Mechanismen für Vertrauenswürdigkeit und Sicherheit in Cloud Computing Infrastrukturen

Florian Franzmann

siflfran@hawo.stw.uni-erlangen.de

Ausgewählte Kapitel der Systemsoftware
Cloud Computing

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität Erlangen-Nürnberg

8. Juli 2010

Motivation

Motivation

- Firmen wollen Berechnungen in der Cloud ausführen

$\Rightarrow$ Daten werden in der Cloud

  - gelagert
  - verarbeitet

Daten umfassen

- Geschäftsgeheimnisse
- personenbezogene Daten
  - Datenschutzrecht anwendbar
  - Veröffentlichung rufschädigend

$\Rightarrow$ **Datenschutzproblem** der Cloud

Datenschutzrecht in Deutschland

Personenbezogene Daten

- Verhältnisse natürlicher Person
- namentliche Erwähnung nicht notwendig
- Identifizierbarkeit reicht

Geltungsbereich Bundesdatenschutzgesetz (BDSG)

- Erhebung
- Verarbeitung
- Nutzung

Bundesdatenschutzgesetz (BDSG)

Grundsatz: Datenerhebung prinzipiell untersagt

- es sei denn { Gesetz erlaubt Erhebung / Einverständniserklärung
- Einverständnis zweckgebunden
- Prinzip der Datensparsamkeit
- Umgehung durch Transfer ins Ausland verboten

$\Rightarrow$ **Widerspruch** zum Cloud-Ansatz

Inhalt

1 Angriffsvektoren

2 Theoretische Ansätze

3 Praktische Ansätze

4 Schwachpunkte der praktischen Ansätze

Gefahrenquelle Cloud-Betreiber-Mitarbeiter

- Systemadministrator: `root`-Rechte
  - Installation von Software
  - Ausführung von Programmen
  - Zugriff auf Arbeitsspeicher

Gegenmaßnahmen

- Rechteanhäufung verhindern
- Sicherheitsgeräte/Kameras
- keine Rechte für Administrator auf Produktivsystem

Angriff durch Mitbewerber

Gefahrenquelle Mitbewerber

- mehrere Kunden auf gleicher Hardware
- Side-Channel-Angriff
- Ablauf des Angriffs
  1. Platzierung der virtuellen Maschine
  2. Extraktion von Daten durch Messung der Prozessorlast

Einzige verlässliche Lösung

Exklusiver Zugang zur Hardware

Angriff durch den Staat

Polizeibehörden

- Auskunftspflicht?
- Betreiberhaftung?

Industriespionage durch Geheimdienste

- 20 % aller deutschen Unternehmen betroffen[1]
- 50 Mrd. € Verlust pro Jahr für Deutschland
- entspricht $\approx$ 30.000 Arbeitsplätzen

[1] The Guardian: *Germany accuses China of industrial espionage*. 22. Juli 2009

Inhalt

1 Angriffsvektoren

2 Theoretische Ansätze

3 Praktische Ansätze

4 Schwachpunkte der praktischen Ansätze

Traditionelle Verschlüsselung

Kryptographische Algorithmen

symmetrisch: AES, Blowfish, ...

asymmetrisch: Diffie-Hellman, ElGamal, ...

- nur für Datenlagerung
- nur noch Entschlüsselung möglich

$\Rightarrow$ **nicht ausreichend** für die Cloud

Ansatz I: Kryptographische Hardware

Kryptographische Hardware[1]

Rivests Idee (1978)

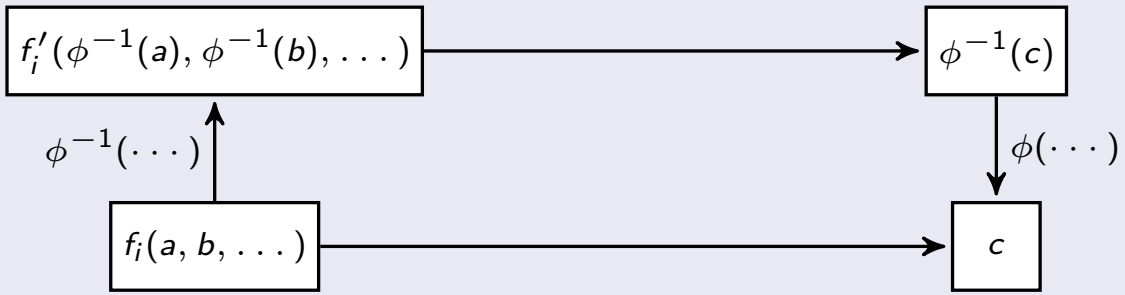
- modifizierte arithmetisch-logische Einheit (ALU)
- Daten verschlüsselt im Hauptspeicher
- Ver-/Entschlüsselung atomar durch ALU
- Beispiel: DS5240[2], 25 MHz Systemtakt

[1] nach Rivest et al.: *On Data Banks and Privacy Homomorphisms*. 1978.

[2] Elbaz et al.: *Hardware Engines for Bus Encryption: a Survey of Existing Techniques*. 2005.

Vertraulichkeits-Homomorphismen

Vertraulichkeits-Homomorphismus



Definition (Rivest, 1978)

- Ver-/Entschlüsselungsfunktion ϕ^{-1}/ϕ
- Funktion f_i' im Rechensystem korrespondiert mit f_i beim Benutzer
- für jedes $f_i'(a, b, \dots) = c$ gilt $f_i(\phi(a), \phi(b), \dots) = \phi(c)$
- Prädikat p_j' korrespondiert mit p_j
- Konstanten: $\phi(s_i') = \phi(s_i)$

Rivests Anforderungen

Effizienz

- ϕ und ϕ^{-1} einfach berechenbar
- f_i' und p_i' im Rechensystem effizient berechenbar
- Platzbedarf verschlüsselter Daten gering

Sicherheit

Kenntnis

- von $\phi^{-1}(d_i)$ für viele Datenpunkte
- von d_i und $\phi^{-1}(d_i)$ für einige Werte
- der Operationen und Prädikate des Rechensystems

nicht ausreichend um ϕ zu bestimmen

Verwirklichung I

Yao 1986

- funktionierendes Schema
- mehrere Interaktionen
- sehr rechenintensiv

Gentry 2009

- größte Errungenschaft der Kryptographie in 2009
- keine Interaktionen
- arbeitet auf algebraischen Verbände
- Konzept schwer verständlich
- rechenintensiv

Verwirklichung II

Smart und Vercauteren, 2010

- „Cloud braucht keine Verbände“
- Spezialfall von Gentrys Schema
- Polynome und Ganzzahlen anstatt Matrizen
- einfacher zu implementieren
- weniger rechenintensiv

Ausführungszeit

Schema nach Smart und Vercauteren

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

n für das Maß an Sicherheit verantwortlich
Schema verwendet Polynome vom Grad 2^n

Rivests Anforderungen – aktueller Stand I

Effizienz

- ✗✓ ϕ und ϕ^{-1} einfach berechenbar
  - Schlüsselerstellung sehr rechenaufwändig
  - mehrere Stunden für kleine Schlüssel
  - Verschlüsselung rechenaufwändig
- ✗✓ f_i' und p_i' im Rechensystem effizient berechenbar
  - ms vs. wenige ns für einfache Operationen
- ✓ Platzbedarf verschlüsselter Daten gering

Rivests Anforderungen – aktueller Stand II

Sicherheit

✓ Kenntnis

- von $\phi^{-1}(d_i)$ für viele Datenpunkte
- von d_i und $\phi^{-1}(d_i)$ für einige Werte
- der Operationen und Prädikate des Rechensystems

nicht ausreichend um ϕ zu bestimmen

$\Rightarrow$ **nicht ausreichend** für die Cloud

Inhalt

1 Angriffsvektoren

2 Theoretische Ansätze

3 Praktische Ansätze

4 Schwachpunkte der praktischen Ansätze

Konzepte

Praktische Ansätze

- **Ansprüche senken**
  Schränke Sicherheits- und Vertraulichkeitsbegriff ein
- **bereits Vorhandenes nutzen**
  Nutze kryptographische Hardware
- **Bedienung vereinfachen**
  Unterstütze Benutzer bei der Verwaltung seiner Privatsphäre

Einschränkung des Vertraulichkeitsbegriffs

Grundprinzip der Cloud

- **Effizienz** hat oberste Priorität
- wichtiger als { Vertraulichkeit / Sicherheit

Verschleiern anstatt Verschlüsseln

- Zugriff billig, wenn Daten genau identifiziert
- sonst Zugriff teuer
- Offenlegung **einzelnen** Datums akzeptabel

Trusted Platform Module (TPM) I

Trusted Platform Module (TPM)

- in jedem neueren Rechner enthalten
- kryptographische Operationen
  - private/öffentliche Verschlüsselung
  - Signierung
  - Schutz der Schlüssel
  - Bescheinigung über Identität

Grundlage des TPM: Indossament-Schlüssel

- Schlüssel in Hardware fest verdrahtet
- identifiziert System

Trusted Platform Module (TPM) II

Platform Configuration Register

- bewahrt Hashwerte der Messungen auf
- nach TPM-Initialisierung: Wert „0“
- einzige schreibende Operation: `extend`
  $PCR_{neu} := \text{SHA-1}(PCR_{alt} || M)$

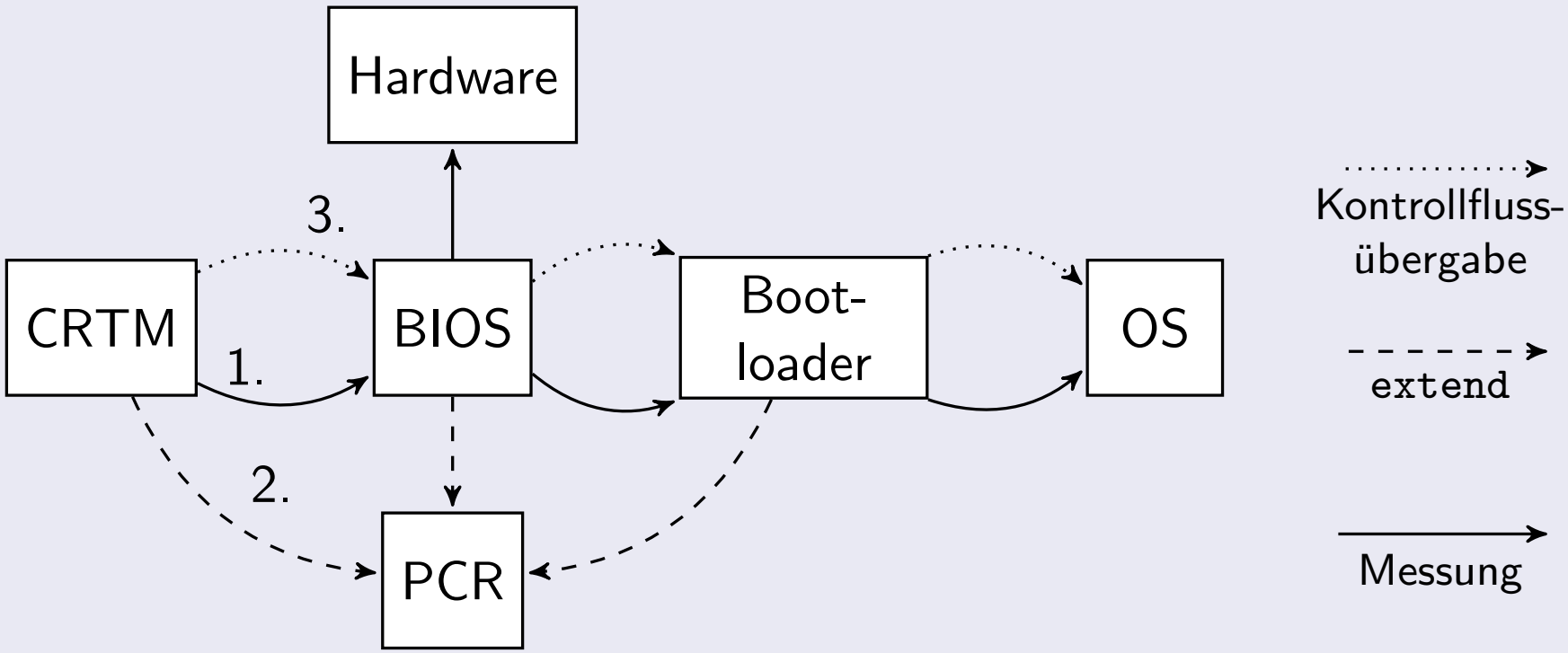
Ablauf einer Authentisierung

- Partner schickt Zufallszahl
- System verschlüsselt PCR-Inhalt und Zufallszahl

$\Rightarrow$ Angriff durch Wiedereinspielen ausgeschlossen

Bescheinigung der Systemidentität

Erstellung der Kette des Vertrauens[1]



CRTM Core Root of Trust Measurement

BIOS Basic Input Output System

PCR Platform Configuration Register

OS Betriebssystem

[1] nach Dinh et al.: *Trusted Computing: TCG proposals*. 2006.

Trusted Cloud Computing Platform

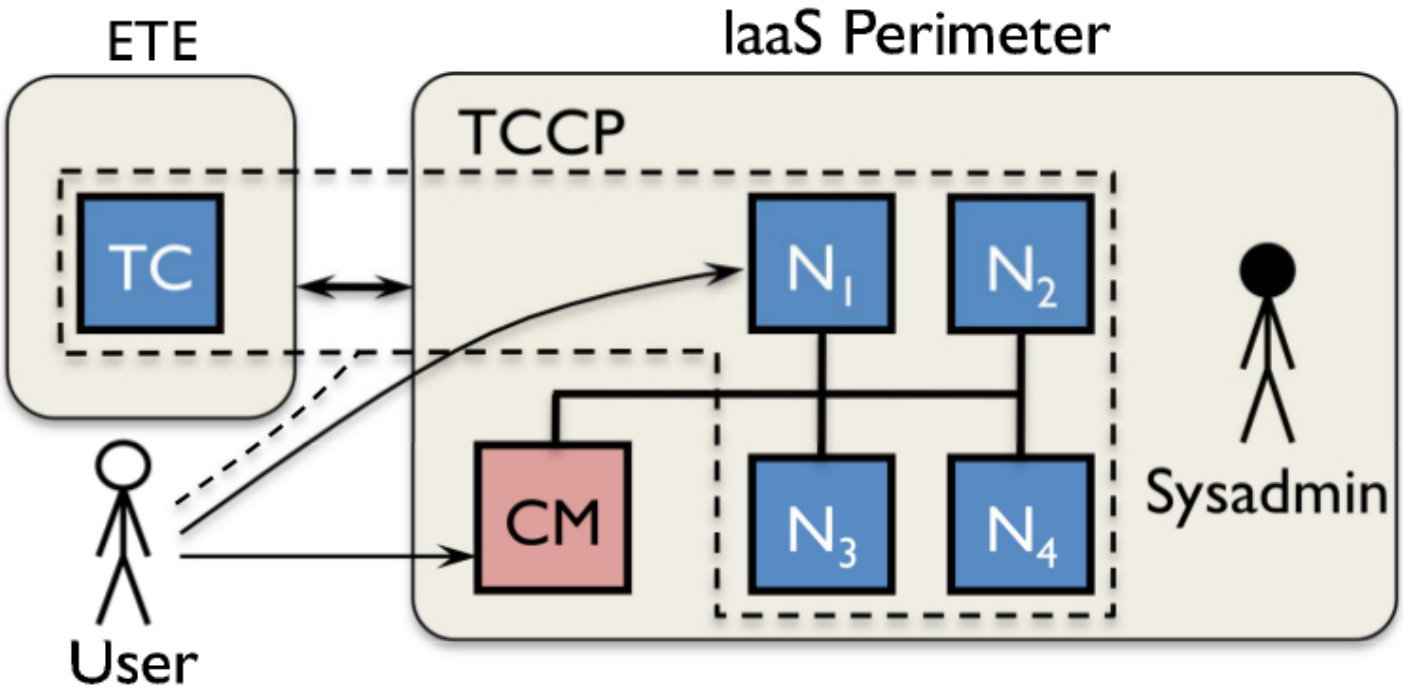
bisher

- Menge von **Knoten** N_i
- nicht vertrauenswürdiger **Cloud Manager (CM)**

Erweiterung

- **Trusted Coordinator (TC)**
  - Betrieb durch vertrauenswürdigen Dritten
  - verwaltet Liste vertrauenswürdiger Knoten
- **Trusted Virtual Machine Monitor**
  - läuft auf Knoten
  - Vertrauenswürdigkeit durch TPM sichergestellt
  - schützt vor Einsichtnahme/Manipulation
- **Virtuelle Maschine** verschlüsselt und signiert, wenn in Transit

Trusted Cloud Computing Platform



TC Trusted Coordinator

CM Cloud Manager

N_i Knoten

TCCP Trusted Cloud Computing Platform

Quelle: Santos et al.: *Towards Trusted Cloud Computing*. In: HOTCLOUD, 2009

Privacy Manager

Motivation

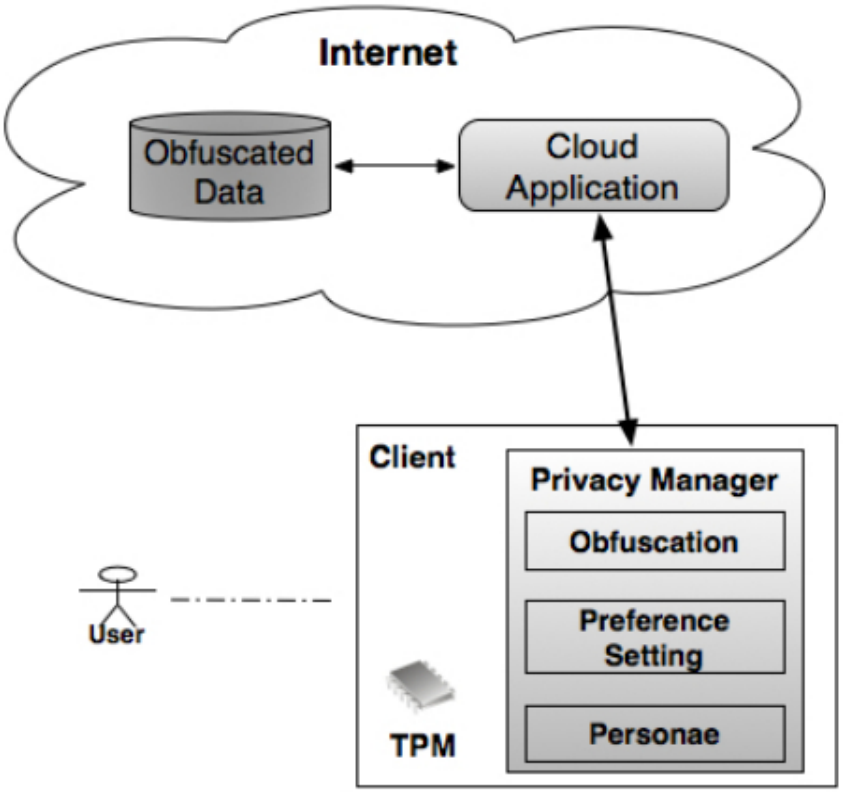
- vielfältige private Daten
- Verwaltung kompliziert

$\Rightarrow$ unterstütze Benutzer

Idee

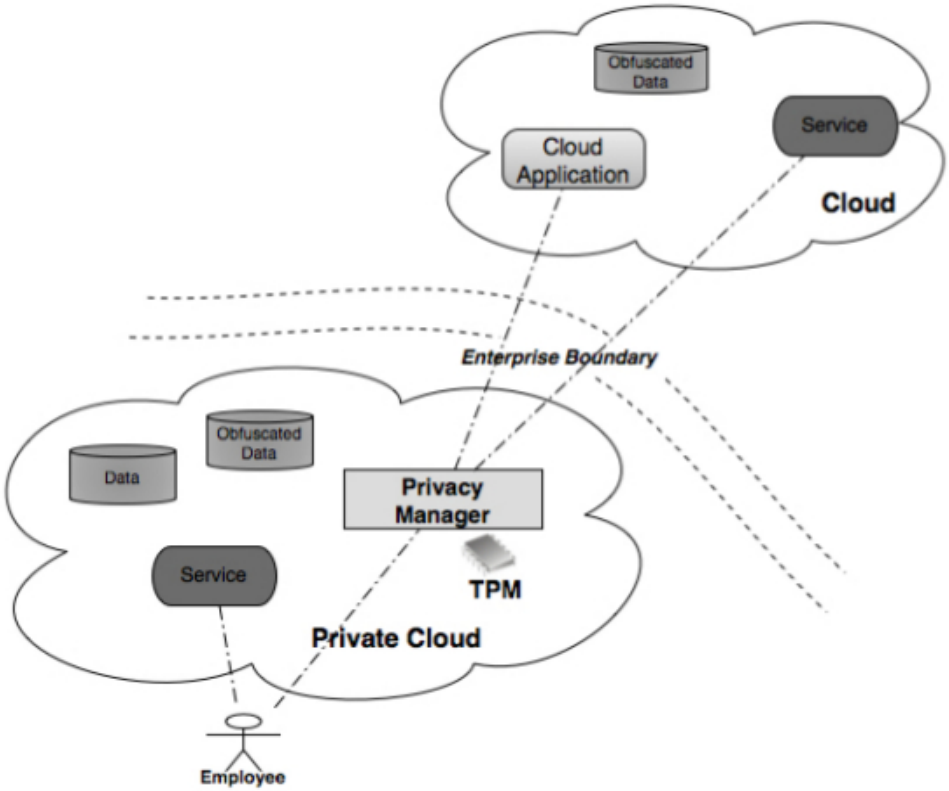
- **Voraussetzung**: funktionierendes Verschleierungsschema
- **Privacy Manager**: Ver- und Entschleierungsdienst
  - **Aspekt**: feingranularer Bestandteil der Privatsphäre
  - **Rolle**: fasst mehrere Aspekte zusammen

Platzierung des Privacy Managers I



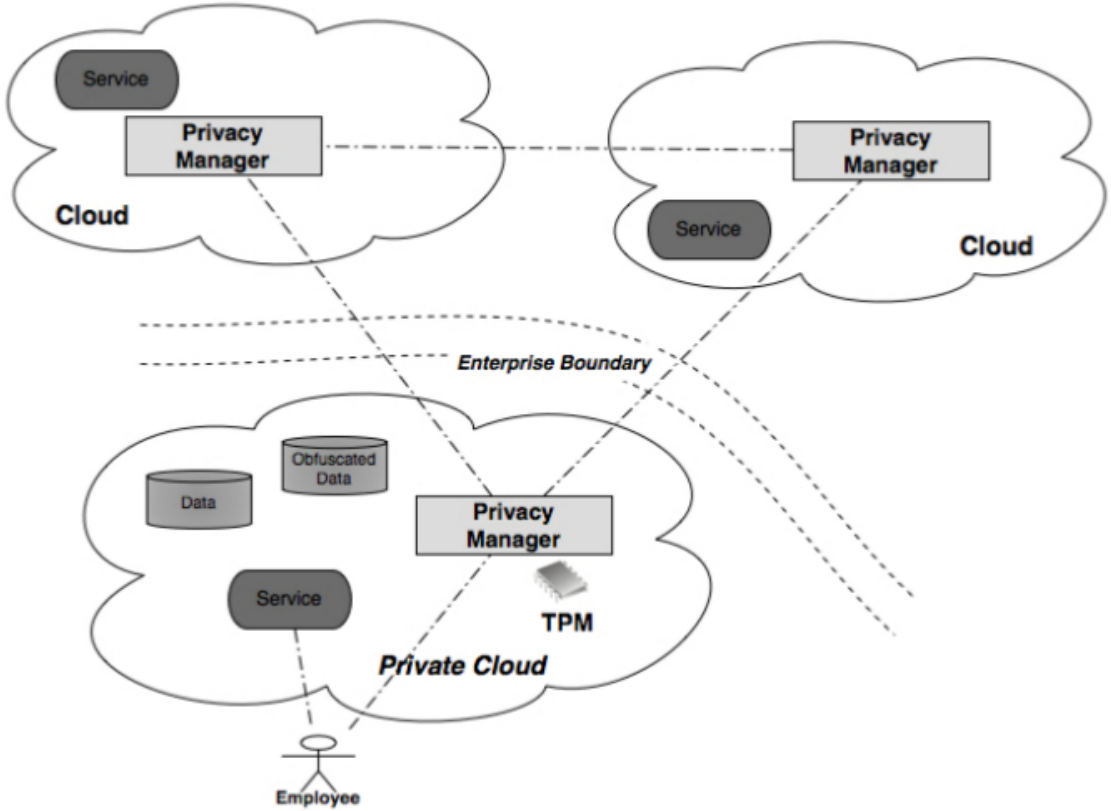
Quelle: Pearson et al.: *A Privacy Manager for Cloud Computing*. In: Proceedings of CloudCom '09, 2009

Platzierung des Privacy Managers II



Quelle: Pearson et al.: *A Privacy Manager for Cloud Computing*. In: Proceedings of CloudCom '09, 2009

Platzierung des Privacy Managers III



Quelle: Pearson et al.: *A Privacy Manager for Cloud Computing*. In: Proceedings of CloudCom '09, 2009

Inhalt

1 Angriffsvektoren

2 Theoretische Ansätze

3 Praktische Ansätze

4 Schwachpunkte der praktischen Ansätze

Exkurs: Kryptographische Hashfunktion

Anforderungen

- $f : S \mapsto S'$, wobei $|S| \gg |S'|$
- mit geringem Aufwand zu berechnen

in NP zu berechnen: $\begin{cases} \text{Schluss von Bild auf Urbild} \\ \text{Finden zweiten Urbilds, das auf Bild abgebildet wird} \end{cases}$

Trusted Platform Module: SHA-1

- Kollision zu finden sollte im Mittel 2^{80} Versuche brauchen
- Wang et al. 2005[1]: $< 2^{69}$ Versuche

[1] Wang et al: Finding Collisions in the Full SHA-1. Springer, 2005

Codeumfang von Betriebssystemen

Codeumfang

- Linux-2.6.32-Kernel: $\approx 12{,}6 \cdot 10^6$ Zeilen Code
- komplettes Windows XP: $\approx 40 \cdot 10^6$ Zeilen Code

Korrektheit des Codes

- Systematisches Korrekturlesen?
- Korrektheitsbeweis?

$\Rightarrow$ Bescheinigung über Betriebssystem **nicht glaubwürdig**

Angriff gegen die Software

Konzept des TPM

- TPM berechnet Prüfsumme beim Laden von Code
- nachträgliche Veränderung des Codes wird nicht bemerkt
- unproblematisch, weil Linux Manipulation am Text-Segment verhindert

Angriff

- schreibe Kernelmodul, das Seitenkacheltabelle „korrigiert“
  $\Rightarrow$ Text-Segment beschreibbar
- Angriff funktioniert auch unter Windows

Angriff durch Zurücksetzen

Low-Pin-Count-Bus

- BIOS, TPM und ältere Ein-/Ausgabegeräte
- Initialisierungsleitung

Angriff

- Hardwaremodifikation
- trenne Initialisierungsleitung von TPM
- initialisiere TPM separat

Angriff gegen den Bootloader

LILO

- *Real Mode Setup Code* Teil des Kerns
- Real Mode Setup Code lädt Rest vom Kern
- LILO lädt Real Mode Setup Code, ohne diesen zu hashen

GRUB

lädt Kernel zwei Mal

- einmal zur Prüfsummenberechnung
- das zweite Mal zum Ausführen

TrustedGRUB

- hasht sich nicht selbst nach dem Laden von Festplatte

Bernhard Kauer: OSLO: Improving the security of Trusted Computing. In the 16th USENIX Security Symposium, 2007

Angriff durch Zeitmessung

Angriff gegen RSA-Implementierung

- Brumley und Boneh[1]:
  Chinesischer Restesatz verwundbar
- hochauflösende Zeitmessung kryptographischer Operationen
- Aufruf mit vorbereiteten Zeichenketten

$\Rightarrow$ privaten Schlüssel in ca. 40 Tagen erraten

[1] Brumley et al.: *Remote timing attacks are practical*
. In proceedings of the 12th Usenix Security Symposium, 2003

Angriff gegen das BIOS

BIOS

- erster Code, der geladen wird
- überschreibbar für Updates

Angriff

- ändere ein Bit

$\Rightarrow$ PCRs werden nicht mehr geschrieben beim Booten

$\Rightarrow$ aufgezeichnete Bus-Transaktionen später einspielen

Abhilfe: Dynamic Root of Trust Measurement

Idee

- PCR 17: wird mit „-1“ initialisiert
- führe Messung spät per `skinit`-Anweisung aus

$\Rightarrow$ Angriff durch Zurücksetzen verhindert

`skinit` (atomar)

1. initialisiere Prozessor
2. setze PCR 17 = 0
3. lade *Secure Loader* in Prozessorcache
4. hänge Hash des Secure Loader an PCR 17 an
5. führe Secure Loader aus

Zusammenfassung TPM

Sicherheit

- ist ein Konzept
- erreicht man **nicht**, indem man
  - eine Personal Firewall installiert
  - einen „Sicherheits-Chip“ in seinem System hat
- eine einzelne Lücke im Konzept bringt dieses zu Fall
- Entwurf eines Sicherheitskonzepts erfordert Fachkenntnis

TPM

- hat mehrere Lücken
- ist kein Allheilmittel

Fazit

Motivation

- Reputation
- Rechtslage

Angriffsvektoren

- Mitarbeiter
- Mitbenutzer
- Staat

Lösungsansätze

- vollständig homomorphe Verschlüsselung $\Rightarrow$ Forschung
- Trusted Computing $\Rightarrow$ bedingt geeignet

Fragen?