Mechanismen für Vertrauenswürdigkeit und Sicherheit in Cloud Computing Infrastrukturen

Florian Franzmann
siflfran@hawo.stw.uni-erlangen.de

ABSTRACT

Viele Firmen sind in den letzten Jahren dazu übergegangen, aufwändige Berechnungen durch externe Unternehmen in öffentlichen Clouds ausführen zu lassen. Dies birgt viele Vorteile für diese Firmen, das Risiko, das durch die Speicherung und Verarbeitung von sensiblen Daten entsteht, darf jedoch nicht unterschätzt werden. Es wäre fatal, wenn Geschäftsgeheimnisse Mitbewerbern bekannt würden.

Deutsches Recht untersagt prinzipiell die Erhebung, Speicherung und Verarbeitung personenbezogener Daten, sofern nicht eine rechtliche Grundlage dafür besteht oder der Betroffene ausdrücklich zugestimmt hat. Die Datenerhebung muss sich am Prinzip der Datensparsamkeit orientieren und darf nur zweckgebunden erfolgen. Es ist nicht zulässig, das Gesetz dadurch zu unterlaufen, dass personenbezogene Daten in Länder mit weniger strengem Datenschutzrecht übertragen werden.

Die aufgeführten rechtlichen Rahmenbedingungen stellen Nutzer von Cloud-Diensten vor eine Herausforderung. Prinzipiell sind zwei Ansätze zu unterscheiden, mit denen versucht wird, dem Datenschutzrecht in der Cloud zu genügen. Theoretisch orientierte Ansätze versuchen, Berechnungen homomorph auf verschlüsselten Daten auszuführen, so dass der Cloud-Betreiber keine Möglichkeit hat, Einsicht in die Daten zu nehmen. Dieses Konzept der *Vertraulichkeits-Homomorphismen* befindet sich jedoch noch in einem frühen Stadium der Erforschung.

Deswegen versuchen praktisch orientierte Ansätze Vertrauen in das Rechensystem mit Hilfe des *Trusted Platform Modules (TPM)* herzustellen.

Diese Arbeit analysiert mögliche Angriffsvektoren in der Cloud und geht unter Analyse ihrer Stärken und Schwächen auf die beiden Ansätze zur Wahrung der Privatsphäre ein.

1. MOTIVATION

Firmen gehen dazu über, Rechenleistung in Form von *Infrastruktur als Dienstleistung* in Anspruch zu nehmen. Bei dieser Art von Dienstleistung vermietet ein Anbieter den Zugang zu virtuellen Maschinen, die auf der physikalischen Hardware des Anbieters laufen. Der Kunde betreibt seine Software in diesen virtuellen Maschinen. Es ist ihm möglich, jederzeit mehr Rechenleistung vom Anbieter zu erwerben, die ihm zeitnah zur Verfügung gestellt wird. Der hierbei verwendete Ansatz, Virtualisierung, preisgünstige Hardware „von der Stange“, verteiltes Rechnen und Abrechnung nach tatsächlich erfolgtem Betriebsmittelverbrauch zu kombinieren, wird unter dem Schlagwort *Cloud Computing* zusammengefasst.

Der Kunde hat in der Regel ein Interesse daran, dass die Daten, auf denen er in der Cloud Berechnungen ausführt, vertraulich behandelt werden. Häufig handelt es sich bei diesen Daten um Firmengeheimnisse, die auf keinen Fall Mitbewerbern bekannt werden dürfen. Noch kritischer sind personenbezogene Daten, deren Preisgabe einerseits rufschädigend für die Firma wirken würde, andererseits existieren in vielen Ländern Gesetze, die genau regeln, welche Daten öffentlich sein dürfen und welche nicht.

In den Staaten der Europäischen Union wird dies durch Richtlinie 95/46/EG geregelt [1], die für alle Mitgliedsstaaten bindend ist. Die Bundesrepublik Deutschland setzt diese durch das, in der Europäischen Union in Bezug auf den Datenschutz strengste Gesetz [2], das *Bundesdatenschutzgesetz*, um [3]. Dieses Gesetz untersagt prinzipiell die Erhebung, Verarbeitung und Nutzung personenbezogener Daten. Nur dann, wenn entweder die Nutzung der Daten durch ein Gesetz erlaubt wird oder der Betroffene (in der Regel schriftlich) der Nutzung seiner Daten zugestimmt hat, ist die Erhebung und Nutzung personenbezogener Daten zulässig. Dabei muss derjenige, der die Daten erhebt, sich am Prinzip der *Datensparsamkeit* orientieren, d. h. es dürfen nur die für den Verwendungszweck zwingend notwendigen Daten erhoben werden. Die Erlaubnis ist dabei auf den ursprünglichen Verwendungszweck, dem der Betroffene zugestimmt hat, beschränkt. Ein Transfer personenbezogener Daten in ein Land mit unzureichender Datenschutzgesetzgebung ist nicht zulässig.

Im Folgenden werden zunächst die möglichen Angriffsvektoren, die bei der Verarbeitung sensibler Daten in der Cloud berücksichtigt werden müssen, diskutiert. Anschliessend wird analysiert, welche Möglichkeiten zur Unterbindung erfolgreicher Angriffe im Rahmen der Cloud zur Verfügung stehen. Eine Analyse der bisher verwirklichten Ansätze zur Wahrung der Privatsphäre in der Cloud schließen diese Arbeit ab.

2. ANGRIFFSVEKTOREN

Grundsätzlich sind in einer Cloud drei Angriffsvektoren zu unterscheiden. Erstens Angriffe von innen durch Mitarbeiter des Cloud-Anbieters, zweitens von aussen durch Mitbewerber und drittens durch staatliche Behörden, wie Polizeiorganisationen und Geheimdienste. Dieser Abschnitt stellt die drei Angriffsvektoren vor und schlägt Ansätze vor, sie zu unterbinden.

2.1 Angriffe durch Mitarbeiter

Es ist möglich, dass Mitarbeiter des Cloud-Anbieters administrative Privilegien missbrauchen und sich Zugang zu den Daten, die der Kunde in der Cloud speichert, verschaffen. Systemadministratoren haben in der Regel als Benutzer `root` Zugang zu den physikalischen Rechnern einer Cloud. Sie können beliebige Software installieren und ausführen. Außerdem sind sie in der Lage, zur Laufzeit auf den Arbeitsspeicher virtueller Maschinen zuzugreifen. Hat ein Systemadministrator Zugang zur Hardware der Cloud, so ist es ihm möglich diese zu manipulieren oder Kaltstartangriffe durchzuführen.

Ristenpart et al. schlagen mehrere Maßnahmen [4] gegen Missbrauch durch Mitarbeiter des Cloud-Anbieters vor. So soll der Cloud-Anbieter Sorge tragen, dass keine einzelne Person genug Privilegien anhäuft, um im Alleingang missbräuchlich auf das System zuzugreifen. Rechnerräume sollen durch Sicherheitsgeräte und Kameras überwacht werden, so dass Manipulationen an der Hardware aufgedeckt werden können.

Dadurch sind physikalische Manipulationen weitgehend ausgeschlossen – es besteht aber immer noch die Möglichkeit, dass Systemadministratoren sich über das Netzwerk auf den Rechnern, auf denen die virtuellen Maschinen laufen, einloggen. Es wird deswegen gefordert, dass Administratoren auf Systemen, die sich im Produktivbetrieb befinden, keinen `root`-Zugang erhalten. Virtuelle Maschinen, an denen Wartungsarbeiten durchgeführt werden sollen, müssten dann erst auf ein System migrieren, auf das ein Administrator uneingeschränkten Zugriff hat.

2.2 Angriffe durch Mitbewerber

Der zweite zu beachtende Angriffsvektor kommt durch andere Benutzer der selben Cloud zustande. Prinzipbedingt kommt es in einer Cloud vor, dass die virtuellen Maschinen unterschiedlicher Cloud-Kunden auf der selben physikalischen Hardware laufen. Hierbei besteht die Gefahr, dass ein Angreifer in der Lage ist, die Isolierung zwischen seiner eigenen virtuellen Maschine und der des Opfers zu durchbrechen.

Dies kann entweder durch einen Ausbruch aus dem Hypervisor oder durch einen Side-Channel-Angriff, d. h. durch Beobachtung von Phänomenen, die aufgrund der Eigenschaften der physikalischen Hardware zustandekommen, geschehen.

Ein Angriff durch einen Mitbenutzer einer Cloud erfolgt in zwei Schritten. Zunächst muss der Angreifer dafür sorgen, dass seine eigene virtuelle Maschine auf der selben physikalischen Hardware läuft, wie die seines Opfers. Ristenpart et al. haben die Wahrscheinlichkeit dafür, dass dies unter Aufwendung weniger Dollars in Amazons EC2-Cloud gelingt, mit 40 % bestimmt und außerdem einen effizienten Test entwickelt, der in Erfahrung bringt, ob die Platzierung der virtuellen Maschine gelungen ist [4].

Der zweite Schritt des Angriffs besteht in der Extraktion vertraulicher Daten. Ristenpart et al. nennen als Beispiel eines möglichen Angriffs die Messung der Prozessorlast anderer virtueller Maschinen über die Speicherzugriffszeit der eigenen Maschine. Aus der Prozessorlast ist es dann möglich auf die Netzwerkdatenrate zu schließen oder zu erkennen, wann ein Benutzer per SSH welche Tastendrücke an eine virtuelle Maschine schickt. Hierbei waren die Autoren so erfolgreich, dass sie es für möglich halten, auf Passwörter, die ein Benutzer eingibt, zu schließen.

Ristenpart et al. halten den exklusiven Zugang zur physikalischen Hardware für die einzig wirksame Maßnahme gegen Angriffe durch Mitbenutzer desselben Rechensystems.

2.3 Angriffe durch den Staat

Der dritte Angriffsvektor kommt dadurch zustande, dass sich der Staat in vielen Ländern per Gesetz die Einsichtnahme in vertrauliche Daten vorbehält, wodurch ein Cloud-Betreiber gezwungen ist, Daten offenzulegen, sofern er dazu in der Lage ist. Dies dient Staaten einerseits zur Durchführung von Industriespionage, andererseits könnte es geschehen, dass der Cloud-Betreiber für Straftaten, die seine Kunden begangen haben, zur Rechenschaft gezogen wird.

3. THEORETISCHE ANSÄTZE

Im Kontext des Cloud Computings erscheinen gewöhnliche symmetrische oder asymmetrische Verschlüsselungsalgorithmen ungeeignet, um Daten vor Einsichtnahme und Manipulation zu schützen. Dieser Abschnitt stellt theoretische Ansätze vor, die es ermöglichen, Berechnungen auf verschlüsselten Daten auszuführen.

3.1 Gewöhnliche Verschlüsselung

Traditionelle kryptographische Algorithmen wie DES [5] oder RSA [6] wirken darauf hin, dass Daten geschützt vor Einsichtnahme und Manipulation gespeichert und übertragen werden können. Hierbei ist es nicht möglich (abgesehen von der Entschlüsselung), sinnvolle Operationen auf den Daten auszuführen. Bei Berechnungen in einer Cloud sollen Daten aber nicht nur aufbewahrt werden. Schließlich erwirbt der Kunde auch Rechenleistung vom Cloud-Anbieter, die er dazu nutzen möchte, Berechnungen auszuführen, für die seine eigene Hardware ungeeignet ist.

Rivest ging davon aus, dass es prinzipiell zwei Möglichkeiten [7] gibt, das Problem, Berechnungen auf verschlüsselten Daten auszuführen, zu lösen.

3.2 Kryptographische Hardware

Der erste Weg, den Rivest vorschlägt ist, die Hardware um einen physikalisch gesicherten Registersatz und eine zweite arithmetisch-logische Einheit zu erweitern. Im Hauptspeicher des Rechners befinden sich verschlüsselte Daten, die, bevor sie in den sicheren Registersatz geladen werden, durch eine Hardwareeinheit entschlüsselt werden.

Die Anweisungen, welche die Daten zurück in den Hauptspeicher schreiben, sind dafür zuständig, diese vor dem Schreibvorgang wieder zu verschlüsseln. Rivest zeigt zwar, dass dieser Ansatz funktioniert und sicher ist, geht aber davon aus, dass derart modifizierte Hardware grundsätzlich teurer ist als unmodifizierte Hardware.

3.3 Vertraulichkeits-Homomorphismen

Rivest sagte 1978 die Existenz von Funktionen voraus, die er als *Vertraulichkeits-Homomorphismen (privacy homomorphisms)* [7] bezeichnete. Diese sollten es ermöglichen, Berechnungen auf vertraulichen Daten auszuführen, ohne dass die Institution, die diese Berechnungen ausführt, Kenntnis der Daten oder der unverschlüsselten Ergebnisse der Berechnungen erhält.

3.3.1 Definition

Ein solches vollständig homomorphes Verschlüsselungsschema ist ein algebraisches System, das aus einer Menge S, Operationen f_1, f_2, ..., Prädikaten p_1, p_2, ... und Konstanten s_1, s_2, ... besteht. Das System wird dargestellt als

$$\langle S; f_1, \ldots, f_k; p_1, \ldots, p_l; s_1, \ldots, s_m \rangle \tag{1}$$

Ein Beispiel eines solchen Systems ist die Menge der ganzen Zahlen $\mathbb{Z}$, die Operationen Addition, Subtraktion, Multiplikation und Division, das Prädikat $\leq$ und die Konstanten 0 und 1, das ausgedrückt wird durch

$$\langle \mathbb{Z}; +, -, \times, \div; \leq; 0, 1 \rangle \tag{2}$$

Zusätzlich zu dem algebraischen System U, das der Benutzer verwendet, gibt es ein weiteres algebraisches System C, das vom Rechensystem verwendet wird. Um Elemente der Menge S des Benutzersystems in die Menge S' des Rechners zu überführen, werden zwei zueinander inverse Ver- bzw. Entschlüsselungsfunktionen

$$\phi^{-1} : S \mapsto S' \tag{3}$$

und

$$\phi : S' \mapsto S \tag{4}$$

benötigt.

Bevor der Benutzer einen Datensatz an das Rechensystem übergibt, verschlüsselt er diesen mit Hilfe der Funktion ϕ^{-1}. Damit das System in der Lage ist, Operationen auf den verschlüsselten Daten auszuführen, muss ϕ vollständig homomorph von C nach U sein. D. h. eine Funktion des algebraischen Systems des Rechensystems angewandt auf verschlüsselte Parameter muss auf ein verschlüsseltes Ergebnis führen, das dem unverschlüsselten Ergebnis der korrespondierenden Funktion des algebraischen Systems des Benutzers, angewandt auf die unverschlüsselten Parameter entspricht.

$$\begin{aligned}(\forall i)(a, b, c, \ldots)[f_i'(a, b, \ldots) = c \\ \Rightarrow f_i(\phi(a), \phi(b), \ldots) = \phi(c)]\end{aligned} \tag{5}$$

Prädikate des algebraischen Systems des Rechensystems angewendet auf verschlüsselte Parameter müssen zu dem gleichen Ergebnis führen, wie die entsprechenden Prädikate des algebraischen Systems des Benutzers, angewendet auf die unverschlüsselten Parameter.

$$(\forall i)(a, b, \ldots)p'(a, b, \ldots) \equiv p(\phi(a), \phi(b), \ldots) \tag{6}$$

Die verschlüsselten Konstanten müssen mit den unverschlüsselten Konstanten korrespondieren.

$$(\forall i)\phi(s_i') = s_i \tag{7}$$

Angenommen der Benutzer möchte den Wert von $f_1(d_1, d_2)$ berechnen. Dann fordert er das Rechensystem auf,

$$f_1'(\phi^{-1}(d_1), \phi^{-1}(d_2)) \tag{8}$$

zu bestimmen und ermittelt das unverschlüsselte Ergebnis der Berechnung zu

$$\phi(f_1'(\phi^{-1}(d_1), \phi^{-1}(d_2))) = f_1(d_1, d_2) \tag{9}$$

Rivest zufolge soll das algebraische System und die Funktionen ϕ und ϕ^{-1} folgende Anforderungen erfüllen:

1. ϕ und ϕ^{-1} sollten einfach zu berechnen sein.
2. f_i' und p_i' in C sollten effizient berechenbar sein.
3. Die verschlüsselte Version der Daten sollte nicht viel mehr Speicherplatz erfordern, als die unverschlüsselte Version.
4. Kenntnis von $\phi^{-1}(d_i)$ für viele Datenpunkte d_i sollte nicht ausreichen, um ϕ zu bestimmen.
5. Kenntnis von d_i und $\phi^{-1}(d_i)$ für einige Werte von d_i sollte nicht genügen, um ϕ zu bestimmen.
6. Kenntnis der Operationen und Prädikate in C sollte nicht genügen, um ϕ effizient zu bestimmen.

Tabelle 1: Messergebnisse von Smart und Vercauteren für die Ausführung ihres homomorphen Verschlüsselungsschemas für relativ kleine Schlüsselgrößen.

n	Verschlüsselung	Entschlüsselung	Multiplikation
	ms		
8	4,2	0,2	0,2
9	38,8	0,3	0,2
10	386,4	0,6	0,4
11	3717,2	3,0	1,6

Rivest ging zwar davon aus, dass vollständig homomorphe Verschlüsselungssysteme existieren, konnte jedoch kein praktisch verwendbares Beispiel eines solchen Systems angeben.

3.3.2 Verwirklichung der Vertraulichkeits-Homomorphismen

Erst acht Jahre später gelang Yao die Konstruktion eines Verschlüsselungsschemas [8], das Rivests Anforderungen zumindest teilweise genügte. Das Konzept der Cloud war Yao noch nicht bekannt. Sein Schema ist auf dieses aber trotzdem anwendbar und garantiert, dass der Cloud-Anbieter durch die Berechnungen, die er auf den verschlüsselten Daten ausführt, weder Informationen über die Daten selbst, noch die Ergebnisse der Berechnungen erhält. Yaos Protokoll erfordert jedoch mehrere Interaktionen zwischen Cloud-Anbieter und Kunde und ist rechenintensiv.

Gentry veröffentlichte 2009 einen Ansatz, der die Notwendigkeit für diese Interaktionen beseitigt [9]. Sein Verschlüsselungsschema arbeitet auf algebraischen Verbänden, wodurch es zwar sehr mächtig ist, allerdings auch ein hohes Maß an Rechenleistung erfordert [10].

Für vertrauliche Berechnungen in der Cloud ist es allerdings hinreichend, wenn ein Vertraulichkeits-Homomorphismus mit Ganzzahlen und Polynomen rechnen kann. Smart und Vercauteren geben deswegen einen Spezialfall von Gentrys Verschlüsselungsschema an, der sich auf ganze Zahlen und Polynomen beschränkt. Diese Einschränkung erleichtert die Implementierung des Ansatzes und sorgt dafür, dass der Rechenaufwand auf ein Maß sinkt, das es realistisch erscheinen lässt, dass Vertraulichkeits-Homomorphismen in Zukunft in der Cloud Verwendung finden werden [11].

Das Schema ist trotz dieser Verbesserungen aber noch nicht praktisch anwendbar, da es zur Schlüsselerzeugung Primzahlen benötigt, die über 90.000 Bit lang sind. Smart

und VERCAUTEREN ist es nicht gelungen, Schlüssel von hinreichender Länge zu erzeugen. Tabelle 1 zeigt die von SMART und VERCAUTEREN gemessenen Rechenzeiten für Verschlüsselung, Entschlüsselung und Multiplikation in einem teilweise homomorphen Verschlüsselungsschema. Der Parameter n ist verantwortlich für das Maß an Sicherheit, welches das Schema bietet. SMART et al. halten die für die Messung verwendeten Werte von n für zu niedrig, sind jedoch der Ansicht, dass bereits Berechnungen für $n = 15$ außerhalb ihrer Möglichkeiten liegen. Zum Vergleich: Auf einem aktuellen Intel-Rechner (Core2-Architektur mit 2,4 GHz Systemtakt) dauert die Ausführung einer leeren Funktion ca. 400 ns, die Berechnung einer Multiplikation weniger als 10 ns.

4. PRAKTISCHE ANSÄTZE

Der vorangegangene Abschnitt hat gezeigt, dass vertrauliche Berechnungen in der Cloud zwar prinzipiell möglich, in Anbetracht des aktuellen Stands der Forschung jedoch nicht umzusetzen sind. Dieser Abschnitt wird deswegen zunächst darauf eingehen, welche Einschränkungen des Vertraulichkeitsbegriffs in Zusammenhang mit der Cloud hinnehmbar sind und dann Ansätze für vertrauliche Berechnungen vorstellen, die unter der Annahme eines eingeschränkten Vertraulichkeitsbegriffs entworfen wurden.

4.1 Sicherheits- und Vertraulichkeitsbegriff

NARAYANAN und SHMATIKOV [12] schlagen vor, für den Fall, dass Vertraulichkeits-Homomorphismen zur Verwirklichung des Datenschutzes nicht in Frage kommen, den Begriff der Vertraulichkeit folgendermaßen einzuschränken:

Es soll einfach sein, ein einzelnes Datum oder eine kleine Untermenge von Daten in Erfahrung zu bringen, falls derjenige, der diese Daten in Erfahrung bringen möchte, sie genau identifizieren kann. Alle Anfragen, die nicht den Datenschutzrichtlinien entsprechen, sollen dagegen nur mit sehr großem Rechenaufwand durchführbar sein. Beispielsweise soll es möglich sein, die persönlichen Daten eines Nutzers in Erfahrung zu bringen, wenn die interessierte Partei bereits dessen vollständigen Namen und sein Geburtsdatum kennt. Dagegen soll es schwierig sein, beliebige persönliche Daten abzufragen, für die dieser Schlüssel nicht bekannt ist.

Datenschutz so umzusetzen wird als *Verschleierung (obfuscation)* bezeichnet, um den Ansatz von der *Verschlüsselung (encryption)* abzugrenzen. Bei diesem Ansatz wird der Recheneffizienz eine höhere Priorität zugestanden als der Sicherheit.

4.2 Trusted Cloud Computing Platform

SANTOS et al. schlagen zur Lösung des Datenschutzproblems der Cloud – in Anbetracht des unzureichenden Stands der Forschung auf dem Gebiet der Vertraulichkeits-Homomorphismen – einen anderen Weg vor. Ziel ihres Konzepts ist es nicht, Berechnungen verschlüsselt auszuführen, sondern dem Benutzer der Cloud eine Ausführungsumgebung zu bieten, der er vertrauen kann. Zu diesem Zweck schlagen sie die Verwendung kryptographischer Hardware in Form einer Trusted Cloud Computing Platform vor [13].

4.2.1 Hintergrund – TPM

Hierbei soll der Trusted-Platform-Module-Chip der *Trusted Computing Group* [14, 15, 16] zum Einsatz kommen. Dieser enthält einen *Indossament-Schlüssel (endorsement key*[1]*)*, der während der Produktion des Chips in der Hardware verankert wird und deswegen zur Identifikation des Chips dienen kann. Der Chip ist in der Lage, einer entfernten Partei glaubhaft zu versichern, dass das System, mit dem sie kommuniziert, dasjenige ist, das sie erwartet hat. Hierzu berechnet der Trusted-Platform-Module-Chip während des Boot-Vorgangs Prüfsummen über die Software, die am Startvorgang beteiligt ist. Zur Verwahrung der Prüfsummen verfügt der Chip über eine Reihe von Registern (*Platform Configuration Register,* PCR).

Die *Core Root of Trust Measurement* ist ein Teil des *Basic Input Output System* (BIOS), der selbst bei einer Aktualisierung desselbigen nicht verändert wird. Sie ist dafür zuständig, während des Startvorgangs zunächst eine Prüfsumme über den Programmcode und die Parameter des BIOS zu erstellen und diese an das Platform Configuration Register mit der Nummer 0 anzuhängen. Anschließend wird eine Prüfsumme des Bootloaders erstellt und ebenfalls an dieses Platform Configuration Register angehängt.

Erst danach übergibt das BIOS die Steuerung des Bootvorgangs an den Bootloader, der die Prüfsumme des Betriebssystemkerns berechnet. Auf diese Art wird eine *Kette des Vertrauens* erstellt, die sich vom BIOS über den Bootloader und das Betriebssystem bis hin zu den Anwendungen, die auf dem Rechner betrieben werden, erstrecken kann.

Eine entfernte Partei schickt an den Trusted-Platform-Module-Chip eine *Zufallszahl zur einmaligen Verwendung* (*nonce* [18]) und fordert ihn auf, die Prüfsummenliste und die Zufallszahl mit dem *privaten* Indossament-Schlüssel zu verschlüsseln. Die Zufallszahl kann der befragte Trusted-Platform-Module-Chip nicht vorhersehen, weswegen sie einen Angriff durch Wiedereinspielung verhindert.

Das Ergebnis schickt der Trusted-Platform-Module-Chip zurück. Die entfernte Partei kann dieses dann mit Hilfe des *öffentlichen* Indossament-Schlüssels entschlüsseln und so verifizieren, dass ihr Kommunikationspartner tatsächlich der ist, der er vorgibt zu sein. Anhand der Prüfsummenliste kann die entfernte Partei prüfen, ob die Software, die auf ihrem Kommunikationspartner läuft, vertrauenswürdig ist. Die Trusted Cloud Computing Platform erweitert diesen Mechanismus so, dass sich nicht nur ein einzelnes System authentifizieren kann, sondern eine vollständige Cloud.

4.2.2 Architektur

Auf jedem Knoten der *Trusted Cloud Computing Platform* kommt ein *Trusted Virtual Machine Monitor* zum Einsatz, der dafür zuständig ist, die Vertraulichkeit und Integrität des Knotens sicherzustellen. Das bedeutet, dass der Trusted Virtual Machine Monitor dafür sorgt, dass Unbefugte weder Einsicht in den Speicherzustand der virtuellen Maschine nehmen können, noch in der Lage sind, diesen zu manipulieren.

Der *Trusted Coordinator* ist ein System, auf das Administratoren der Cloud keinen Zugriff haben dürfen. Im Optimalfall wird dieses System von einer externen, vertrauenswürdigen Partei betrieben. Die Aufgabe des Trusted Coordinator ist es, eine Menge von Knoten zu verwalten, auf denen der

[1]Der Begriff *Indossament* stammt aus dem Wirtschaftsrecht und bezeichnet einen zwingend vorgeschriebenen schriftlichen Vermerk, durch den das Eigentum eines Orderpapiers (Schecks, ...) vom bisherigen Inhaber auf einen neuen Eigentümer übergeht. Siehe [17].

Cloud-Benutzer seine virtuellen Maschinen sicher ausführen kann. Diese vertrauenswürdigen Knoten müssen sich in einem sicheren Bereich befinden und den Trusted Virtual Machine Monitor betreiben. Der Trusted Coordinator führt eine Liste der Knoten, die sich in dem sicheren Bereich befinden und bescheinigt, dass auf diesen Knoten ein Trusted Virtual Machine Monitor läuft.

Gemeinsam sollen Trusted Coordinator und Trusted Virtual Machine Monitor zwei Ziele erreichen. Sie sorgen zum einen dafür, dass virtuelle Maschinen ausschließlich auf vertrauenswürdigen Knoten ausgeführt werden. Zum anderen schützen sie den Zustand der virtuellen Maschine vor Einsichtnahme und Manipulation, während diese über das Netzwerk übertragen wird, indem sie den Zustand der virtuellen Maschine vor der Übertragung verschlüsseln und signieren.

4.3 Privacy Manager

Pearson, Shen und Mowbray schlagen die Verwendung eines *Privacy Managers* [10] vor, um Benutzer bei der Verwaltung ihrer Privatsphäre zu unterstützen. Dieser bietet die Möglichkeit, sowohl feingranular einzelne *Aspekte (preferences)* des Datenschutzes zu konfigurieren, als auch grobgranular Einstellungen vorzunehmen, die mehrere Aspekte zu einer *Rolle (persona)* zusammenfassen. Die Auswahl einer Rolle ermöglicht es dem Benutzer beispielsweise vollständig anonym aufzutreten oder aber Teilaspekte seiner Privatsphäre für die Nutzung in der Cloud freizugeben.

Der Privacy Manager bietet einen Ver- und Entschleierungsdienst für die in der Cloud gelagerten Daten gemäß der Benutzereinstellungen an. Für die Implementierung der unteren Schichten des Datenschutzes schlagen die Autoren das Trusted Platform Module vor. Dieses soll auf dem Client-Rechner die Verschleierungsschlüssel sicher verwahren und die Integrität des Privacy Managers sicherstellen.

Als Ort, an dem der Privacy Manager ausgeführt werden kann, schlagen Pearson et al. verschiedene Lösungen vor. Beispielsweise könnte der Privacy Manager im Client-Rechner betrieben werden, was das höchste Maß an Sicherheit gewährleisten würde.

Er könnte allerdings auch in einer privaten Cloud laufen, die mit der öffentlichen Cloud kommuniziert. Dies hätte den Vorteil, dass der Privacy Manager nicht nur einem einzelnen Mitarbeiter einer Firma zur Verfügung stehen würde, sondern dem gesamten Unternehmen. Der entscheidende Nachteil dieser Lösung ist die Tatsache, dass der Privacy Manager einen Engpass darstellen würde, weswegen sie möglicherweise nicht gut skaliert.

Eine weitere Möglichkeit, den Privacy Manager zu platzieren, wäre die Cloud in mehrere Vertraulichkeitsbereiche aufzuteilen. Sollen Daten von einem Vertraulichkeitsbereich in einen anderen übertragen werden, so entscheidet ein Privacy Manager gemäß den Wünschen des Benutzers, ob dieser Transfer stattfinden darf und ent- bzw. verschleiert die Daten. Um Transparenz und Nachvollziehbarkeit zu gewährleisten, würde der Privacy Manager den Benutzer über die Datenübergabe informieren. Die Vertrauenswürdigkeit des Privacy Managers könnte auch hier durch eine Hardwarelösung wie das Trusted Platform Module sichergestellt werden.

4.4 Schwachpunkte des TPM

Der in den vorangegangenen Abschnitten beschriebene Ansatz, Datenschutz in der Cloud zu verwirklichen, basiert auf der Annahmen, dass der Trusted-Platform-Module-Chip vertrauenswürdig ist. Sparks [19] und Kauer [20] konnten jedoch unabhängig voneinander zeigen, dass dies nur eingeschränkt der Fall ist.

Die Vertraulichkeitsgarantien des Trusted Platform Module beruhen darauf, dass für jeglichen Programmcode, der auf dem System ausgeführt wird, vor der Ausführung eine Messung vorgenommen wird. Das Ergebnis dieser Messung wird an den im Platform Configuration Register derzeit gespeicherten Wert angehängt und mittels SHA-1 gehasht [21]:

$$\mathrm{PCR}_{\mathrm{neu}} := \text{SHA-1}(\mathrm{PCR}_{\mathrm{alt}} || M) \tag{10}$$

wobei $\mathrm{PCR}_{\mathrm{neu}}$ und $\mathrm{PCR}_{\mathrm{alt}}$ den neuen bzw. alten Zustand des Platform Configuration Registers, $||$ die Konkatenation und M die neue Messung symbolisiert.

Aufgrund der Konkatenation und der kryptographischen Eigenschaften der SHA-1-Hashfunktion – sie ist einfach zu berechnen, es ist nicht möglich vom Bild auf das Urbild zu schließen, es ist nicht möglich ein zweites Urbild zu finden, das auf das Bild abgebildet wird [22] – ist es sehr schwierig, eine solche Kette von Hashwerten zu fälschen. Wang et al. haben jedoch einen Weg vorgestellt, mit verhältnismäßig geringem Rechenaufwand Kollisionen für SHA-1-Hashes zu finden, was Grund zu der Annahme gibt, dass der SHA-1-Algorithmus als kryptographische Hashfunktion mittelfristig unbrauchbar [23] wird.

Die Anhängeoperation ist die einzige schreibende Operation, die auf einem Platform Configuration Register möglich ist. Deswegen kehrt ein Platform Configuration Register nie in seinen Initialzustand zurück, es sei denn der Trusted-Platform-Module-Chip wird zurückgesetzt.

Das Trusted Platform Module nimmt lediglich eine Messung beim Laden des Programms vor. Ist das Programm erst einmal im Speicher, so bemerkt das Trusted Platform Module Modifikationen am Programmcode nicht mehr.

Schellekens et al. geben zu bedenken, dass es nicht trivial ist, dafür zu sorgen, dass eine Ausführungsumgebung und die von ihr geladenen Programme vertrauenswürdig sind [21]. Aktuelle Betriebssysteme verfolgen einen monolithischen Ansatz, weswegen die Codebasis, über die das Trusted Platform Module eine Bescheinigung ausstellen muss, zu groß ist, als dass die Bescheinigung glaubwürdig erscheinen könnte. Auch ist die Zahl der möglichen Konfigurationen, in denen die zu attestierende Software auftreten kann, sehr groß – jede neue Version hat einen neuen Hashwert, der in einer Datenbank verfügbar gemacht werden muss.

Einige spezielle Angriffe gegen das Trusted Platform Module werden in den folgenden Abschnitten vorgestellt.

4.4.1 Angriffe gegen die Software

Das Trusted Platform Module errechnet Hashwerte über die geladene Software und liefert so dem Benutzer Informationen über das Maß an Vertrauen, das dem System zusteht. Es beinhaltet jedoch keinen Mechanismus, der sicherstellt, dass sich der Systemzustand seit der letzten Messung nicht verändert hat [19].

Linux sorgt dafür, dass der ausführbare Code eines Programms (das Text-Segment) nach dem Laden in der Seitenkacheltabelle als nicht veränderbar markiert wird. Insofern ist diese Einschränkung des Trusted Platform Module unter normalen Umständen kein Problem, da ein Seitenfehler ausgelöst wird, wenn schreibend auf eine solche Seite zugegriffen

wird. Der bereits als vertrauenswürdig bescheinigte Kernel erhält dann die Kontrolle und kann die Situation bearbeiten.

Sparks war mit Hilfe eines Kernelmoduls in der Lage, die Seitenkacheltabelle dahingehend zu verändern, dass das Text-Segment eines beliebigen auf dem Rechner laufenden Programms schreibbar wurde, ohne dass das Trusted Platform Module dies bemerkt hätte.

Ähnliche Angriffe sind auch unter Microsoft Windows möglich, da Gerätetreiber Schreibzugriff auf beliebige Speicheradressen haben.

4.4.2 Angriff durch Zurücksetzen

Der *Angriff durch Zurücksetzen* macht sich die Tatsache zunutze, dass der Trusted-Platform-Module-Chip an den Low-Pin-Count-Bus angeschlossen ist, mit dem auch das BIOS und entwicklungsgeschichtlich ältere Ein- bzw. Ausgabekomponenten verbunden sind. Dieser Bus verfügt über eine Initialisierungsleitung, über die sich der Trusted-Platform-Module-Chip mit Hilfe eines relativ einfach durchzuführenden Hardwareeingriffs im laufenden Betrieb zurücksetzen lässt. Ist dies gelungen, so kann zuvor aufgezeichneter Datenverkehr des Low-Pin-Count-Bus verwendet werden, um die Platform Configuration Register des Chips mit gültigen Werten zu beschreiben, obwohl der durch diese Werte attestierte Systemzustand nie erreicht wurde.

Dieser Angriffsvektor erfordert physikalischen Zugang zur Hardware des Systems, dürfte in einem Rechenzentrum also schwierig unbemerkt durchzuführen sein. Kauer beschreibt jedoch noch weitere Angriffstechniken, die ohne einen Eingriff in die Hardware auskommen.

4.4.3 Angriff gegen den Bootloader

Sparks sieht eine mögliche Schwachstelle des Trusted Platform Module im Bootloader. Zum Veröffentlichungszeitpunkt seiner Arbeit gab es drei Bootloader, die in der Lage waren den Trusted-Platform-Module-Chip zu verwenden: LILO, GRUB und TrustedGRUB. Keiner dieser drei Bootloader wird den Anforderungen eines Trusted-Platform-Module-geschützten Systems gerecht.

LILO versäumt es, den Real Mode Setup Code des Linux-Kernels, der dafür zuständig ist, den eigentlichen Kernel zu laden, in die Hashberechnung mit einzubeziehen.

GRUB lädt den Kernel zwei Mal – das erste Mal um den Hash zu berechnen und das zweite Mal um den Kernel auszuführen. Ein Angreifer könnte den Kernel zwischen diesen beiden Ladevorgängen austauschen, so dass das Trusted Platform Module die Ausführung eines anderen Kernels attestiert, als tatsächlich auf dem System läuft.

TrustedGRUB schließlich bezieht seinen eigenen Code nicht in die Berechnung des Hashs mit ein.

Somit unterbrechen alle drei Bootloader die Vertrauenskette, wodurch jeglicher nach der Unterbrechung ausgeführter Code als nicht vertrauenswürdig eingestuft werden muss.

4.4.4 Angriff durch Zeitmessung

Sparks ist der Ansicht, dass ein *Angriff durch Zeitmessung* gegen die RSA-Implementierung des Trusted Platform Modules vielversprechend ist. Er schlägt vor, den von Brumley und Boneh in [24] beschriebenen Angriff gegen RSA-Implementierungen, die auf dem *Chinesischen Restesatz* [25] basieren, zu verwenden, da dies auf die von ihm untersuchte Implementierung zutrifft. Für diesen Angriff ist es erforderlich, hochauflösende Messungen der Ausführungszeit von Operationen, die den privaten Schlüssel des Trusted Platform Modules verwenden, vorzunehmen.

Sparks schlägt vor, die `TPM_Seal`-Funktion mit entsprechend vorbereiteten Zeichenketten aufzurufen und die Ausführungszeiten zu messen. Der private Schlüssel sollte so innerhalb von 40 Tagen zu ermitteln sein. Da dies jedoch den zeitlichen Rahmen seiner Arbeit überstiegen hätte, war er nicht in der Lage den Angriff zu testen. Gelingt er aber tatsächlich, so untergräbt dieser Angriff jegliche Sicherheits- und Vertraulichkeitsgarantien, die das Trusted Platform Module verspricht.

4.4.5 Angriff gegen das BIOS

Die letzte Angriffstechnik setzt am BIOS an. Das BIOS ist in Bezug auf das Trusted Platform Module die Core Root of Trust Measurement, d. h. ist das BIOS kompromittiert, so sind alle Messungen des Trusted-Platform-Module-Chips wertlos.

Moderne Intel-basierte Rechner erlauben es, das BIOS zu überschreiben, um es zu ermöglichen, bei bereits im Betrieb befindlichen Systemen nachträglich Fehler zu beseitigen. Im BIOS eines Trusted-Platform-Module-fähigen Systems befindet sich ein Treiber für den Trusted-Platform-Module-Chip, den Kauer durch die Umsetzung eines einzelnen Bits dazu bewegen konnte, während des Bootvorgangs vorgenommene Messungen nicht mehr an den Chip zu senden. Dadurch verbleibt das Platform Configuration Register in seinem Initialzustand und kann zu einem späteren Zeitpunkt beliebig beschrieben werden.

Es ist zu erwarten, dass das *Extensible Firmware Interface (EFI)* [26] in naher Zukunft die Rolle des BIOS übernehmen wird. Da EFI nicht grundlegend anders funktioniert, als das BIOS, ist nicht davon auszugehen, dass es in Bezug auf die Sicherheit bei TPM einen Vorteil bietet.

4.4.6 Ansatz zur Behebung der Schwachstellen

Kauer schlägt vor, eine *Dynamic Root of Trust Measurement* zu verwenden, um den beschriebenen Angriffen entgegenzuwirken. In diesem Fall bietet das Platform Configuration Register 17 eine besondere Funktionalität – beim Zurücksetzen des Trusted Platform Modules geht dieses nicht in den Zustand „0“, sondern in den Zustand „-1“ über.

Bei Verwendung einer Dynamic Root of Trust Measurement würden das BIOS, die OptionROMs und der Bootloader aus der Kette des Vertrauens entfernt. Der Hauptprozessor kann das Platform Configuration Register 17 jederzeit durch Aufruf der `skinit`-Anweisung zurücksetzen. Diese initialisiert den Hauptprozessor und lädt einen *Secure Loader* in dessen Cache. Die Prüfsumme des Secure Loaders wird in das Platform Configuration Register 17 geschrieben. Anschließend wird der Secure Loader ausgeführt. Die beschriebenen Initialisierungsschritte laufen dabei atomar ab.

Der Angriff durch Zurücksetzen funktioniert bei der Verwendung einer Dynamic Root of Trust Measurement wegen des besonderen Startzustands des Platform Configuration Register 17 nicht, da dieses erst durch das Ausführen der Anweisung, die die oben beschriebenen Initialisierungsschritte vornimmt, einen korrekten Initialzustand erreicht.

Zusammenfassend lässt sich sagen, dass eine vertrauenswürdige Cloud auf Basis des Trusted Platform Module schwie-

rig zu verwirklichen ist. Ihre Umsetzung stellt hohe Anforderungen an die Fähigkeiten und Kentnisse der Entwickler, die sie entwerfen. Möchte ein Cloud-Benutzer das Trusted Platform Module verwenden, so ist er auf die Kooperation des Cloud-Betreibers angewiesen.

Sobald Administratoren unbeobachtet Zugang zur Hardware der Cloud erhalten, bietet das Trusted Platform Module keine Garantien mehr. Sollten sich Angriffe durch Zeitmessung als durchführbar erweisen, so sind viele derzeit erhältliche Trusted Platform Module unbrauchbar.

5. FAZIT

Sowohl die Rechtslage, als auch die Sorge um die eigene Reputation motivieren Nutzer von Cloud-Diensten, die von ihnen in der Cloud gespeicherten und verarbeiteten Daten vor Einsichtnahme oder Manipulation durch Dritte zu schützen. Auch wäre es für viele Unternehmen fatal, wenn ihre Geschäftsgeheimnisse durch Sicherheitslücken Mitbewerbern bekannt würden. Angriffe auf eine Cloud können sowohl von Mitarbeitern des Cloud-Betreibers, als auch durch andere Kunden, deren virtuelle Maschinen auf der gleichen Hardware laufen, durchgeführt werden. Folglich besteht ein Bedarf an Mechanismen, die die vertraulichen Daten schützen.

Prinzipiell gibt es zwei Ansätze, die versuchen Datenschutz in der Cloud zu verwirklichen. Auf der einen Seite stehen die vollständig homomorphen Verschlüsselungsschemata, deren Ziel es ist, dafür zu sorgen, dass sensible Daten nicht einsehbar sind, aber verarbeitet werden können. Theoretisch ist ein Schutz der Daten durch die Verwendung dieser Schemata möglich. Dies würde in allen Aspekten dem geltenden Datenschutzrecht genügen, da dann weder der Cloud-Betreiber noch ein anderer nicht dazu Befugter die Möglichkeit hätte, herauszufinden, auf welchen Daten welche Berechnungen stattfinden und was die Ergebnisse dieser Berechnungen sind. Jeder Angriff auf die Privatsphäre des Nutzers würde so abgewehrt.

Die bisher entwickelten Schemata dieser Art erfordern jedoch ein hohes Maß an Rechenleistung auf Seiten des Cloud-Benutzers, was dem Grundgedanken der Cloud – aufwändige Berechnungen durch einen externen Dienstleister, der über die dafür notwendigen Betriebsmittel verfügt, ausführen zu lassen – zuwider läuft.

Unter der Annahme, dass in einer Cloud Recheneffizienz einen höheren Stellenwert als Sicherheitsanforderungen hat, erscheint deswegen ein anderer Ansatz sinnvoll. Vertreter dieses Ansatzes argumentieren, dass es hinreichend ist, wenn der Vertraulichkeitsbegriff dahingehend eingeschränkt wird, dass es sehr aufwändig sein muss, unbefugt auf Daten zuzugreifen. Dieser Ansatz zielt darauf ab, dass der Benutzer Vertrauen in das System hat, was unter anderem durch den Einsatz des Trusted Platform Module erreicht werden soll.

Ob dem Bundesdatenschutzgesetz hiermit noch genügt wird, ist allerdings fraglich. Insbesondere, wenn die Cloud Berechnungen auf hochsensiblen Daten wie Finanz- oder Patientendaten ausführen soll, erscheint unbefugter Zugriff auch dann noch inakzeptabel, wenn nur einzelne Datenpunkte betroffen sind.

Ein Privacy Manager kann Kunden von Cloud-Benutzern dabei helfen, Einstellungen bezüglich ihrer Privatsphäre vorzunehmen. Dieser Ansatz erscheint sinnvoll, da er dem vom Bundesverfassungsgericht bereits 1983 definierten *informationellen Selbstbestimmungsrecht* [27] entspricht.

Aktuelle Ansätze zur Wahrung der Privatsphäre in der Cloud auf Basis des Trusted Platform Module, sind nur eingeschränkt vertrauenswürdig, da Forscher bereits erfolgversprechende Angriffe veröffentlicht haben. Ob eine Cloud, die auf dem Trusted Platform Module basiert, der geltenden Rechtslage genügt, wird im Einzelfall zu entschieden sein. Sollte sich jedoch herausstellen, dass Angriffe durch Zeitmessung beim Trusted Platform Module tatsächlich funktionieren, so kann es nicht mehr als Garant für die Privatsphäre der Kunden von Cloud-Benutzern gelten.

Derzeit werden vollständig homomorphe Verschlüsselungsschemata intensiv erforscht. Es bleibt abzuwarten, ob diese sich dahingehend erweitern lassen, dass die Schlüsselerzeugung und Verschlüsselung in akzeptabler Zeit erfolgen kann. Sollte dies der Fall sein, sind die Homomorphismen dem Trusted Platform Module gegenüber vorzuziehen.

6. LITERATUR

[1] *Das europäische Parlament und der Rat: Richtlinie 95/46/EG*. Amtsblatt der Europäischen Gemeinschaften Nr. 1, 281/31, 1995

[2] Doelitzscher, Frank ; Reich, Christoph ; Sulistio, Anthony: Designing Cloud Services Adhering to Government Privacy Laws. In: *Proceedings of the 3rd IEEE International Symposium on Trust, Security and Privacy for Emerging Applications (TSP-10)*, 2010

[3] *Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. Januar 2003 (BGBl. I S. 66), das zuletzt durch Artikel 1 des Gesetzes vom 14. August 2009 (BGBl. I S. 2814) geändert worden ist.*

[4] Ristenpart, Thomas ; Tromer, Eran ; Shacham, Hovav ; Savage, Stefan: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA : ACM, 2009. – ISBN 978–1–60558–894–0, S. 199–212

[5] *Data Encryption Standard (DES)*. Bd. *46-3*. U. S. Department of Commerce/National Institute of Standards and Technology, 1999

[6] Rivest, L. R. ; Shamir, A.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications of the ACM* 21 (1978), S. 120–126

[7] Rivest, L. R. ; Adleman, L. ; Dertouzos, M.: On data banks and privacy homomorphisms, Academic Press, 1978, S. 169–177

[8] Yao, Andrew Chi-Chih: How to generate and exchange secrets. In: *Proceedings of the 27th Annual Symposium on Foundations of Computer Science* (1986), S. 162–167. – ISSN 0272–5428

[9] Gentry, Craig: Fully homomorphic encryption using ideal lattices. In: *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*. New York, NY, USA : ACM, 2009. – ISBN 978–1–60558–506–2, S. 169–178

[10] Pearson, Siani ; Shen, Yun ; Mowbray, Miranda: A Privacy Manager for Cloud Computing. In: *CloudCom '09: Proceedings of the 1st International Conference on Cloud Computing*. Berlin, Heidelberg : Springer-Verlag, 2009. – ISBN 978–3–642–10664–4, S. 90–106

[11] Smart, Nigel P. ; Vercauteren, Frederik: Fully Homomorphic Encryption with Relatively Small Key

and Ciphertext Sizes. In: *PKC 2010, LNCS 6056*, International Association for Cryptologic Research, 2010, S. 420–443

[12] Narayanan, Arvind ; Shmatikov, Vitaly: Obfuscated databases and group privacy. In: *In Proceedings of the 12th ACM Conference on Computer and Communications Security*, ACM, 2005, S. 102–111

[13] Santos, Nuno ; Gummadi, Krishna P. ; Rodrigues, Rodrigo: Towards Trusted Cloud Computing. In: *HOTCLOUD*, 2009

[14] Trusted Computing Group, Incorporated (Hrsg.): *TPM Main Part 1 Design Principles*. Specification Version 1.2 Level 2 Revision 103. Trusted Computing Group Administration, 3855 SW 153rd Drive, Beaverton, Oregon 97006: Trusted Computing Group, Incorporated, July 2007

[15] Trusted Computing Group, Incorporated (Hrsg.): *TPM Main Part 2 TPM Structures*. Specification Version 1.2 Level 2 Revision 103. Trusted Computing Group Administration, 3855 SW 153rd Drive, Beaverton, Oregon 97006: Trusted Computing Group, Incorporated, July 2007

[16] Trusted Computing Group, Incorporated (Hrsg.): *TPM Main Part 3 Commands*. Specification Version 1.2 Level 2 Revision 103. Trusted Computing Group Administration, 3855 SW 153rd Drive, Beaverton, Oregon 97006: Trusted Computing Group, Incorporated, July 2007

[17] Holzhammer, Richard: *Allgemeines Handelsrecht und Wertpapierrecht*. 8., überarbeitete Auflage. Springer, 1998. – ISBN 9783211831564

[18] Needham, Roger M. ; Schroeder, Michael D.: Using encryption for authentication in large networks of computers. In: *Communications of the ACM* 21 (1978), Nr. 12, S. 993–999. – ISSN 0001–0782

[19] Sparks, Evan R.: A Security Assessment of Trusted Platform Modules / Dartmouth College, Computer Science. Hanover, NH, June 2007 (TR2007-597). – Forschungsbericht

[20] Kauer, Bernhard: OSLO: Improving the security of Trusted Computing / Technische Universität Dresden. Dresden, 2007. – Forschungsbericht

[21] Schellekens, Dries ; Wyseur, Brecht ; Preneel, Bart: Remote attestation on legacy operating systems with trusted platform modules. In: *Science Computer Programming* 74 (2008), Nr. 1-2, S. 13–22. – ISSN 0167–6423

[22] Paar, Christof ; Pelzl, Jan: *Understanding Cryptography: A Textbook for Students and Practitioners*. 1st Edition, 2nd Printing. Springer, 2009. – ISBN 9783642041006

[23] Wang, Xiaoyun ; Yin, Yiqun L. ; Yu, Hongbo: Finding Collisions in the Full SHA-1, Springer, 2005, S. 17–36

[24] Brumley, David ; Boneh, Dan: Remote Timing Attacks are Practical. In: *In Proceedings of the 12th USENIX Security Symposium*, 2003, S. 1–14

[25] Knuth, Donald: *The Art of Computer Programming*. Bd. 2: Seminumerical Algorithms. Third Edition. Addison-Wesley, 1997

[26] Singh, Amit: *Mac OS X Internals: A Systems Approach*. Addison-Wesley Professional, 2006. – ISBN 9780321278548

[27] Benda, Ernst ; Simon, Helmut ; Hesse, Konrad ; Katzenstein, Dietrich ; Niemeyer, Gisela ; Heussner, Hermann ; Henschel, Johann F.: BVerfGE 65, 1. In: *Entscheidungen des Bundesverfassungsgerichts 65*. Mohr, Tübingen : Mitglieder des Bundesverfassungsgerichts, 1983. – ISSN 0433–7646