlich des ersten Sitzungstickets, wird bestimmt. Das Netzwerkgerät bestimmt auch einen Sitzungsstatus eines Proxy-Servers in der zweiten Sitzung. Die Kombination aus dem Sitzungsstatus des Proxy-Clients, einschließlich des ersten Sitzungstickets, und dem Sitzungsstatus des Proxy-Servers wird als Teil eines zweiten Sitzungstickets gekapselt.

KURZDARSTELLUNG

[0011] Von einem ersten Aspekt aus betrachtet, stellt die vorliegende Erfindung ein Verfahren bereit, das innerhalb eines Vermittlers wirksam ist, der zwischen einem Transport Layer Security-(TLS-)Client und einem TLS-Server angeordnet ist und eine TLS-Prüffunktion während einer TLS-Sitzung bereitstellt, wobei das Verfahren aufweist: nach dem Empfang eines ursprünglichen Sitzungstickets von dem TLS-Server Erzeugen eines neuen Sitzungstickets; Anwenden einer Verschlüsselungsfunktion auf das neue Sitzungsticket; Ausgeben des verschlüsselten neuen Sitzungstickets an den TLS-Client; nach dem Empfang des verschlüsselten neuen Sitzungstickets von dem TLS-Client Entschlüsseln des verschlüsselten neuen Sitzungstickets, um das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen wiederherzustellen; und Verwenden des wiederhergestellten ursprünglichen Sitzungstickets und der Sitzungskontextinformationen, um die TLS-Sitzung wieder aufzunehmen.

[0012] Von einem weiteren Aspekt aus betrachtet, stellt die vorliegende Erfindung eine Vorrichtung bereit, die aufweist: einen Prozessor, Computerspeicher, der durch den Prozessor ausgeführte Computerprogrammanweisungen enthält, wobei die Computerprogrammanweisungen Programmcode aufweisen, der so konfiguriert ist, dass er: nach dem Empfang eines ursprünglichen Sitzungstickets von dem TLS-Server ein neues Sitzungsticket erzeugt; eine Verschlüsselungsfunktion auf das neue Sitzungsticket anwendet; das verschlüsselte neue Sitzungsticket an den TLS-Client ausgibt; nach dem Empfang des verschlüsselten neuen Sitzungstickets von dem TLS-Client das verschlüsselte neue Sitzungstickets entschlüsselt, um das

ursprüngliche Sitzungsticket und die Sitzungskontextinformationen wiederherzustellen; und das wiederhergestellte ursprüngliche Sitzungsticket und die Sitzungskontextinformationen verwendet, um die TLS-Sitzung wieder aufzunehmen.

[0013] Von einem weiteren Aspekt aus betrachtet, stellt die vorliegende Erfindung ein Computerprogrammprodukt für eine Transport Layer Security-(TLS-)Prüffunktion während einer TLS-Sitzung bereit, wobei das Computerprogrammprodukt ein durch einen Computer lesbares Speichermedium aufweist, das durch eine Verarbeitungsschaltung gelesen werden kann und Anweisungen zur Ausführung durch die Verarbeitungsschaltung speichert, um ein Verfahren zur Durchführung der erfindungsgemäßen Schritte auszuführen.

[0014] Von einem weiteren Aspekt aus betrachtet, stellt die vorliegende Erfindung ein Computerprogramm bereit, das auf einem durch einen Computer lesbaren Datenträger gespeichert ist und in den internen Hauptspeicher eines digitalen Computers geladen werden kann, wobei das Computerprogramm Teile von Software-Code aufweist, wenn das Programm auf einem Computer ausgeführt wird, um die erfindungsgemäßen Schritte durchzuführen.

[0015] Von einem weiteren Aspekt aus betrachtet, stellt die vorliegende Erfindung eine zwischen einem Transport Layer Security-(TLS-)Client und einem TLS-Server angeordnete Vorrichtung bereit, die aufweist: einen Hardware-Prozessor sowie Computerspeicher, der Computerprogrammanweisungen enthält, die als ein cachespeicherloser TLS-Inspektor-Mechanismus konfiguriert sind, um: ein Sitzungsticket von dem TLS-Server zu empfangen und als Reaktion darauf: (a) ein zusammengesetztes Sitzungsticket zu erzeugen und (b) das zusammengesetzte Sitzungsticket an den TLS-Client auszugeben, anstatt das Sitzungsticket zwischenzuspeichern, und das zusammengesetzte Sitzungsticket von dem TLS-Client zu empfangen und als Reaktion darauf: (c) das Sitzungsticket wiederherzustellen und (d) das wiederhergestellte Sitzungsticket zur Wiederaufnahme einer Sitzung mit dem TLS-Server zu verwenden.

[0016] Eine netzbasierte Appliance enthält einen Mechanismus, um es der Appliance zu ermöglichen, eine TLS-Prüfung mit Sitzungswiederaufnahme bereitzustellen, ohne dass jedoch ein Sitzungscachespeicher in dem Inspektor verwaltet werden muss. Zu diesem Zweck ist der Inspektor ohne einen Sitzungscachespeicher konfiguriert und verwaltet somit nicht länger die Zuordnung zwischen einem (von dem TLS-Server empfangenen) Sitzungsticket und dem Sitzungskontext. Stattdessen ist der Inspektor so konfiguriert, dass er den TLS-Client veranlasst, an der Verwaltung des Sitzungskontexts teilzunehmen, praktisch im Namen des TLS-Inspektors. Im Betrieb, wenn der Inspektor das Sitzungsticket von dem TLS-Server erstmalig empfängt, speichert der Inspektor das Sitzungsticket nicht zwischen, sondern erzeugt ein zusammengesetztes Ticket, welches das ursprüngliche Ticket und Sitzungskontextinformationen enthält, die einen oder mehrere Sitzungsschlüssel enthalten, und gibt es an den Client aus. Üblicherweise gibt es zwei Sitzungsschlüssel, einen für die Clientseite und einen für die Serverseite. Das zusammengesetzte Ticket (bzw. das zusammengesetzte Sitzungsticket) wird vorzugsweise von dem Inspektor verschlüsselt, um die Sitzungsinformationen zu sichern. Wenn der TLS-Client das zusammengesetzte Sitzungsticket (wieder dem Inspektor) vorlegt, um die TLS-Verbindung wieder aufzunehmen, entschlüsselt der Inspektor das Ticket und ruft den Sitzungskontext direkt daraus ab. Der Inspektor verwendet dann das ursprüngliche Sitzungsticket, um die TLS Sitzung mit dem TLS-Server wieder aufzunehmen. Diese Methode macht eine Suche im Cachespeicher oder selbst die Notwendigkeit, einen lokalen Sitzungscachespeicher an dem TLS-Inspektor zu verwalten, überflüssig. Vielmehr wird das zusammengesetzte Ticket praktisch zum Cachespeicher für das Sitzungsticket selbst.

[0017] Verallgemeinernd gesagt und gemäß einem ersten Aspekt dieser Offenbarung ist ein Verfahren innerhalb eines Vermittlers, der zwischen einem Client und einem Server angeordnet ist und eine TLS-Prüffunktion bereitstellt, wirksam. Nach dem Empfang eines ursprünglichen Sitzungstickets von dem TLS-Server wird ein neues Sitzungsticket erzeugt, anstatt das ursprüngliche Sitzungsticket in einem Sitzungscachespeicher zwischenzuspeichern. Das neue Sitzungsticket weist einen Wert auf, der abgeleitet wird, indem eine Umsetzungsfunktion auf das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen (nämlich die Cipher-Suite, der Masterschlüssel und dergleichen, was zur Verwendung während der Sitzung ausgehandelt wird) angewendet wird. Die Umsetzungsfunktion kann variieren. Eine repräsentative Umsetzungsfunktion verknüpft die Sitzungskontextinformationen mit dem ursprünglichen Sitzungsticket. Eine Verschlüsselungsfunktion wird dann auf das neue Sitzungsticket (das hierin zuweilen als ein zusammengesetztes Sitzungsticket bezeichnet wird) angewendet, um die Sitzungsinformationen zu sichern. Das verschlüsselte umgesetzte Sitzungsticket wird dann dem anfordernden TLS-Client übermittelt. Wenn dieses Ticket später von dem TLS-Client zurückempfangen wird, wird es entschlüsselt, um das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen wiederherzustellen. Unter Verwendung der wiederhergestellten Informationen wird die TLS-Sitzung mit dem TLS-Server dann wieder aufgenommen.

[0018] Gemäß einem zweiten Aspekt dieser Offenbarung ist eine Vorrichtung zwischen einem Transport Layer Security-(TLS-)Client und einem TLS-Server angeordnet, um eine TLS-Prüffunktion während einer TLS-Sitzung bereitzustellen. Die Vorrichtung weist einen Satz von einem oder mehreren Hardware-Prozessoren sowie Computerspeicher auf, der durch die Hardware-Prozessoren ausgeführte Computerprogrammanweisungen enthält, um einen Satz von Operationen, wie beispielsweise die vorstehend beschriebenen Verfahrensschritte, durchzuführen.

[0019] Gemäß einem dritten Aspekt dieser Offenbarung wird ein Computerprogrammprodukt in einem nicht flüchtigen, durch einen Computer lesbaren Datenträger zur Verwendung in einem Datenverarbeitungssystem beschrieben. Das Datenverarbeitungssystem ist zwischen einem Transport Layer Security-(TLS-)Client und einem TLS-Server angeordnet, um eine TLS-Prüffunktion während einer TLS-Sitzung bereitzustellen. Das Computerprogrammprodukt enthält Computerprogrammanweisungen, die in dem Datenverarbeitungssystem ausgeführt werden, und ist so konfiguriert, dass es Operationen, wie beispielsweise die vorstehend beschriebenen Verfahrensschritte, durchführt.

[0020] Das Vorstehende gab einen Überblick über einige der relevanteren Merkmale des offenbarten Gegenstands. Diese Merkmale sollten als lediglich veranschaulichende Merkmale aufgefasst werden. Viele andere vorteilhafte Ergebnisse können erzielt werden, indem der offenbarte Gegenstand auf eine andere Weise angewendet oder indem der Gegenstand geändert wird, wie beschrieben werden wird.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0021] Für ein vollständigeres Verständnis des Gegenstands und seiner Vorteile wird nun Bezug auf die folgenden Beschreibungen in Zusammenschau mit den beiliegenden Zeichnungen genommen, bei denen:

Fig. 1 ein beispielhaftes Blockschaubild einer verteilten Datenverarbeitungsumgebung darstellt, in der beispielhafte Aspekte der veranschaulichenden Ausführungsformen ausgeführt werden können;

Fig. 2 ein beispielhaftes Blockschaubild eines Datenverarbeitungssystems ist, in dem beispielhafte Aspekte der veranschaulichenden Ausführungsformen ausgeführt werden können;

Fig. 3 eine beispielhafte netzbasierte sichere Appliance veranschaulicht, in der der offenbarte Gegenstand ausgeführt werden kann;

Fig. 4 veranschaulicht, wie eine herkömmliche SSL/TLS-Übertragung in einer Man-in-the-Middle-Appliance verarbeitet wird, um die Prüfung von sicherem Verkehr zu vereinfachen;

Fig. 5 eine Wiederaufnahme einer TLS-Sitzung unter Verwendung eines TLS-Inspektors veranschaulicht, der einen Sitzungscachespeicher enthält; und

Fig. 6 eine Wiederaufnahme einer TLS-Sitzung gemäß dem Gegenstand dieser Offenbarung darstellt, wobei der TLS-Inspektor eine cachespeicherlose Sitzungsticket-Unterstützung bereitstellt.

AUSFÜHRLICHE BESCHREIBUNG EINER VERANSCHAULICHENDEN AUSFÜHRUNGSFORM

[0022] Unter Bezugnahme auf die Zeichnungen und insbesondere unter Bezugnahme auf die **Fig.** 1 bis **Fig.** 2 sind beispielhafte Schaubilder von Datenverarbeitungsumgebungen bereitgestellt, in denen veranschaulichende Ausführungsformen der Offenbarung ausgeführt werden können. Es sollte sich verstehen, dass die **Fig.** 1 bis **Fig.** 2 lediglich beispielhaft sind und dass sie in Bezug auf die Umgebungen, in denen Aspekte oder Ausführungsformen des offenbarten Gegenstands umgesetzt werden können, keinerlei Beschränkung bestätigen oder andeuten sollen. Viele Änderungen an den dargestellten Umgebungen können vorgenommen werden, ohne vom Umfang der vorliegenden Erfindung abzuweichen.

Client-Server-Technologien

[0023] Unter Bezugnahme auf die Zeichnungen ist **Fig.** 1 eine bildliche Darstellung eines beispielhaften verteilten Datenverarbeitungssystems, in dem Aspekte der veranschaulichenden Ausführungsformen umgesetzt werden können. Das verteilte Datenverarbeitungssystem 100 kann ein Netzwerk aus Computern enthalten, in dem Aspekte der veranschaulichenden Ausführungsformen umgesetzt werden können. Das verteilte Datenverarbeitungssystem 100 enthält mindestens ein Netzwerk 102, bei dem es sich um das Medium handelt, das zur Bereitstellung von Datenübertragungsverbindungen zwischen verschiedenen Einheiten und Computern verwendet wird, die in dem verteilten Datenverarbeitungssystem 100 miteinander verbunden sind. Das Netzwerk 102 kann Verbindungen wie beispielsweise drahtgebundene, drahtlose Datenübertragungsverbindungen oder Lichtwellenleiterkabel enthalten.

[0024] In dem dargestellten Beispiel sind der Server 104 und der Server 106 nebst der Speichereinheit 108 mit dem Netzwerk 102 verbunden. Ferner sind die Clients 110, 112 und 114 ebenfalls mit dem Netzwerk 102 verbunden. Diese Clients 110, 112 und 114

können zum Beispiel Personal Computer, Netzwerkcomputer oder dergleichen sein. In dem dargestellten Beispiel stellt der Server 104 den Clients 110, 112 und 114 Daten wie beispielsweise Bootdateien, Betriebssystem-Abbilder und Anwendungen bereit. Die Clients 110, 112 und 114 sind Clients des Servers 104 in dem dargestellten Beispiel. Das verteilte Datenverarbeitungssystem 100 kann zusätzliche Server, Clients und andere Einheiten enthalten, die nicht gezeigt sind.

[0025] In dem dargestellten Beispiel ist das verteilte Datenverarbeitungssystem 100 das Internet, wobei das Netzwerk 102 einen weltweiten Verbund von Netzwerken und Gateways darstellt, die die Transmission Control Protocol/Internet Protocol-(TCP/IP-) Folge von Protokollen verwenden, um untereinander Daten auszutauschen. Im Mittelpunkt des Internets befindet sich ein Backbone-Netz aus Hochgeschwindigkeits-Datenübertragungsleitungen zwischen Hauptknoten oder Hostcomputern, die aus Tausenden von kommerziellen, staatlichen, in Bildungseinrichtungen genutzten und anderen Computersystemen bestehen, welche Daten und Nachrichten weiterleiten. Natürlich kann das verteilte Datenverarbeitungssystem 100 auch so ausgeführt sein, dass mehrere verschiedene Arten von Netzwerken wie zum Beispiel ein Intranet, ein lokales Netz (LAN), ein Weitverkehrsnetz (WAN) oder dergleichen zu ihm gehören. Wie vorstehend angegeben ist, ist **Fig.** 1 als ein Beispiel gedacht, nicht als eine architektonische Einschränkung für verschiedene Ausführungsformen des offenbarten Gegenstands, und daher sollten die jeweiligen in **Fig.** 1 gezeigten Elemente in Bezug auf die Umgebungen, in denen die veranschaulichenden Ausführungsformen der vorliegenden Erfindung ausgeführt werden können, nicht als einschränkend betrachtet werden.

[0026] Unter Bezugnahme auf **Fig.** 2 ist ein Blockschaubild eines beispielhaften Datenverarbeitungssystems gezeigt, in dem Aspekte der veranschaulichenden Ausführungsformen umgesetzt werden können. Das Datenverarbeitungssystem 200 ist ein Beispiel eines Computers wie des Clients 110 in **Fig.** 1, in dem sich durch einen Computer verwendbarer Code oder durch einen Computer verwendbare Anweisungen befinden können, welcher/welche die Prozesse für veranschaulichende Ausführungsformen der Offenbarung ausführt/ausführen.

[0027] Unter Bezugnahme auf **Fig.** 2 ist ein Blockschaubild eines Datenverarbeitungssystems gezeigt, in dem veranschaulichende Ausführungsformen ausgeführt werden können. Das Datenverarbeitungssystem 200 ist ein Beispiel eines Computers wie beispielsweise des Servers 104 oder des Clients 110 in **Fig.** 1, in dem sich durch einen Computer verwendbare(r) Programmcode oder Anweisungen, welcher/welche die Prozesse ausführen, für die veranschaulichenden Ausführungsformen befinden können. In diesem veranschaulichenden Beispiel enthält das Datenverarbeitungssystem 200 eine Übertragungsstruktur 202, die Übertragungen zwischen der Prozessoreinheit 204, dem Hauptspeicher 206, dem persistenten Speicher 208, der Übertragungseinheit 210, der Eingabe-/Ausgabe-(E/A-)Einheit 212 und dem Bildschirm 214 bereitstellt.

[0028] Die Prozessoreinheit 204 dient zur Ausführung von Anweisungen für Software, die in den Hauptspeicher 206 geladen werden kann. Bei der Prozessoreinheit 204 kann es sich in Abhängigkeit von der jeweiligen Ausführung um einen Satz aus einem oder mehreren Prozessoren oder um einen Mehrprozessorkern handeln. Des Weiteren kann die Prozessoreinheit 204 unter Verwendung von einem oder mehreren heterogenen Prozessorsystemen ausgeführt sein, bei denen sich ein Hauptprozessor mit sekundären Prozessoren auf einem einzigen Chip befindet. Als ein weiteres veranschaulichendes Beispiel kann es sich bei der Prozessoreinheit 204 um ein symmetrisches Mehrprozessor-(SMP-)System handeln, das mehrere Prozessoren von demselben Typ enthält.

[0029] Der Hauptspeicher 206 und der persistente Speicher 208 sind Beispiele für Speichereinheiten. Eine Speichereinheit ist ein beliebiges Stück Hardware, das Informationen entweder auf temporärer Basis und/oder auf permanenter Basis speichern kann. Der Hauptspeicher 206 in diesen Beispielen kann zum Beispiel ein Direktzugriffsspeicher oder eine beliebige andere geeignete flüchtige oder nicht flüchtige Speichereinheit sein. Der persistente Speicher 208 kann in Abhängigkeit von der jeweiligen Ausführung verschiedene Formen annehmen. Zum Beispiel kann der persistente Speicher 208 eine oder mehrere Komponenten oder Einheiten enthalten. Zum Beispiel kann der persistente Speicher 208 ein Festplattenlaufwerk, ein Flashspeicher, eine wieder beschreibbare optische Platte, ein wieder beschreibbares Magnetband oder eine Kombination des Vorstehenden sein. Der von dem persistenten Speicher 208 verwendete Datenträger kann auch auswechselbar sein. Zum Beispiel kann ein auswechselbares Festplattenlaufwerk für den persistenten Speicher 208 verwendet werden.

[0030] Die Übertragungseinheit 210 ermöglicht in diesen Beispielen den Datenaustausch mit anderen Datenverarbeitungssystemen oder -einheiten. In diesen Beispielen ist die Übertragungseinheit 210 eine Netzschnittstellenkarte. Die Übertragungseinheit 210 kann durch die Verwendung von physischen oder drahtlosen oder aber durch die Verwendung von physischen als auch drahtlosen Datenübertragungsverbindungen Übertragungen bereitstellen.

[0031] Die Eingabe-/Ausgabeeinheit 212 ermöglicht die Ein- und Ausgabe von Daten mit anderen Einheiten, die mit dem Datenverarbeitungssystem 200 verbunden sein können. Zum Beispiel kann die Eingabe-/Ausgabeeinheit 212 eine Verbindung für eine Benutzereingabe über eine Tastatur und eine Maus bereitstellen. Des Weiteren kann die Eingabe-/Ausgabeeinheit 212 eine Ausgabe an einen Drucker senden. Der Bildschirm 214 stellt einen Mechanismus bereit, um einem Benutzer Informationen anzuzeigen.

[0032] Anweisungen für das Betriebssystem und Anwendungen oder Programme befinden sich im persistenten Speicher 208. Diese Anweisungen können zur Ausführung durch die Prozessoreinheit 204 in den Hauptspeicher 206 geladen werden. Die Prozesse der verschiedenen Ausführungsformen können durch die Prozessoreinheit 204 unter Verwendung von durch einen Computer ausgeführten Anweisungen durchgeführt werden, welche sich in einem Hauptspeicher wie beispielsweise dem Hauptspeicher 206 befinden können. Diese Anweisungen werden als Programmcode, durch einen Computer verwendbarer Programmcode oder durch einen Computer lesbarer Programmcode bezeichnet, der von einem Prozessor in der Prozessoreinheit 204 gelesen und ausgeführt werden kann. Der Programmcode in den verschiedenen Ausführungsformen kann auf verschiedenen physischen oder physisch greifbaren, durch einen Computer lesbaren Datenträgern wie beispielsweise dem Hauptspeicher 206 oder dem persistenten Speicher 208 ausgebildet sein.

[0033] Der Programmcode 216 befindet sich in funktionaler Form auf einem durch einen Computer lesbaren Datenträger 218, der selektiv auswechselbar ist, und er kann zur Ausführung durch die Prozessoreinheit 204 auf das Datenverarbeitungssystem 200 geladen oder an es übertragen werden. Der Programmcode 216 und der durch einen Computer lesbare Datenträger 218 bilden in diesen Beispielen das Computerprogrammprodukt 220. In einem Beispiel kann der durch einen Computer lesbare Datenträger 218 in physisch greifbarer Form wie zum Beispiel einer optischen Platte oder einer Magnetplatte vorliegen, die in ein Laufwerk oder in eine andere Einheit eingelegt oder eingesetzt wird, das bzw. die Teil des persistenten Speichers 208 ist, um an eine Speichereinheit wie beispielsweise ein Festplattenlaufwerk übertragen zu werden, das Teil des persistenten Speichers 208 ist. In einer physisch greifbaren Form kann der durch einen Computer lesbare Datenträger 218 auch die Form eines persistenten Speichers wie beispielsweise eines Festplattenlaufwerks, eines Thumb-Drives oder eines Flashspeichers annehmen, der mit dem Datenverarbeitungssystem 200 verbunden ist. Die physisch greifbare Form des durch einen Computer lesbaren Datenträgers 218 wird auch als ein durch einen Computer beschreibbares Speichermedium bezeichnet. In einigen Fällen ist der durch einen Computer lesbare Datenträger 218 gegebenenfalls nicht auswechselbar.

[0034] Alternativ kann der Programmcode 216 von dem durch einen Computer lesbaren Datenträger 218 über eine Datenübertragungsverbindung zu der Übertragungseinheit 210 und/oder über eine Verbindung zu der Eingabe-/Ausgabeeinheit 212 an das Datenverarbeitungssystem 200 übertragen werden. Die Datenübertragungsverbindung und/oder die Verbindung kann in den veranschaulichenden Beispielen physisch oder drahtlos sein. Der durch einen Computer lesbare Datenträger kann auch die Form eines nicht physisch greifbaren Datenträgers, wie beispielsweise Datenübertragungsverbindungen oder drahtlose Übertragungen, die den Programmcode enthalten, annehmen. Die für das Datenverarbeitungssystem 200 veranschaulichten verschiedenen Komponenten sind nicht dazu gedacht, architektonische Einschränkungen für die Art und Weise, in der verschiedene Ausführungsformen ausgeführt werden können, vorzusehen. Die verschiedenen veranschaulichenden Ausführungsformen können in einem Datenverarbeitungssystem ausgeführt werden, das zusätzlich zu den oder anstelle der Komponenten, die für das Datenverarbeitungssystem 200 veranschaulicht sind, Komponenten enthält. Andere in **Fig.** 2 gezeigte Komponenten können sich von den gezeigten veranschaulichten Beispielen unterscheiden. Als ein Beispiel ist eine Speichereinheit in dem Datenverarbeitungssystem 200 eine beliebige Hardware-Vorrichtung, die Daten speichern kann. Der Hauptspeicher 206, der persistente Speicher 208 und der durch einen Computer lesbare Datenträger 218 sind Beispiele für Speichereinheiten in einer physisch greifbaren Form.

[0035] In einem weiteren Beispiel kann ein Bussystem verwendet werden, um die Übertragungsstruktur 202 auszuführen, und es kann aus einem oder mehreren Bussen bestehen, wie beispielsweise einem Systembus oder einem E/A-Bus. Natürlich kann das Bussystem unter Verwendung einer beliebigen geeigneten Art von Architektur ausgeführt sein, die eine Übertragung von Daten zwischen verschiedenen, an das Bussystem angeschlossenen Komponenten oder Einheiten ermöglicht. Ferner kann eine Übertragungseinheit eine oder mehrere Einheiten enthalten, die zum Senden und Empfangen von Daten verwendet werden, wie beispielsweise ein Modem oder einen Netzwerkadapter. Des Weiteren kann ein Hauptspeicher zum Beispiel der Hauptspeicher 206 oder ein Cachespeicher sein, wie er in einer Schnittstelle und einem Hauptspeicher-Controller-Hub zu finden ist, die bzw. der in der Übertragungsstruktur 202 vorhanden sein kann.

[0036] Computerprogrammcode zur Durchführung von Operationen der vorliegenden Erfindung kann in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen, darunter einer objektorientierten Programmiersprache wie beispielsweise Java™, Smalltalk, C++, C#, Objective-C oder dergleichen, sowie in herkömmlichen prozeduralen Programmiersprachen geschrieben sein. Der Programmcode kann vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem fernen Computer oder vollständig auf dem fernen Computer oder Server ausgeführt werden. In letzterem Fall kann der entfernt angeordnete Computer mit dem Computer des Benutzers durch eine beliebige Art Netzwerk verbunden sein, darunter ein lokales Netzwerk (LAN) oder ein Weitverkehrsnetz (WAN), oder die Verbindung kann mit einem externen Computer hergestellt werden (zum Beispiel über das Internet unter Verwendung eines Internet-Dienstanbieters). Java und alle auf Java beruhenden Warenzeichen und Logos sind Warenzeichen oder eingetragene Warenzeichen von Oracle und/oder von Tochterfirmen von Oracle.

[0037] Der Fachmann versteht, dass die Hardware in den **Fig.** 1 bis **Fig.** 2 in Abhängigkeit von der Ausführung variieren kann. Andere interne Hardware oder periphere Einheiten wie beispielsweise ein Flashspeicher, ein gleichwertiger nicht flüchtiger Speicher oder optische Plattenlaufwerke und dergleichen können zusätzlich zu oder anstelle der in den **Fig.** 1 bis **Fig.** 2 dargestellten Hardware verwendet werden. Auch können die Prozesse der veranschaulichenden Ausführungsformen neben dem zuvor erwähnten SMP-System auch auf ein Datenverarbeitungssystem mit mehreren Prozessoren angewendet werden, ohne vom Umfang des offenbarten Gegenstands abzuweichen.

[0038] Wie zu sehen sein wird, können die hierin beschriebenen Techniken in Verbindung mit dem standardmäßigen Client-Server-Paradigma, wie beispielsweise dem in **Fig.** 1 veranschaulichten, arbeiten, bei dem Client-Maschinen mit einem über das Internet zugänglichen, webbasierten Portal Daten austauschen, das auf einem Satz aus einer oder mehreren Maschinen ausgeführt wird. Endbenutzer bedienen an das Internet anschließbare Einheiten (z.B. Desktop-Computer, Notebook-Computer, internetfähige mobile Geräte oder dergleichen), die auf das Portal zugreifen und mit dem Portal interagieren können. Üblicherweise ist jede Client- oder Server-Maschine ein Datenverarbeitungssystem, wie es beispielsweise in **Fig.** 2 veranschaulicht ist, das Hardware und Software aufweist, und diese Entitäten tauschen miteinander über ein Netzwerk wie beispielsweise das Internet, ein Intranet, ein Extranet, ein privates Netzwerk oder ein(e) beliebige(s) andere(s) Datenübertragungsmedium oder -verbindung Daten aus. Ein Datenverarbeitungssystem enthält üblicherweise einen oder mehrere Prozessoren, ein Betriebssystem, eine oder mehrere Anwendungen und ein oder mehrere Dienstprogramme. Die Anwendungen auf dem Datenverarbeitungssystem stellen native Unterstützung für Web-Services bereit, darunter, ohne darauf beschränkt zu sein, Unterstützung für HTTP, SOAP, XML, WSDL, UDDI sowie WSFL u.a. Informationen zu SOAP, WSDL, UDDI und WSFL werden vom World Wide Web Consortium (W3C) zur Verfügung gestellt, das für die Entwicklung und die Pflege dieser Standards zuständig ist; weitere Informationen zu HTTP und XML werden von der Internet Engineering Task Force (IETF) zur Verfügung gestellt. Vertrautheit mit diesen Standards wird vora usgesetzt.

[0039] Als weiterer Hintergrund ist Secure Sockets Layer/Transport Layer Security (SSL/TLS) ein hinlänglich bekanntes Verschlüsselungsprotokoll, das verwendet wird, um Übertragungen über Netzwerke wie beispielsweise das Internet zu sichern. Verschlüsselungsprotokolle wie beispielsweise SSL/TLS beruhen oftmals auf Verschlüsselungssystemen mit öffentlichem Schlüssel, wie etwa dem RSA-Verschlüsselungsalgorithmus von Rivest, Shamir und Adelman. Für eine herkömmliche RSA-basierte SSL-Sitzung einigen sich die beiden Seiten einer Verbindung auf einen geheimen Pre-Master-Secret-Wert (PMS, pre-master secret), der verwendet wird, um die Parameter für die restliche Sitzung zu erzeugen. Üblicherweise verwenden die beiden Seiten eine RSA-asymmetrische Verschlüsselung, um den geheimen Pre-Master-Secret-Wert festzulegen, ohne den tatsächlichen Wert in Klartext auszutauschen. Im Betrieb erzeugt der SSL-Client den geheimen Pre-Master-Secret-Wert und verschlüsselt ihn mit dem öffentlich zugänglichen RSA-Schlüssel des SSL-Servers. Dies erzeugt einen verschlüsselten geheimen Pre-Master-Secret-Wert (ePMS, encrypted pre-master secret), der dann dem SSL-Server bereitgestellt wird. Der SSL-Server hat einen privaten Entschlüsselungsschlüssel, mit dem dann der verschlüsselte geheime Pre-Master-Secret-Wert entschlüsselt wird. An dieser Stelle verfügen sowohl der Client als auch der Server über den ursprünglichen geheimen Pre-Master-Secret-Wert und können ihn zur Erzeugung des symmetrischen Schlüssels verwenden, der für den tatsächlichen verschlüsselten und sicheren Datenaustausch verwendet wird.

[0040] Verschlüsselter Verkehr im Web findet durch eine Zertifikatskette statt. Jeder Webserver hat ein Zertifikat, das er jedem Client (gewöhnlich einem Webbrowser) vorlegt, aus dem hervorgeht, dass er derjenige ist, für den er sich ausgibt. Webserver erhalten diese Zertifikate oftmals von einer Stelle (einer Zertifizierungsstelle (Certificate Authority oder CA)), die für die Rechtmäßigkeit des Webservers

bürgen kann. Das Zertifikat des Servers gibt die Stelle an, von der das Zertifikat erhalten wurde (den „Aussteller“). Webbrowser haben üblicherweise eine Liste von Ausstellern, denen sie vertrauen. Wenn einem Webbrowser ein Zertifikat von einem Webserver vorgelegt wird, prüft der Browser den Aussteller und vergleicht ihn mit seiner vertrauenswürdigen Liste. Wenn eine Übereinstimmung festgestellt wird, wird die Verbindung fortgesetzt; wenn keine Übereinstimmung festgestellt wird, gibt der Browser gewöhnlich eine Warnung aus und weist die Verbindung möglicherweise zurück. Abgesehen davon, dass sie vertrauenswürdig ist, ist eine CA nicht unbedingt eine besondere Entität. Jede beliebige Entität kann sich so einrichten, dass sie Zertifikaten vertraut oder Zertifikate signiert. Ein Zertifikat kann sich selbst vertrauen, was als ein selbst signiertes Zertifikat bezeichnet wird. Um mit einem Client unter Verwendung von SSL/TLS zusammenzuarbeiten, ist es notwendig, Zertifikate zu erstellen, denen der Client stillschweigend vertraut. Mit Bezug auf eine Netzappliance (wie nachstehend beschrieben) wird davon ausgegangen, dass ein Administrator Unternehmensclients so konfigurieren kann, dass sie der Appliance dahingehend vertrauen, dass sie Zertifikate signiert. Tatsächlich befindet sich der Aussteller der Appliance dann auf der Liste des Browsers der vertrauenswürdigen Aussteller.

Mit dem Netz verbundene sichere Appliances

[0041] Eine Netzappliance ist üblicherweise eine Einschubeinheit. Die Einheit enthält physische Sicherheit, die es der Appliance ermöglicht, als ein sicherer Vault für kritische Informationen zu dienen. Üblicherweise wird die Appliance mit vorinstallierter Software hergestellt und dann innerhalb eines oder zusammen mit einem Unternehmen(s) oder einer anderen Netzwerkbetriebsumgebung eingesetzt; alternativ kann die Box lokal angeordnet und dann mit standardmäßigen oder kundenspezifischen virtuellen Middleware-Abbildern versehen werden, die sicher eingesetzt und verwaltet werden können, z.B. innerhalb einer privaten oder in den eigenen Räumen befindlichen Cloud-Computing-Umgebung. Die Appliance kann kryptografische Hardware- und Firmware-Unterstützung enthalten, möglicherweise, um Daten auf Festplatte zu verschlüsseln. Keine Benutzer, darunter administrative Benutzer, können auf irgendwelche Daten auf einer physischen Platte zugreifen. Im Einzelnen sperrt vorzugsweise das Betriebssystem (z.B. Linux ®) das Root-Account und stellt keine Befehlsshell bereit und der Benutzer hat keinen Zugriff auf das Dateisystem. Üblicherweise enthält die Appliance keinen Bildschirm, kein CD- oder anderes optisches Laufwerk oder irgendwelche USB-, Firewire- oder andere Ports, damit Einheiten daran angeschlossen werden können. Sie ist als eine abgeschlossene und sichere Umgebung mit eingeschränkter Zugänglichkeit und dann nur für authentifizierte und autorisierte Einzelpersonen konzipiert. Linux ist ein eingetragenes Warenzeichen von Linus Torvalds in den Vereinigten Staaten von Amerika, in anderen Ländern oder sowohl in den Vereinigten Staaten von Amerika als auch in anderen Ländern.

[0042] Unter Bezugnahme auf **Fig.** 3 enthält eine repräsentative Betriebsumgebung die physische Appliance 300, die über eine Schnittstelle mit einem Netzwerk 302 verbunden ist. Die Appliance kann unter Verwendung eines Datenverarbeitungssystems wie beispielsweise des vorstehend unter Bezugnahme auf **Fig.** 2 beschriebenen ausgeführt sein und sie kann einen der in **Fig.** 1 gezeigten Server (oder Clients) darstellen. Üblicherweise enthält die Appliance 300 eine Web 2.0-basierte Benutzerschnittstelle (UI, user interface), eine Befehlszeilenschnittstelle (CLI, command line interface) und REST-basierte Anwendungsprogrammierschnittstellen (APIs, application programming interfaces). In diesem Beispiel wurde die Appliance mit einem Abbild versehen, das ein Betriebssystem 304, einen Anwendungsserver 306, einen HTTP-Server 308 sowie andere Anwendungsprogramme 310 aufweist. Zusätzliche Software-Lösungen (nicht gezeigt) können in das Abbild aufgenommen werden. Diese Software-Elemente sind gegebenenfalls in der Appliance vorinstalliert, die andere Daten (z.B. Vorlagen, Skripte, Dateien usw.) enthalten kann. Die jeweilige Softwarekonfiguration hängt natürlich davon ab, welcher Verwendung die Appliance zugeführt wird. Die Appliance enthält eine oder mehrere Speichereinheiten (z.B. eine Platte 315). Die Art und die Anzahl der Speichereinheiten kann variieren.

Abfangen, Entschlüsselung und Prüfung von sicheren Netzwerkübertragungen

[0043] Rein der weiteren Hintergrundinformation halber veranschaulicht **Fig.** 4 den grundlegenden Betrieb einer bekannten Man-in-the-Middle-(MITM-) Einheit 400, um sichere Netzwerkübertragungen gemäß einer bekannten Technik abzufangen, zu entschlüsseln und zu prüfen. Die Einheit ist innerhalb einer sicheren Netzappliance ausgeführt, wie sie beispielsweise vorstehend beschrieben und in **Fig.** 3 veranschaulicht ist. Allgemeiner ausgedrückt ist die Einheit ein Datenverarbeitungssystem wie es beispielsweise in **Fig.** 1 gezeigt ist.

[0044] Wie veranschaulicht ist, ist die Einheit 400 zwischen einem Client 402 und einem Server 404 angeschlossen. Der Client und der Server werden bereitgestellt, um Übertragungen unter Verwendung von SSL oder TLS zu sichern. Vertrautheit mit SSL/TLS wird vorausgesetzt. In dieser Ausführungsform stellt die Einheit 400 einen transparenten (oder Man-in-the-Middle-)Proxy zwischen dem Client 402 und dem Server 404 bereit, indem sie zwei (2)

getrennte SSL/TLS-Sitzungen aufbaut und verwaltet, eine als ein Clientprozess X_{ss} 406 zum Zielserver 404 und eine weitere als ein Serverprozess X_{cs} 408 zu dem einleitenden Client 402. Die X_{ss}- und X_{cs}-Komponenten werden hierin zuweilen als SSL-Instanzen bezeichnet, wobei eine SSL-Instanz üblicherweise ein Stück Code ist, aus dem eine SSL-Sitzung besteht. Eine SSL-Sitzung (oder Sitzungskontext) ist die Übertragung selbst, die zwischen zwei Endpunkten stattfindet. Der Zwischenproxy erscheint dem Server 404 somit als ein Client und dem Client 402 als der vorgesehene Server. Von dem Client 402 aus eingeleitete Übertragungen und etwaige von dem Server 404 empfangene Antworten stehen dann zur Prüfung (oder für eine andere Verarbeitung wie zum Beispiel Umschreiben) und eine anschließende Aktion zur Verfügung. Zu diesem Zweck kann die Einheit 400 ein Protokollanalysemodul (z.B. IBM® Security Network Protection PAM) enthalten, das eine Paketprüffunktion bereitstellt, um Netzbedrohungen zu erkennen und möglicherweise abzuschwächen. Die jeweiligen Einzelheiten des Moduls (oder von anderen Paketprüfanwendungen, die gegebenenfalls unterstützt werden) stellen keinen Aspekt dieser Offenbarung dar.

[0045] Im Betrieb und wie in **Fig.** 4 zu sehen ist, erzeugt der Client 402 im Anschluss an einen anfänglichen TCP-Handshake (nicht gezeigt) die SSL/TLS-Sitzungseinleitungsanforderungsnachricht (nachstehend als „Client-Hello-Nachricht“ bezeichnet), um den SSL/TLS-Handshake mit dem Server zu beginnen. Dies ist der Schritt 1. Der Proxy fängt diese Verbindung ab und leitet sie an die dem Client zugewandte Serverkomponente X_{cs} 408 weiter. Im Schritt 2 liest die Xcs-Komponente die Client-Hello-Nachricht, interpretiert die Daten und antwortet dem Client 402, üblicherweise mit einer Server-Hello-Nachricht, einem Zertifikat und einer Server-fertig-Nachricht (server done message). Im Schritt 3 wird eine ganz neue SSL-Verbindung konfiguriert und in der Appliance aufgebaut. Dies ist eine dem Server zugewandte Verbindung, die von der X_{ss} eingeleitet wird. X_{ss} erzeugt dann eine neue Client-Hello-Nachricht (die hier als ClientHello2 bezeichnet wird, um sie von der ClientHello im Schritt 1 zu unterscheiden) und sendet die (neue) Client-Hello-Nachricht an den Server. Im Schritt 4 liest der Server 404 die neue Client-Hello-Nachricht und antwortet mit ServerHello2, Certificate2 und ServerDone2. Diese Nachrichten unterscheiden sich jedoch noch mal von den durch X_{cs} an den Client im Schritt 2 ausgegebenen Nachrichten. Folglich gibt es zwei (2) verschiedene Verbindungen, eine zwischen dem Client 402 und X_{cs} 408 und die andere zwischen X_{ss} 406 und dem Server 404. Wenn die MITM-Verarbeitung (z.B. durch das PAM oder eine andere Anwendung) feststellt, dass dies keine (Client-Server-)Verbindung ist, die auf Wunsch geprüft werden soll, muss das System an diesem Punkt die Verbindung entweder weiterhin prüfen (und dabei die Ergebnisse vielleicht ignorieren) oder sie komplett schließen. Die Feststellung kann in beliebiger geeigneter Weise erfolgen, z.B., indem ein richtlinienbasierter Regelabgleich mit Informationen in dem von dem Server empfangenen Zertifikat (dem vorstehenden Certificate2) erfolgt.

Cachespeicherlose Sitzungsticket-Unterstützung bei TLS-Prüfung

[0046] Mit dem Vorstehenden als Hintergrund wird die cachespeicherlose Sitzungsticket-Unterstützung bei einer TLS-Prüfung dieser Offenbarung nun beschrieben. Vertrautheit mit dem TLS Session Ticket nach Internet RFC 5077 wird vorausgesetzt. Ein repräsentatives kommerzielles Produkt, in dem die Technik ausgeführt sein kann, ist IBM® QRadar® Network Security (XGS) (früher bekannt als IBM Security Network Protection (XGS)), ein Intrusion-Prevention-System (IPS) der nächsten Generation. Natürlich soll die Bezeichnung dieses kommerziellen Produkts nicht einschränkend sein, da die Methode in jeder/jedem beliebigen zwischengeschalteten Einheit, Appliance, Produkt oder System durchgeführt werden kann. IBM und QRadar sind Warenzeichen der International Business Machines Corporation, die weltweit in vielen Ländern eingetragen sind.

[0047] Wie vorstehend erwähnt wurde und gemäß dieser Offenbarung enthält eine netzbasierte Appliance, wie beispielsweise die beschriebene, einen Mechanismus, um es der Appliance zu ermöglichen, eine TLS-Prüfung mit Sitzungswiederaufnahme bereitzustellen, ohne dass jedoch ein Sitzungscachespeicher in dem Inspektor verwaltet werden muss. Zu diesem Zweck ist der Inspektor ohne einen Sitzungscachespeicher konfiguriert (oder bei der Alternative mit einem Cachespeicher, der nicht verwendet wird) und verwaltet somit nicht länger die Zuordnung zwischen einem (von dem TLS-Server empfangenen) Sitzungsticket und dem Sitzungskontext. Stattdessen ist der Inspektor so konfiguriert, dass er den TLS-Client veranlasst, an der Verwaltung des Sitzungskontexts teilzunehmen, praktisch im Namen des TLS-Inspektors. Im Betrieb, wenn der Inspektor das Sitzungsticket von dem TLS-Server erstmalig empfängt, speichert der Inspektor das Sitzungsticket nicht zwischen, sondern erzeugt ein zusammengesetztes Ticket, welches das ursprüngliche Ticket und Sitzungskontextinformationen enthält, die den Sitzungsschlüssel enthalten, und gibt es an den Client aus. Das zusammengesetzte Ticket (bzw. das zusammengesetzte Sitzungsticket) wird vorzugsweise von dem Inspektor verschlüsselt, um die Sitzungsinformationen zu sichern. Wenn der TLS-Client das zusammengesetzte Sitzungsticket (wieder dem Inspektor) vorlegt, um die TLS-Verbindung wieder aufzunehmen, entschlüsselt der Inspektor das Ticket und ruft den Sitzungskontext direkt daraus ab. Der Inspektor verwendet dann das

ursprüngliche Sitzungsticket, um die TLS Sitzung wieder aufzunehmen. Diese Methode macht eine Suche im Cachespeicher oder selbst die Notwendigkeit, einen lokalen Sitzungscachespeicher an dem TLS-Inspektor zu verwalten, überflüssig. Vielmehr wird das zusammengesetzte Ticket praktisch zum Cachespeicher für das Sitzungsticket selbst.

[0048] Wie hinlänglich bekannt ist, wird davon ausgegangen, dass die Appliance einen Mechanismus enthält, um es dem MITM-Prozess (ob proxybasiert oder in anderer Weise) zu ermöglichen, die ursprüngliche Verbindung wieder aufzunehmen und die ursprünglichen Endpunkte (ohne Prüfung) wieder zu verbinden, z.B. nach einem Regelabgleich mit einem Serverzertifikat oder in anderer Weise. Session Ticket bei TLS-Unterstützung wird zu diesem Zweck verwendet.

[0049] **Fig.** 5 stellt einen Ablaufplan dar, der ein herkömmliches Session Ticket bei TLS zeigt, nämlich mit einem Inspektor 500, der einen Sitzungscachespeicher enthält. Vertrautheit mit der standardmäßigen TLS-Handshaking-Semantik wird vorausgesetzt. Wie gezeigt ist, veranschaulicht der Ablaufplan in **Fig.** 5 den zwischen dem TLS-Client 502 und dem TLS-Server 504 befindlichen Inspektor. Bei dem herkömmlichen TLS-Handshake-Ablauf gibt der Client 502 eine Client-Hello-Nachricht an den Server 504 aus. Dies ist der Schritt 506. Im Schritt 508 antwortet der Server 504 mit verschiedenen Nachrichten, nämlich der Server-Hello-Nachricht, dem Zertifikat, der ServerKeyExchange- und der ServerHelloDone-Nachricht. Die Sitzungskontextinformationen (Cipher-Suites, Masterschlüssel usw.) werden im Schritt 510 von dem Inspektor 500 gespeichert. Im Schritt 512 schickt der Client 502 verschiedene Nachrichten an den Server zurück, nämlich die ClientKeyExchange-Nachricht, eine optionale ChangeCipherSpec-Nachricht und eine Finished-Nachricht. Der Server 504 antwortet dann mit einem weiteren Satz von Nachrichten, nämlich NewSessionTicket, einer etwaigen Antwort auf die ChangeCipherSpec-Nachricht (sofern durch den Client gesendet) und einer Finished-Nachricht. Damit ist der anfängliche TLS-Handshake abgeschlossen. Im Schritt 516 speichert der Inspektor 500 das neue Ticket in seinem lokalen Cachespeicher. Im Betrieb, und wie im Schritt 518 dargestellt, fließen Anwendungsdaten zwischen dem Client 502 und dem Server 504. Der Inspektor 500 verwendet die Sitzungsinformationen in dem Ticket zum Entschlüsseln von Nutzlasten, wie im Schritt 520 dargestellt ist, um eine oder mehrere Inspektor-Funktionen auszuführen. Der Inspektor gibt das Sitzungsticket auch an den Client zurück.

[0050] Eine oder mehrere Operationen durch den Inspektor unterbrechen die Sitzung, so dass der Inspektor folglich in der Lage sein muss, eine Sitzungswiederaufnahme durchzuführen. Zu diesem Zweck, und wie im Schritt 522 gezeigt ist, wird davon ausgegangen, dass der Client eine weitere Client-Hello-Nachricht ausgibt, wobei er dieses Mal das Sitzungsticket übergibt.

[0051] Im Schritt 524 führt der Inspektor eine Suche nach dem Sitzungsticket in seinem lokalen Cachespeicher durch. Liegt eine Übereinstimmung mit dem Ticket vor, gibt der Inspektor 500 eine neue Client-Hello-Nachricht an den Server aus, wobei er das Sitzungsticket übergibt. Dies ist der Schritt 526. Im Schritt 528 antwortet der Server 504 mit verschiedenen Nachrichten, nämlich der Server-Hello-Nachricht, dem Zertifikat, der optionalen ChangeCipherSpec-Nachricht und einer Finished-Nachricht. Im Schritt 530 antwortet der Client 502 dann mit einem Satz von Nachrichten, nämlich einer etwaigen Antwort auf die ChangeCipherSpec-Nachricht (sofern durch den Server gesendet) und einer Finished-Nachricht. Damit ist der TLS-Folgehandshake abgeschlossen, der für eine Sitzungswiederaufnahme notwendig wäre. Im Schritt 532 erhält der Inspektor 500 den einen oder die mehreren Sitzungsschlüssel (die im Schritt 510 gespeichert wurden) aus seinem Cachespeicher. Üblicherweise gibt es zwei Sitzungsschlüssel, einen für jede Sitzung (Client <-> Inspektor und Inspektor <-> Server). Während Anwendungsdaten dann zwischen Client und Server übermittelt werden (Schritt 532), verwendet der Inspektor die Sitzungsschlüssel, um Nutzlasten nach Bedarf zu entschlüsseln. Dies ist der Schritt 534.

[0052] **Fig.** 6 ist ähnlich der **Fig.** 5, jedoch stellt diese Zeichnung die Änderungen an der herkömmlichen Sitzungs-Zwischenspeicherungsmethode gemäß der Technik dieser Offenbarung dar. Wie beschrieben wurde, ermöglicht die geänderte Technik eine „cachespeicherlose“ Session-Ticket-Unterstützung bei einer TLS-Prüfung. Zu diesem Zweck ist der Inspektor so konfiguriert, dass er die Zuordnung zwischen einem (von dem TLS-Server empfangenen) Sitzungsticket und dem Sitzungskontext nicht länger verwaltet. Stattdessen ist der Inspektor so konfiguriert, dass er den TLS-Client veranlasst, an der Verwaltung des Sitzungskontexts teilzunehmen, praktisch im Namen des TLS-Inspektors. Diese Methode macht eine Suche im Cachespeicher oder selbst die Notwendigkeit, einen lokalen Sitzungscachespeicher an dem TLS-Inspektor zu verwalten, überflüssig. Vielmehr stellen die Interaktionen zwischen dem Inspektor und dem Client praktisch eine Möglichkeit für den Inspektor dar, seinen eigenen Sitzungscachespeicher von der Zwischenspeicherung zu entlasten und diese an den anfordernden Client zu übertragen, der das Sitzungsticket letzten Endes speichert (wenn auch in einer geänderten Form). Der Client wird praktisch selbst zum Cachespeicher für das Sitzungsticket.

[0053] Im Betrieb, wenn der Inspektor das Sitzungsticket von dem TLS-Server erstmalig empfängt, speichert der Inspektor das Sitzungsticket nicht zwischen, sondern erzeugt ein zusammengesetztes Ticket, welches das ursprüngliche Ticket und Sitzungskontextinformationen enthält, die den Sitzungsschlüssel enthalten, und gibt es an den Client aus. Das zusammengesetzte Ticket (bzw. das zusammengesetzte Sitzungsticket) wird vorzugsweise von dem Inspektor verschlüsselt, um die Sitzungsinformationen zu sichern. Wenn der TLS-Client das zusammengesetzte Sitzungsticket (wieder dem Inspektor) vorlegt, um die TLS-Verbindung wieder aufzunehmen, entschlüsselt der Inspektor das Ticket und ruft den Sitzungskontext direkt daraus ab. Der Inspektor verwendet dann das ursprüngliche Sitzungsticket, um die TLS Sitzung mit dem TLS-Server wieder aufzunehmen.

[0054] Die cachespeicherlose Sitzungsticket-Unterstützung ist in **Fig.** 6 dargestellt. Wieder ist der Inspektor 600 zwischen dem TLS-Client 602 und dem TLS-Server 604 angeordnet gezeigt. Im Gegensatz zu **Fig.** 5 braucht der Inspektor 600 keinen Sitzungscachespeicher zu enthalten. Wie bei dem herkömmlichen TLS-Handshake-Ablauf gibt der Client 602 eine Client-Hello-Nachricht an den Server 604 aus. Dies ist der Schritt 606. Im Schritt 608 antwortet der Server 404 mit verschiedenen Nachrichten, nämlich der Server-Hello-Nachricht, dem Zertifikat, der ServerKeyExchange-Nachricht und der ServerHelloDone-Nachricht. Wieder werden die Sitzungskontextinformationen (Cipher-Suites, Masterschlüssel usw.) im Schritt 610 durch den Inspektor 600 gespeichert. Im Schritt 612 schickt der Client 602 verschiedene Nachrichten zurück, nämlich die ClientKeyExchange-Nachricht, eine optionale ChangeCipherSpec-Nachricht und eine Finished-Nachricht. Dieses Mal werden diese Nachrichten jedoch nicht an den Server weitergereicht (wie in **Fig.** 5), sondern von dem Inspektor abgefangen. Im Schritt 614 stellt der Server 604 die NewSessionTicket-Nachricht, eine optionale ChangeCipherSpec-Nachricht und eine Finished-Nachricht bereit. Die NewSessionTicket-Nachricht enthält ein ursprüngliches Sitzungsticket. Im Schritt 616 und anstelle der Speicherung des durch den Server 604 bereitgestellten ursprünglichen Sitzungstickets (und wie im Schritt 516 in **Fig.** 5) erzeugt der Inspektor 600 ein neues, mit [A] bezeichnetes Sitzungsticket. Weitere Einzelheiten einer bevorzugten Technik zum Erzeugen des neuen Sitzungstickets sind nachstehend beschrieben. Der Inspektor 600 gibt an den Client 602 dann einen Satz von Nachrichten aus, nämlich NewSessionTicket[A], eine optionale ChangeCipherSpec-Nachricht und eine Finished-Nachricht. Dies ist der Schritt 618. Die NewSessionTicket[A-Nachricht enthält das neue Sitzungsticket [A]. Im Schritt 620 fließen Anwendungsdaten zwischen dem Client 602 und dem Server 604. Der Inspektor 600 verwendet die Sitzungsinformationen in dem Ticket zur Entschlüsselung von Nutzlasten, wie im Schritt 622 dargestellt ist, um eine oder mehrere Inspektor-Funktionen auszuführen.

[0055] Wie zuvor unterbrechen eine oder mehrere Operationen durch den Inspektor die Sitzung, so dass der Inspektor 600 folglich in der Lage sein muss, eine Sitzungswiederaufnahme durchzuführen. Zu diesem Zweck, und wie im Schritt 624 dargestellt ist, wird davon ausgegangen, dass der Client eine weitere Client-Hello-Nachricht ausgibt, wobei er dieses Mal das neue Sitzungsticket [A] an den Inspektor zurückgibt. Im Schritt 626 entschlüsselt der Inspektor 600 das neue Sitzungsticket [A], um das ursprüngliche Sitzungsticket (das als [B] bezeichnet wird) wiederherzustellen und um die Sitzungsinformationen wiederherzustellen. Im Schritt 628 ersetzt der Inspektor 600 das Ticket [A] durch das Ticket [B] in der Client-Hello-Nachricht und im Schritt 630 sendet der Inspektor 600 die Client-Hello-Nachricht (mit dem Sitzungsticket (Session Ticket) [B]) an den Server 604. Im Schritt 632 antwortet der Server 604 mit verschiedenen Nachrichten, nämlich der Server-Hello-Nachricht, dem Zertifikat, der ChangeCipherSpec-Nachricht und einer Finished-Nachricht. Im Schritt 634 antwortet der Client 502 dann mit einem Satz von Nachrichten, nämlich der ChangeCipherSpec-Nachricht und einer Finished-Nachricht. Damit ist der TLS-Folgehandshake abgeschlossen, der für eine Sitzungswiederaufnahme notwendig wäre. Im Schritt 636 erhält der Inspektor 560 die Sitzungsschlüssel aus den Sitzungsinformationen, die im Schritt 626 dem Ticket [A] entnommen wurden. Während Anwendungsdaten dann zwischen Client und Server übermittelt werden (Schritt 638), verwendet der Inspektor 600 den Sitzungsschlüssel, um Nutzlasten nach Bedarf zu entschlüsseln. Dies ist der Schritt 640.

[0056] Zusammenfassend, und wie durch einen Vergleich der Schritte 514, 516 und 524 in **Fig.** 5 mit den Schritten 616, 618 und 626 in **Fig.** 6 zu sehen ist, lässt sich sagen, dass die Methode hierin die Zwischenspeicherung des Sitzungstickets in dem Inspektor selbst überflüssig macht. Wenn der Inspektor das Sitzungsticket von dem TLS-Server erstmalig empfängt, speichert der Inspektor das Sitzungsticket nicht zwischen, sondern erzeugt vielmehr ein zusammengesetztes Ticket, welches das ursprüngliche Ticket und Sitzungskontextinformationen enthält, die den Sitzungsschlüssel enthalten, und gibt es an den Client aus. Das zusammengesetzte Ticket (bzw. das zusammengesetzte Sitzungsticket) wird vorzugsweise von dem Inspektor verschlüsselt, um die Sitzungsinformationen zu sichern. Wenn der TLS-Client das zusammengesetzte Sitzungsticket (wieder dem Inspektor) vorlegt, um die TLS-Verbindung wieder aufzunehmen, entschlüsselt der Inspektor das Ticket und ruft den Sitzungskontext direkt daraus

ab. Der Inspektor verwendet dann das ursprüngliche Sitzungsticket, um die TLS Sitzung mit dem TLS-Server wieder aufzunehmen.

[0057] Das Folgende beschreibt eine bevorzugte Technik zum Erzeugen des neuen Sitzungstickets. Dies ist der Schritt 616 in **Fig.** 6. Vorzugsweise wird das neue Sitzungsticket wie folgt erzeugt:

> Encrypt (Transform (Original Session Ticket, Session Information), Encryption Key), wobei Session Information (Sitzungsinformationen) Verschlüsselungsinformationen wie beispielsweise (eine) Cipher-Suite(s), einen Masterschlüssel usw. enthalten), wobei Transform ein beliebiges Verfahren oder eine beliebige Berechnung ist, um die ermittelten Daten, nämlich Original Session Ticket (ursprüngliches Sitzungsticket), Session Information, miteinander zu kombinieren, und wobei sich Encrypt (Verschlüsseln) auf einen beliebigen Verschlüsselungsalgorithmus bezieht, der das neue Sitzungsticket schützen kann. Eine typische Umsetzung (Transform) kann eine Verknüpfungsoperation sein, doch stellt dies keine Einschränkung dar, da komplexere Berechnungsmethoden für die Umsetzung angewendet werden können. Natürlich muss eine Umsetzung eine zugehörige Inverse haben, so dass die ursprünglichen Daten wiederhergestellt werden können. Für den Verschlüsselungswrapper könnte eine typische Lösung eine Verschlüsselung mit öffentlichem Schlüssel nutzen, so dass der Encryption Key (Verschlüsselungsschlüssel) ein öffentlicher Schlüssel mit einem zugehörigen privaten oder geheimen Schlüssel ist, der zum Entschlüsseln verwendet wird, wenn das neue Sitzungsticket von dem Client später empfangen wird. Somit, und wie vorstehend beschrieben wurde, wird das neue Sitzungsticket erzeugt, indem zuerst die Umsetzungsfunktion auf das ursprüngliche Sitzungsticket und zu der TLS-Sitzung gehörende Sitzungskontextinformationen angewendet wird, wobei dann eine Verschlüsselungsfunktion auf das Ergebnis der Umsetzung angewendet wird. Das Ergebnis der Anwendung der Umsetzung wird hierin zuweilen als ein „zusammengesetztes“ Sitzungsticket oder New Session Ticket bezeichnet. Der Inspektor gibt das verschlüsselte neue Sitzungsticket an den Client aus.

[0058] Wenn der Inspektor das verschlüsselte New Session Ticket von dem Client in der Client-Hello-Folgenachricht empfängt, wird die Sitzung durch den Inspektor wieder aufgenommen, der das Ticket entschlüsselt, um das Original Session Ticket, Session Information, zu erhalten (wiederherzustellen), und dann das .... in der Client-Hello-Nachricht durch das Original Session Ticket ersetzt. Wie vorstehend erwähnt wurde, wird die Entschlüsselung vorzugsweise unter Verwendung des privaten Schlüssels des öffentlichen Schlüsselpaares durchgeführt, das an dem Inspektor verwaltet wird. Es gibt vorzugsweise zwei Entschlüsselungsoperationen, nämlich: Get_Ticket (Decrypt (New Session Ticket) und Get_SessionInfo (Decrypt (New Session Ticket). Wenn die Entschlüsselung erfolgreich ist, sollte die entschlüsselte Nutzlast das Original Session Ticket und die Session Information enthalten. Die Get_Ticket-Funktion erhält/stellt das Original Session Ticket wieder her und Get_SessionInfo erhält die Sitzungsinformationen wie beispielsweise den Master-TLS-Schlüssel, der zum Entschlüsseln der Nutzlasten notwendig ist.

[0059] Die vorstehend beschriebene Methode bietet viele Vorteile. Aktuelle TLS-Inspektor-Mechanismen, die TLS-Session-Ticket-Unterstützung bereitstellen, haben mehrere Nachteile, die von der beschriebenen Methode überwunden werden. Vor allem besteht keine Notwendigkeit, einen Sitzungscachespeicher in dem Inspektor zu verwalten. Folglich werden die bekannten Nachteile dieser Methode (schlechte Skalierbarkeit, Verarbeitungs- und Speicherineffizienzen sowie Potenzial für Denial-of-Service-(DoS-)Ausnutzung) überwunden. Diese Methode lässt sich einfach ausführen, da die zusätzlichen Funktionen ohne Weiteres rechnerisch effizient ausgeführt werden können. Die Methode ist äußerst zuverlässig und sicher. Indem die Zwischenspeicherung der Sitzungstickets selbst ausgelagert wird, ermöglicht die Methode eine bessere Performanz des Inspektor-Mechanismus, wodurch der gesamte Betrieb des Vermittlers verbessert wird.

[0060] Die Techniken hierin vereinfachen auch die Skalierung der TLS-Unterstützungsinfrastruktur. Somit, wenn z.B. mehrere Inspektoren ausgeführt werden, kann jeder Inspektor das zusammengesetzte Sitzungsticket verstehen und die Prüfung dann wieder aufnehmen, solange die Umsetzungsfunktion über die Inspektoren hinweg vorzugsweise im Voraus synchronisiert wird (z.B., indem der zur Entschlüsselung des zusammengesetzten Tickets benötigte Schlüssel gemeinsam verwendet wird).

[0061] Zwar wurden die Techniken im Kontext eines Proxys beschrieben, doch stellt dies keine Einschränkung dar. Verallgemeinernd gesagt kann die hierin beschriebene Verarbeitung in einem beliebigen, zwischen Client und Server angeordneten Vermittler durchgeführt werden. In einer einzelnen solchen Ausführungsform ermöglicht der Vermittler eine transparente integrierte Inhaltsprüfung und -änderung. Der Client und der Server sind Datenverarbeitungsentitäten (Endpunkte). Der Vermittler kann als eine physische Einheit, eine virtuelle Einheit oder als eine Kombination daraus konfiguriert sein. Er kann für mehrere verschiedene Anwendungen verwendet werden, darunter, ohne darauf beschränkt

zu sein, die Entschlüsselung von verschlüsselten (SSL/TLS-)Sitzungen, so dass eine Sicherheitsprüfung in der zuvor beschriebenen Weise durchgeführt werden kann.

[0062] Während eine bevorzugte Betriebsumgebung und ein bevorzugter Anwendungsfall (eine sichere Appliance) beschrieben wurden, können die Techniken hierin in jeder beliebigen anderen Betriebsumgebung verwendet werden, in der Netzwerkverkehr an ein(e) und/oder von einem/einer Datenverarbeitungssystem oder Datenverarbeitungseinheit auf Wunsch abgefangen, entschlüsselt, geprüft und/oder geändert (umgeschrieben) werden soll.

[0063] Wie beschrieben wurde, kann die vorstehend beschriebene Funktionalität als eine eigenständige Methode, z.B. eine durch einen Prozessor ausgeführte softwarebasierte Funktion ausgeführt sein oder sie kann als ein Service (darunter ein Web-Service über eine SOAP/XML-Schnittstelle) zur Verfügung stehen. Die hierin beschriebenen jeweiligen Einzelheiten der Hardware- und Software-Ausführung dienen lediglich dem Zweck der Veranschaulichung und sind nicht dazu gedacht, den Umfang des beschriebenen Gegenstands einzuschränken.

[0064] Allgemeiner ausgedrückt, bei jeder der Datenverarbeitungseinheiten im Kontext des offenbarten Gegenstands handelt es sich um ein Datenverarbeitungssystem (wie es beispielsweise in **Fig.** 2 gezeigt ist), das Hardware und Software aufweist, und diese Entitäten tauschen miteinander über ein Netzwerk wie beispielsweise das Internet, ein Intranet, ein Extranet, ein privates Netzwerk oder ein(e) beliebige(s) andere(s) Datenübertragungsmedium oder -verbindung Daten aus. Die Anwendungen auf dem Datenverarbeitungssystem stellen native Unterstützung für Web- und andere bekannte Services und Protokolle bereit, darunter, ohne darauf beschränkt zu sein, Unterstützung für HTTP, FTP, SMTP, SOAP, XML, WSDL, UDDI sowie WSFL u.a. Informationen zu SOAP, WSDL, UDDI und WSFL werden vom World Wide Web Consortium (W3C) zur Verfügung gestellt, das für die Entwicklung und die Pflege dieser Standards zuständig ist; weitere Informationen zu HTTP, FTP, SMTP und XML werden von der Internet Engineering Task Force (IETF) zur Verfügung gestellt. Vertrautheit mit diesen bekannten Standards und Protokollen wird vorausgesetzt.

[0065] Die hierin beschriebenen Techniken können in oder in Verbindung mit verschiedenen clientseitigen Architekturen (z.B. Firewalls, NAT-Einheiten) und in oder in Verbindung mit verschiedenen serverseitigen Architekturen, darunter einfache n-schichtige Architekturen, Webportale, föderierte Systeme und dergleichen, ausgeführt werden. Die Techniken hierin können in einem lose verbundenen Server (darunter einer „cloudbasierten“) Umgebung in die Praxis umgesetzt werden.

[0066] Noch allgemeiner ausgedrückt, kann der hierin beschriebene Gegenstand die Form einer reinen Hardware-Ausführung, einer reinen Software-Ausführung oder einer Ausführung annehmen, die sowohl Hardware- als auch Software-Elemente enthält. In einer bevorzugten Ausführungsform ist die vertrauenswürdige Plattformmodul-Funktion in Software ausgeführt, die, ohne darauf beschränkt zu sein, Firmware, residente Software, Mikrocode und dergleichen beinhaltet. Darüber hinaus können die Download- und Lösch-Schnittstellen und -Funktionalität die Form eines Computerprogrammprodukts annehmen, auf das von einem durch einen Computer verwendbaren oder durch einen Computer lesbaren Datenträger zugegriffen werden kann, welcher Programmcode zur Verwendung durch einen Computer oder ein beliebiges Anweisungsausführungssystem oder in Verbindung mit einem Computer oder einem beliebigen Anweisungsausführungssystem bereitstellt. Zum Zweck dieser Beschreibung kann ein durch einen Computer verwendbarer oder durch einen Computer lesbarer Datenträger eine beliebige Vorrichtung sein, die das Programm zur Verwendung durch das/die oder in Verbindung mit dem/der Anweisungsausführungssystem, -vorrichtung oder -einheit enthalten oder speichern kann. Bei dem Datenträger kann es sich um ein(e) elektronische(s), magnetische(s), optische(s), elektromagnetische(s), Infrarot- oder Halbleitersystem (oder -vorrichtung oder -einheit) handeln. Zu Beispielen für einen durch einen Computer lesbaren Datenträger gehören ein Halbleiter- oder Solid-State-Speicher, ein Magnetband, eine auswechselbare Computerdiskette, ein Direktzugriffspeicher (RAM), ein Nur-Lese-Speicher (ROM), eine magnetische Festplatte und eine optische Platte. Zu aktuellen Beispielen für optische Platten gehören Compact-Disk-Nur-Lese-Speicher (CD-ROM), Compact-Disk-R/W (CD-R/W) und DVD. Der durch einen Computer lesbare Datenträger ist ein physisch greifbarer, nicht flüchtiger Gegenstand.

[0067] Bei dem Computerprogrammprodukt kann es sich um ein Produkt handeln, das über Programmanweisungen (oder Programmcode) verfügt, um eine oder mehrere der beschriebenen Funktionen auszuführen. Diese Anweisungen bzw. dieser Code können/kann auf einem nicht flüchtigen, durch einen Computer lesbaren Speichermedium in einem Datenverarbeitungssystem gespeichert werden, nachdem sie/er über ein Netzwerk von einem fernen Datenverarbeitungssystem heruntergeladen wurde (n). Oder diese Anweisungen bzw. dieser Code können/kann auf einem durch einen Computer lesbaren Speichermedium in einem Server-Datenverarbeitungssystem gespeichert und so angepasst werden, dass sie/er über ein Netzwerk auf ein fernes Daten-

verarbeitungssystem zur Verwendung in einem durch einen Computer lesbaren Speichermedium in dem fernen System heruntergeladen werden können/kann.

[0068] In einer repräsentativen Ausführungsform sind die Schnittstellen und das Dienstprogramm in einer Spezialdatenverarbeitungsplattform ausgeführt, vorzugsweise in Software, die durch einen oder mehrere Prozessoren ausgeführt wird. Die Software wird in einem oder mehreren Datenspeichern oder Hauptspeichern, die zu dem einen oder den mehreren Prozessoren gehören, verwaltet und die Software kann als ein oder mehrere Computerprogramme ausgeführt sein. Gemeinsam weist diese Spezialhardware und - software die vorstehend beschriebene Funktionalität auf.

[0069] Während das Vorstehende eine bestimmte Reihenfolge von Operationen beschreibt, die von bestimmten Ausführungsformen der Erfindung durchgeführt werden, sollte klar sein, dass eine solche Reihenfolge beispielhaft ist, da alternative Ausführungsformen die Operationen in einer anderen Reihenfolge durchführen, bestimmte Operationen kombinieren, bestimmte Operationen überlappen können oder dergleichen. Verweise in der Spezifikation auf eine bestimmte Ausführungsform zeigen an, dass die beschriebene Ausführungsform ein(e) bestimmte(s) Merkmal, Struktur oder Eigenschaft enthalten kann, jede Ausführungsform das/die bestimmte Merkmal, Struktur oder Eigenschaft gegebenenfalls aber nicht unbedingt enthält.

[0070] Schließlich, während bestimmte Komponenten des Systems getrennt beschrieben wurden, versteht der Fachmann, dass einige der Funktionen in bestimmten Anweisungen, Programmabfolgen, Codeteilen und dergleichen kombiniert oder gemeinsam genutzt werden können.

[0071] Die Appliance ist nicht auf irgendeinen bestimmten Typ beschränkt. Die vorstehend beschriebene Operation kann ebenso zusammen mit einer/einem beliebigen bekannten Technik oder Mechanismus verwendet werden, die/der selbst verwendet wird, um Daten von einer beliebigen Maschine, ungeachtet der physischen Konfiguration der Maschine, abzufangen, zu entschlüsseln, zu prüfen, zu ändern, umzuschreiben und erneut zu verschlüsseln.

[0072] Die Techniken hierin ermöglichen allgemein die vorstehend beschriebenen Verbesserungen an einer Technologie oder einem technischen Gebiet sowie die spezifischen technologischen Verbesserungen an mit einem Netzwerk verbundenen sicheren Appliances, wie sie vorstehend beschrieben wurden.

[0073] Die Vorstellung, einen TLS-Inspektor ohne einen Sitzungscachespeicher (die „cachespeicherlose“ Art der Technik) bereitzustellen, setzt nicht unbedingt voraus, dass der TLS-Inspektor physisch ohne einen solchen Cachespeicher konfiguriert ist, obgleich dies die übliche Konfiguration ist. Ein TLS-Inspektor, der über einen Sitzungscachespeicher verfügt, der nicht genutzt oder in anderer Weise umgangen wird, fällt unter den Umfang des offenbarten Gegenstands, der nachstehend beansprucht wird. Somit, in der Verwendung hierin, bedeutet „cachespeicherlos“, dass ein Sitzungscachespeicher entweder weggelassen oder, sofern vorhanden, nicht genutzt wird.

Patentansprüche

1. Verfahren, das innerhalb eines zwischen einem Transport Layer Security-(TLS-)Client (602) und einem TLS-Server (604) angeordneten Vermittlers (600) wirksam ist und das eine TLS-Prüffunktion während einer TLS-Sitzung bereitstellt, wobei das Verfahren aufweist:
nach dem Empfang (614) eines ursprünglichen Sitzungstickets von dem TLS-Server (604) Erzeugen (616) eines neuen Sitzungstickets, wobei das neue Sitzungsticket ein Wert ist, der abgeleitet wird, indem eine Umsetzungsfunktion auf das ursprüngliche Sitzungsticket und Sitzungskontextinformationen angewendet wird, wobei die Sitzungskontextinformationen eine Cipher-Suite und einen Masterschlüssel beinhalten, die zur Verwendung während der TLS-Sitzung ausgehandelt werden;
Anwenden einer Verschlüsselungsfunktion auf das neue Sitzungsticket;
Ausgeben (618) des verschlüsselten neuen Sitzungstickets an den TLS-Client (602), wodurch die Notwendigkeit, die Sitzungskontextinformationen zu verwalten, von dem Vermittler (600) auf den TLS-Client (602) übertragen wird;
nach dem Empfang (624) des verschlüsselten neuen Sitzungstickets von dem TLS-Client (602) Entschlüsseln (626) des verschlüsselten neuen Sitzungstickets, um das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen einschließlich des Masterschlüssels wiederherzustellen; und
Verwenden des wiederhergestellten ursprünglichen Sitzungstickets und der Sitzungskontextinformationen, um die TLS-Sitzung wieder aufzunehmen.

2. Verfahren nach Anspruch 1, wobei die Umsetzungsfunktion das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen verknüpft.

3. Verfahren nach einem der vorhergehenden Ansprüche, wobei das verschlüsselte neue Sitzungsticket von dem TLS-Client (602) in einer Client-Hello-Nachricht empfangen (624) wird.

4. Verfahren nach Anspruch 3, wobei die TLS-Sitzung zumindest teilweise wieder aufgenommen wird, indem das verschlüsselte neue Sitzungsticket in der Client-Hello-Nachricht durch das ursprüngliche Sitzungsticket ersetzt (628) und die Client-Hello-Nachricht an den TLS-Server (604) weitergeleitet (630) wird.

5. Verfahren nach einem der vorhergehenden Ansprüche, wobei das Verwenden der Sitzungskontextinformationen das Erhalten eines Sitzungsschlüssels beinhaltet.

6. Verfahren nach Anspruch 5, das des Weiteren das Verwenden (640) des Sitzungsschlüssels zum Entschlüsseln einer Nutzlast des ursprünglichen Sitzungstickets beinhaltet.

7. Verfahren nach einem der vorhergehenden Ansprüche, wobei das neue Sitzungsticket erzeugt (616) und das verschlüsselte neue Sitzungsticket ausgegeben (618) wird, anstatt das ursprüngliche Sitzungsticket in einem Sitzungscachespeicher zwischenzuspeichern.

8. Vorrichtung (600), die zwischen einem Transport Layer Security-(TLS-)Client (602) und einem TLS-Server (604) angeordnet ist und eine TLS-Prüffunktion während einer TLS-Sitzung bereitstellt, die aufweist:
einen Prozessor (204);
Computerspeicher (206, 208), der durch den Prozessor (204) ausgeführte Computerprogrammanweisungen enthält, wobei die Computerprogrammanweisungen Programmcode aufweisen, der so konfiguriert ist, dass er:
nach dem Empfang (614) eines ursprünglichen Sitzungstickets von dem TLS-Server (604) ein neues Sitzungsticket erzeugt (616), wobei das neue Sitzungsticket ein Wert ist, der abgeleitet wird, indem eine Umsetzungsfunktion auf das ursprüngliche Sitzungsticket und Sitzungskontextinformationen angewendet wird, wobei die Sitzungskontextinformationen eine Cipher-Suite und einen Masterschlüssel beinhalten, die zur Verwendung während der TLS-Sitzung ausgehandelt werden;
eine Verschlüsselungsfunktion auf das neue Sitzungsticket anwendet;
das verschlüsselte neue Sitzungstickets an den TLS-Client (602) ausgibt (618), wodurch die Notwendigkeit, die Sitzungskontextinformationen zu verwalten, von dem Vermittler (600) auf den TLS-Client (602) übertragen wird;
nach dem Empfang (624) des verschlüsselten neuen Sitzungstickets von dem TLS-Client (602) das verschlüsselte neue Sitzungsticket entschlüsselt (626), um das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen einschließlich des Masterschlüssels wiederherzustellen; und
das wiederhergestellte ursprüngliche Sitzungsticket und die Sitzungskontextinformationen verwendet, um die TLS-Sitzung wieder aufzunehmen.

9. Vorrichtung (600) nach Anspruch 8, wobei die Umsetzungsfunktion das ursprüngliche Sitzungsticket und die Sitzungskontextinformationen verknüpft.

10. Vorrichtung (600) nach einem der Ansprüche 8 bis 9, wobei das verschlüsselte neue Sitzungsticket von dem TLS-Client (602) in einer Client-Hello-Nachricht empfangen (624) wird.

11. Vorrichtung (600) nach Anspruch 10, wobei die TLS-Sitzung zumindest teilweise wieder aufgenommen wird, indem das verschlüsselte neue Sitzungsticket in der Client-Hello-Nachricht durch das ursprüngliche Sitzungsticket ersetzt (628) und die Client-Hello-Nachricht an den TLS-Server (604) weitergeleitet (630) wird.

12. Vorrichtung (600) nach einem der Ansprüche 8 bis 11, wobei das Verwenden der Sitzungskontextinformationen das Erhalten eines Sitzungsschlüssels beinhaltet.

13. Vorrichtung (600) nach Anspruch 12, die des Weiteren das Verwenden (640) des Sitzungsschlüssels zum Entschlüsseln einer Nutzlast des ursprünglichen Sitzungstickets beinhaltet.

14. Vorrichtung (600) nach einem der Ansprüche 8 bis 13, wobei das neue Sitzungsticket erzeugt (616) und das verschlüsselte neue Sitzungsticket ausgegeben (618) wird, anstatt das ursprüngliche Sitzungsticket in einem Sitzungscachespeicher zwischenzuspeichern.

15. Computerprogrammprodukt (220) für eine Transport Layer Security-(TLS-)Prüffunktion während einer TLS-Sitzung, wobei das Computerprogrammprodukt aufweist:
ein durch einen Computer lesbares Speichermedium (218), das durch eine Verarbeitungsschaltung lesbar ist und Anweisungen (216) zur Ausführung durch die Verarbeitungsschaltung speichert, um ein Verfahren nach einem der Ansprüche 1 bis 7 durchzuführen.

16. Computerprogramm, das auf einem durch einen Computer (200) lesbaren Datenträger (218) gespeichert ist und in den internen Hauptspeicher (206) eines digitalen Computers (200) ladbar ist, wobei das Computerprogramm Teile von Software-Code (216) aufweist, wenn das Programm auf einem Computer (200) ausgeführt wird, um das Verfahren nach einem der Ansprüche 1 bis 7 durchzuführen.

Es folgen 6 Seiten Zeichnungen

Anhängende Zeichnungen

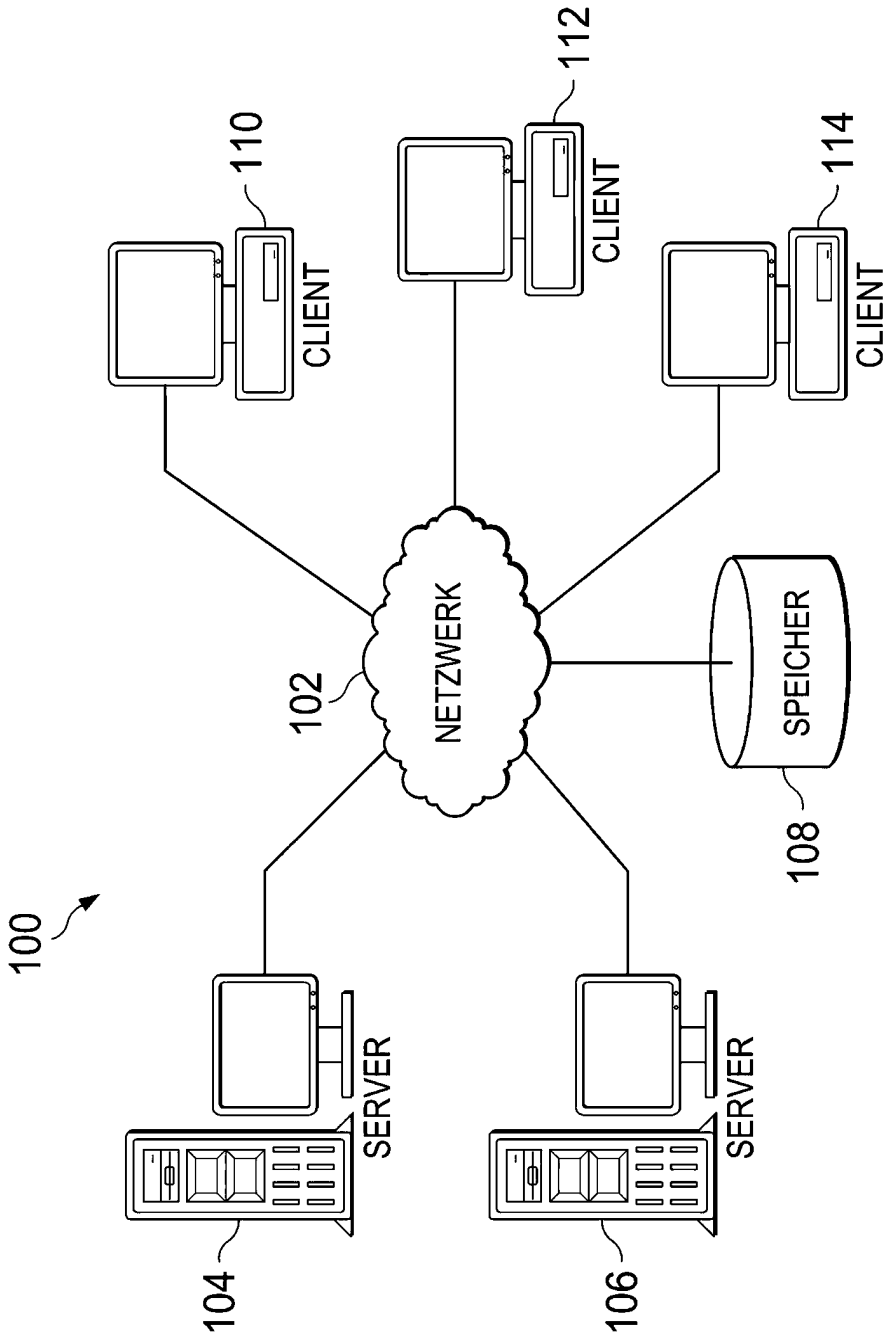


FIG. 1

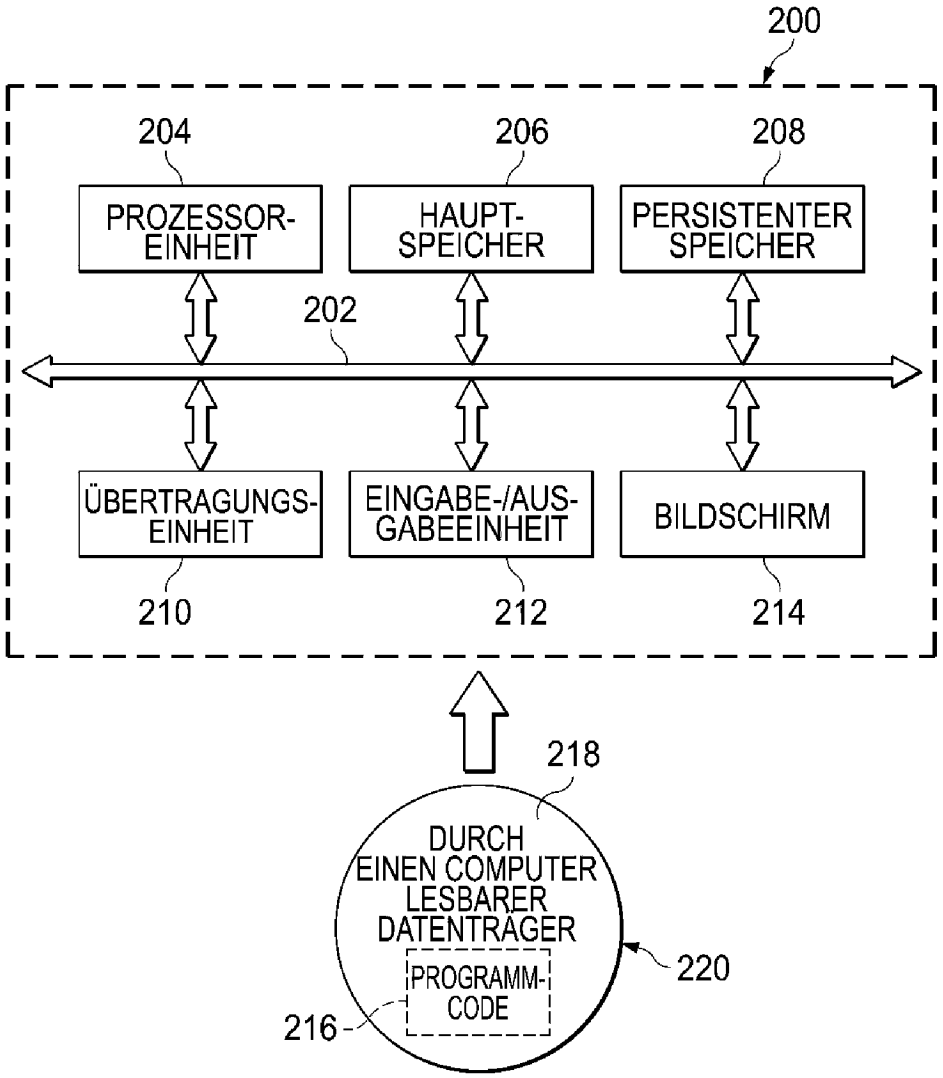

200
204
206
208
PROZESSOR-EINHEIT
HAUPT-SPEICHER
PERSISTENTER SPEICHER
202
ÜBERTRAGUNGS-EINHEIT
EINGABE-/AUS-GABEEINHEIT
BILDSCHIRM
210
212
214
218
DURCH EINEN COMPUTER LESBARER DATENTRÄGER
PROGRAMM-CODE
220
216

FIG. 2

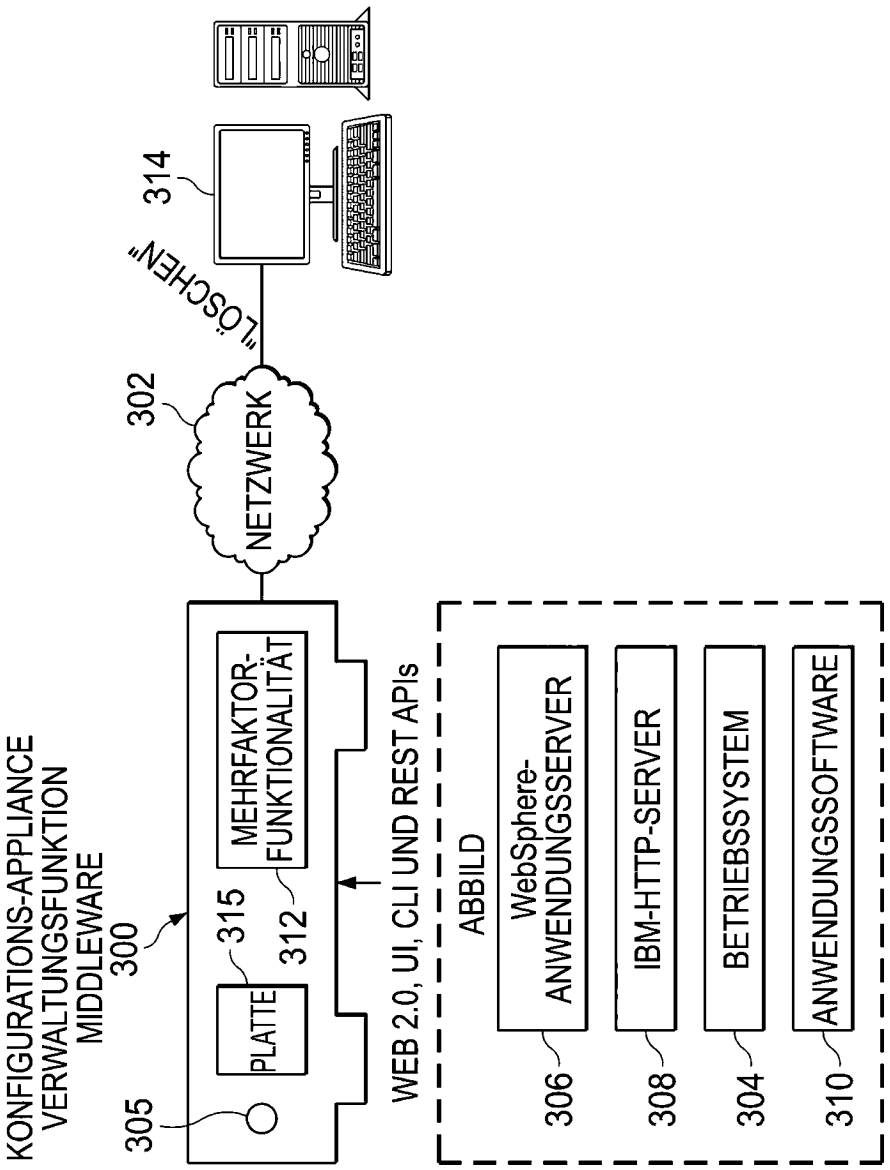

HARDWARE-KONFIGURATIONS-APPLIANCE
VERWALTUNGSFUNKTION
MIDDLEWARE
300
305
PLATTE
315
MEHRFAKTOR-FUNKTIONALITÄT
312
WEB 2.0, UI, CLI UND REST APIs
302
NETZWERK
„LÖSCHEN"
314
ABBILD
WebSphere-ANWENDUNGSSERVER
306
IBM-HTTP-SERVER
308
BETRIEBSSYSTEM
304
ANWENDUNGSSOFTWARE
310

FIG. 3

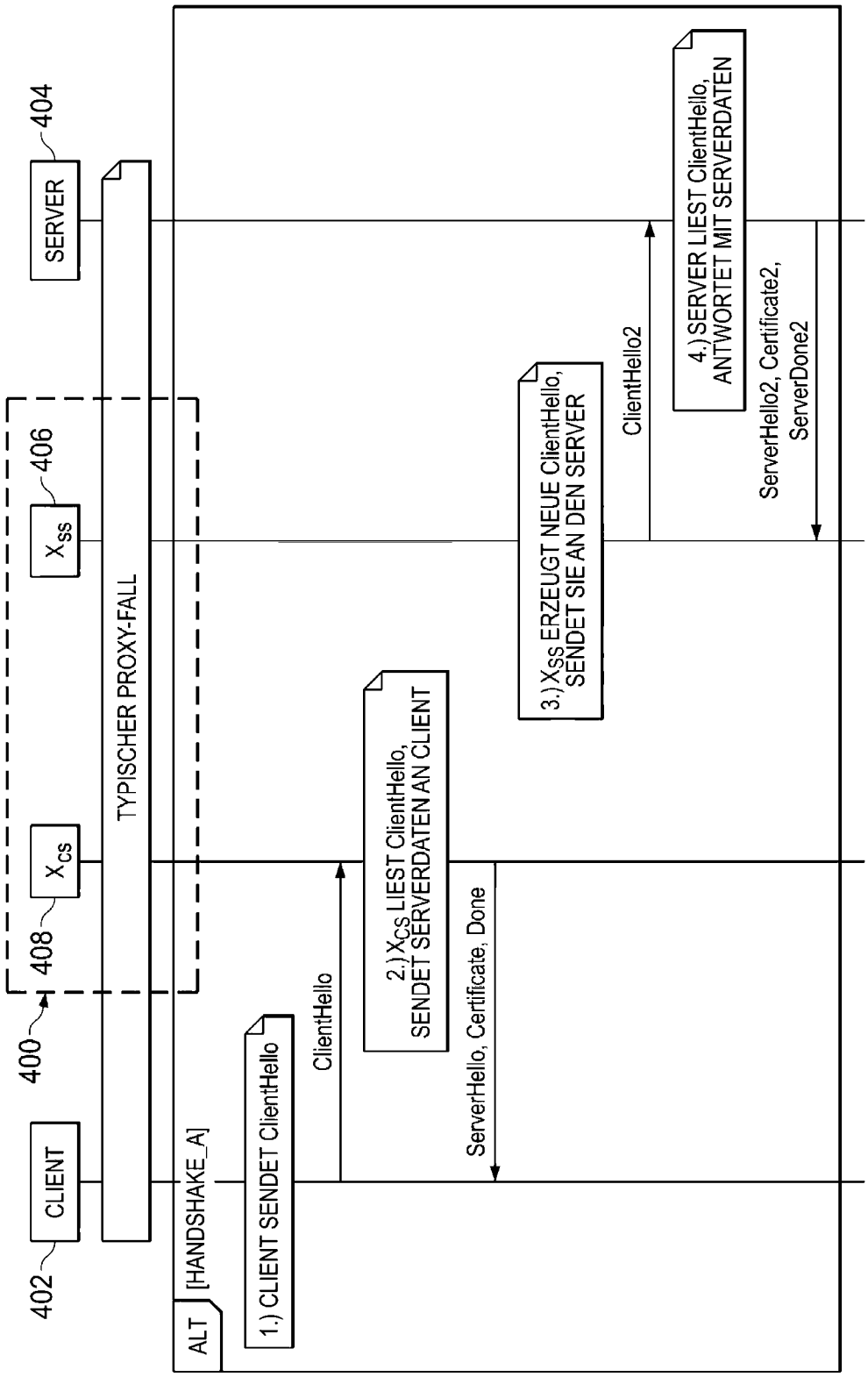


FIG. 4
(Stand der Technik)

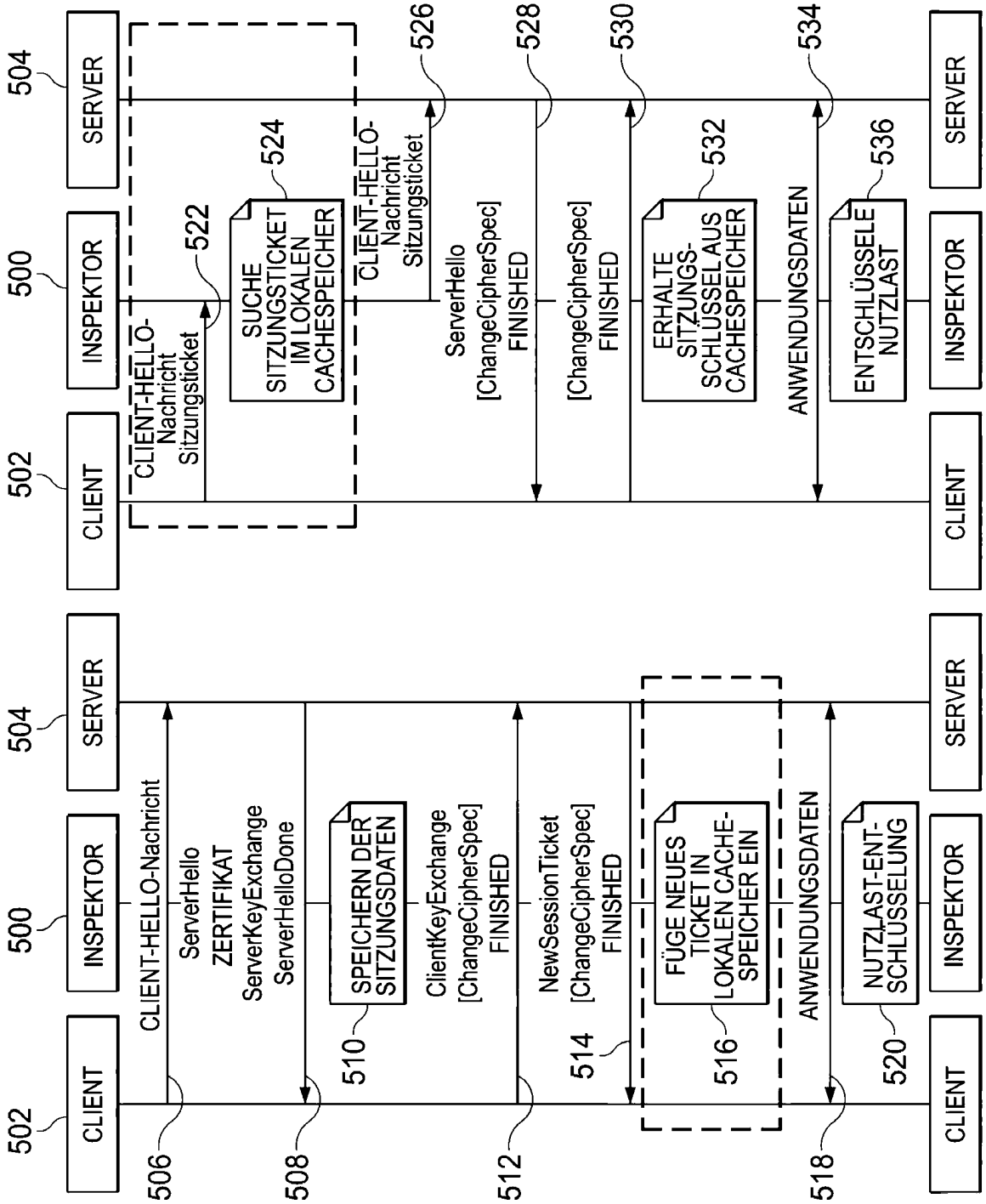


FIG. 5
(Stand der Technik)

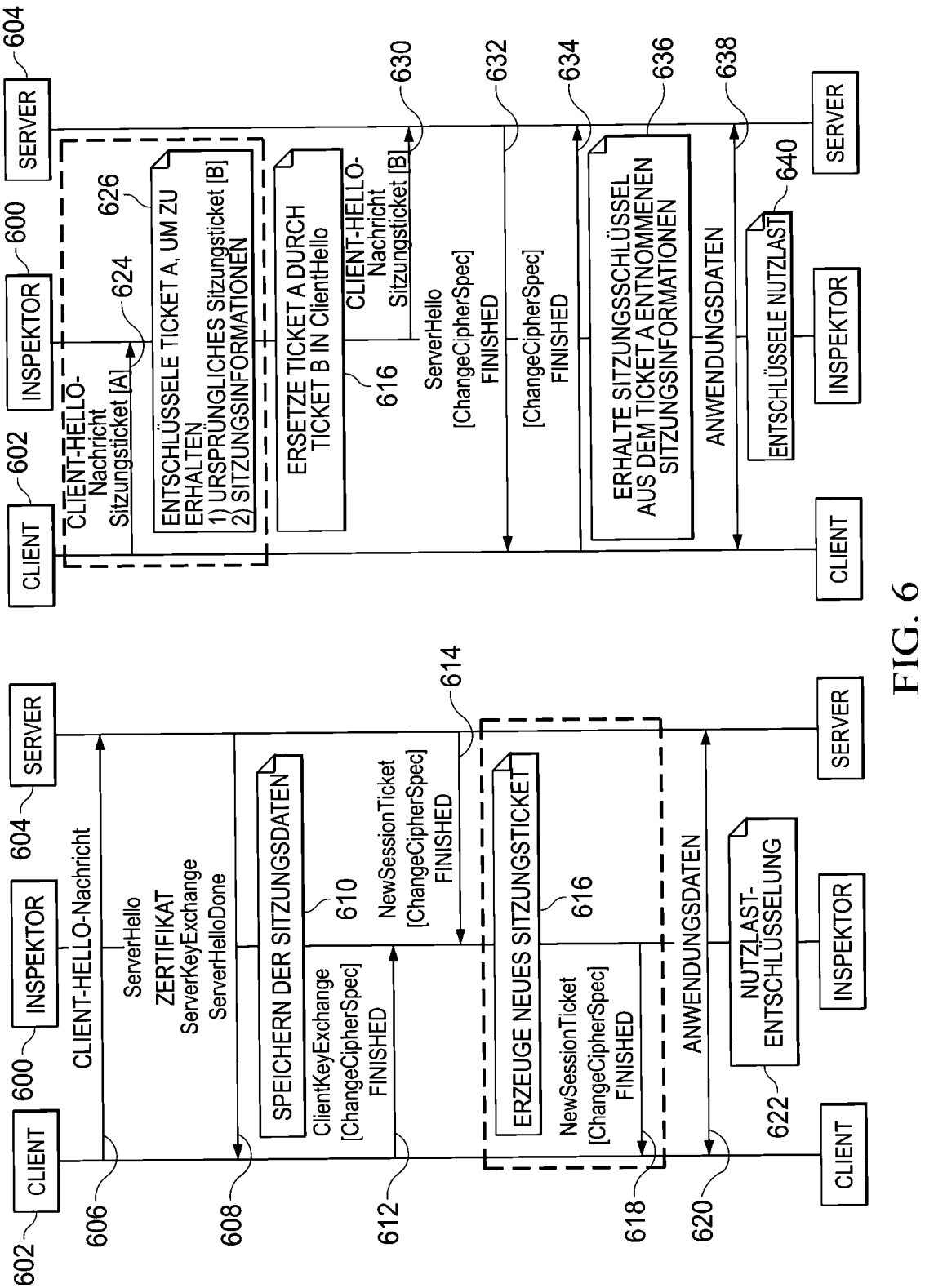


FIG. 6

