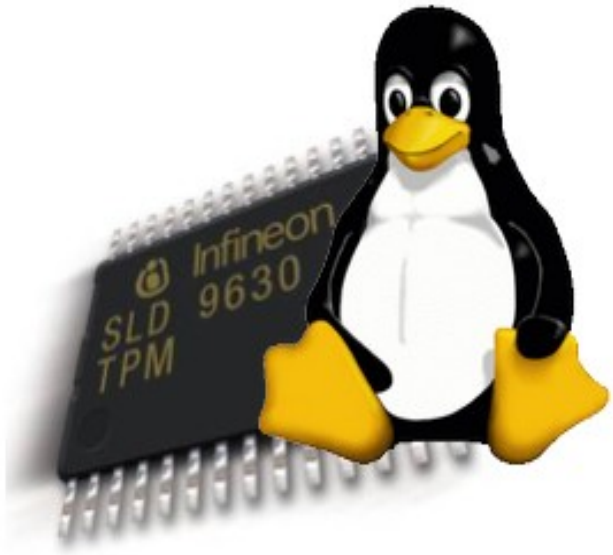
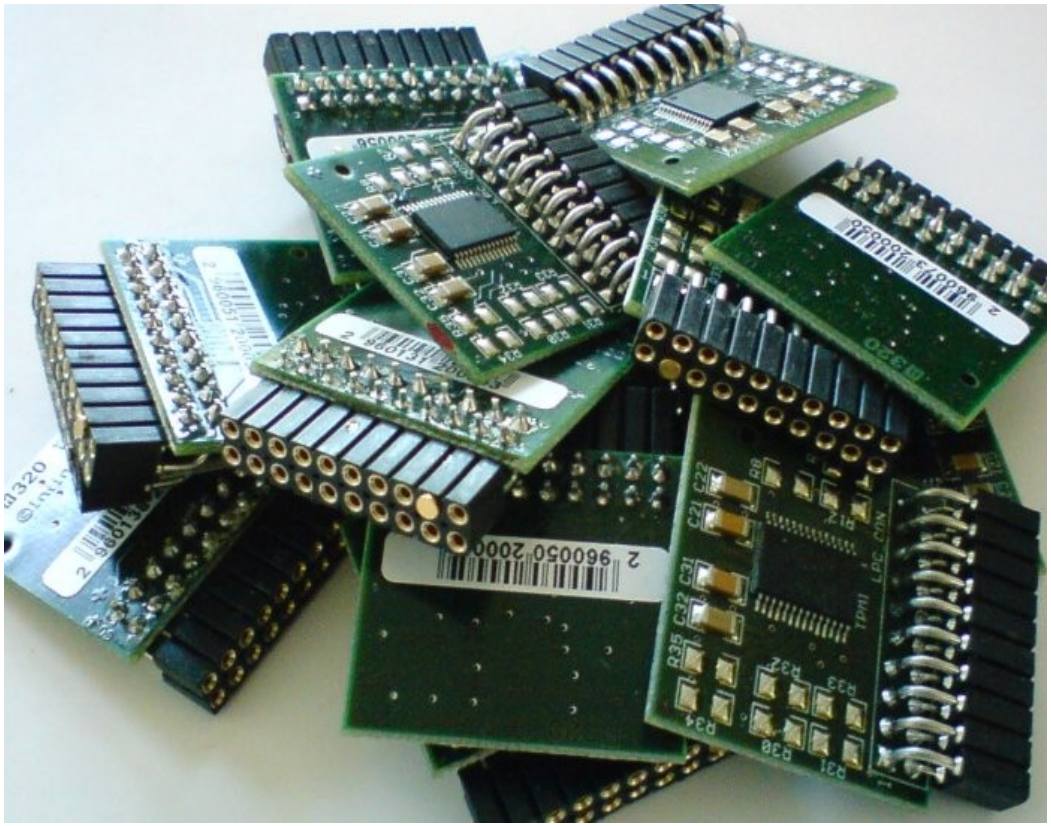
Neues aus Trusted Computing Land



DI Martin Pirker <martin.pirker@iaik.tugraz.at>

Übersicht

- Einführung und Motivation
- TPM Chip
  - Hardware
  - Funktionen
- TC Konzepte
  - Chain of Trust
  - Attestierung
  - Identitäten
- Software
  - Treiber, Middleware, Anwendungen
- Virtualisierung und Trusted Computing



Einführung (1/2)

- Medien voll mit Meldungen über Sicherheitsprobleme
- Grundproblem:
  - Einschätzung der Sicherheit - wem vertrauen?
  - Hardware? Betriebssystem? Applikation? User?
- derzeitiges Modell:
  - Gott gleicher Superuser / Admin am lokalen Rechner.
  - Erwartungshaltung: alle Entscheidungen betreffend Sicherheit (OS patches, Rechte und Rollen der Applikationen, Policies, Resource monitoring, ...) werden optimalst von ihm getroffen... Realität?
  - Kommunikationspartner im Informationsnachteil, keine Anhaltspunkte über wirklichen Systemzustand

Einführung (2/2)

- ständig neue Initiativen um Sicherheit zu verbessern
- ein Ansatz: "Trusted Computing",
  von der Trusted Computing Group (TCG)
- Hardware basierender Vertrauensanker:
  Trusted Platform Module (TPM)
- Bildung einer Vertrauenskette über mehrere Schichten
- Kryptographisches Grundgerüst auf jedem PC
- ermöglicht Auskunft über Systemzustand mit
  verbesserter Manipulationssicherheit
- Vertrauen in Computer erhöhen

Trusted Computing - Organisation

- ursprünglich: TCPA – Trusted Computing Platform Alliance
- jetzt: TCG – Trusted Computing Group (https://www.trustedcomputinggroup.org/)
- Führungspositionen (Promotors):
  - AMD, HP, IBM, Infineon, Intel, Lenovo, Microsoft, Sun
- Abstufungen der Mitgliedschaft bis hin zu akademischen Beobachtern (z.B. IAIK)
- Non-Profit Organisation
- Entwicklung globaler offener Industriestandards für Trusted Computing Hardware, Softwarebibliotheken und Protokollen
- Arbeitsgruppen zu Themenbereichen (HW, SW, Mobile, ...)

TPM Chip - Hardware

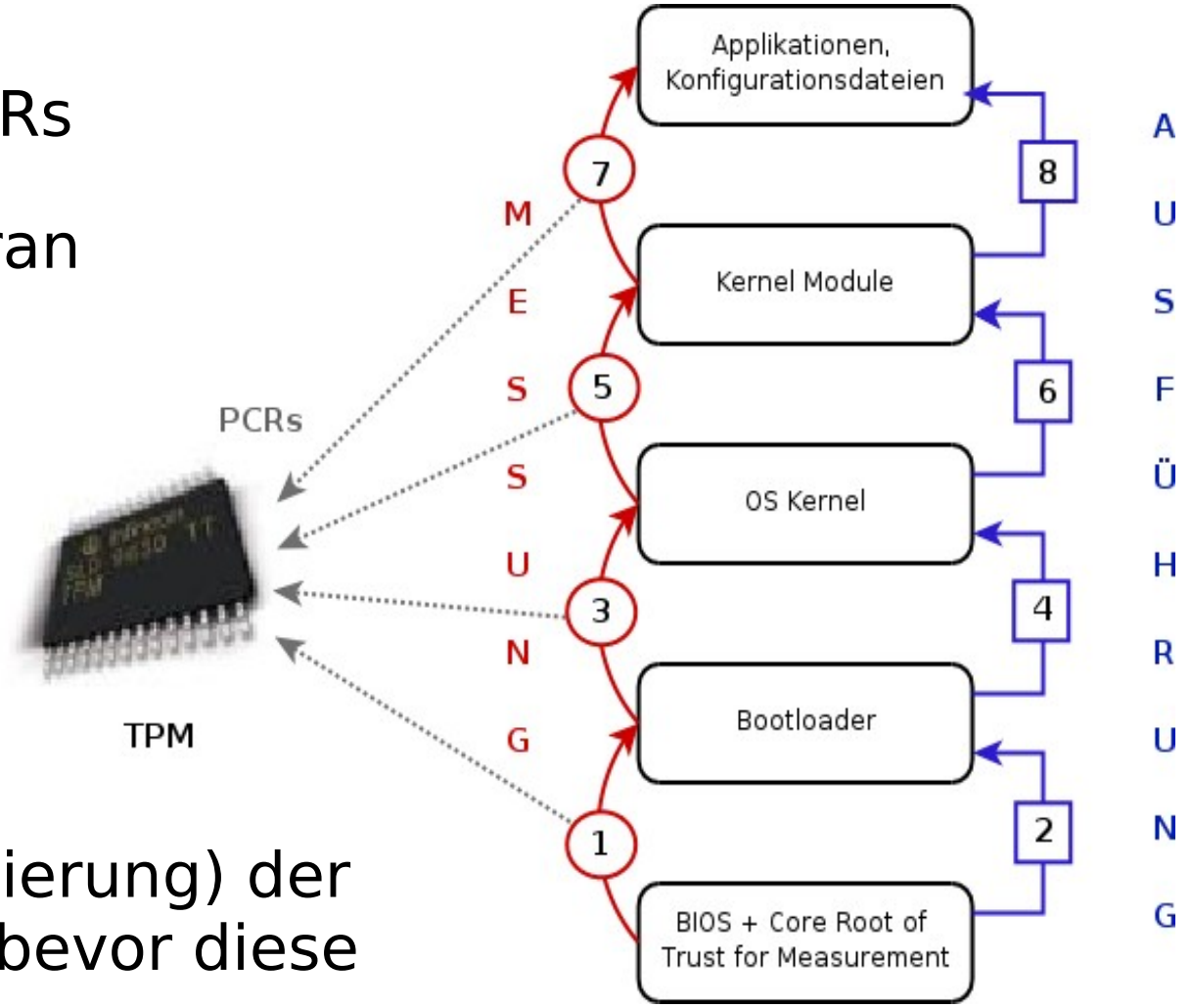
- TPM – Trusted Platform Module
- Hardware basierender Vertrauensanker
- weitaus schwerer manipulierbar als Software
- Low-Cost Device, kein(!) Kryptographie-Beschleuniger
- passives Bauteil, d.h. ausschließlich Befehlsempfänger
- auf neuen Motherboards aufgelötet – d.h. TPM ist fixer Bestandteil der Plattform und kann nicht entfernt werden
- aktuelle Version: 1.2 (1.1b Auslaufmodell)
- für viele Benutzer ist Sinn und Zweck nach wie vor unklar
- oftmals als “Panikreaktion” Deaktivierung des TPM im BIOS
- besser: verstehen der TPM Funktionen und Hintergründe

TPM Chip – Funktionen

- Opt-In: Nutzer bestimmt ob er das TPM aktiviert (BIOS)
- (wichtigste) Funktionseinheiten:
  - RSA
    - Schlüsselerzeugung
    - Ent- und Verschlüsselung
    - Datenbindung an Systemzustand (PCRs)
  - 24 PCRs – Platform Configuration Registers (Plattformzustand)
  - SHA1 Hash Funktion
  - Zufallszahlen Generator
  - Erkennung von Hardware Manipulation
  - temporärer Zwischenspeicher für RSA Schlüssel
  - permanenter Speicher für spezielle Schlüssel

Chain of Trust / Trusted Boot

- Ziel: Abbildung des Systemzustandes in PCRs
- Benutzer wird nicht daran gehindert bestimmte Software auszuführen
- Ausführung wird aber protokolliert und kann im nachhinein nicht geleugnet werden
- Messung (und Protokollierung) der nächsten Komponente bevor diese die Kontrolle bekommt

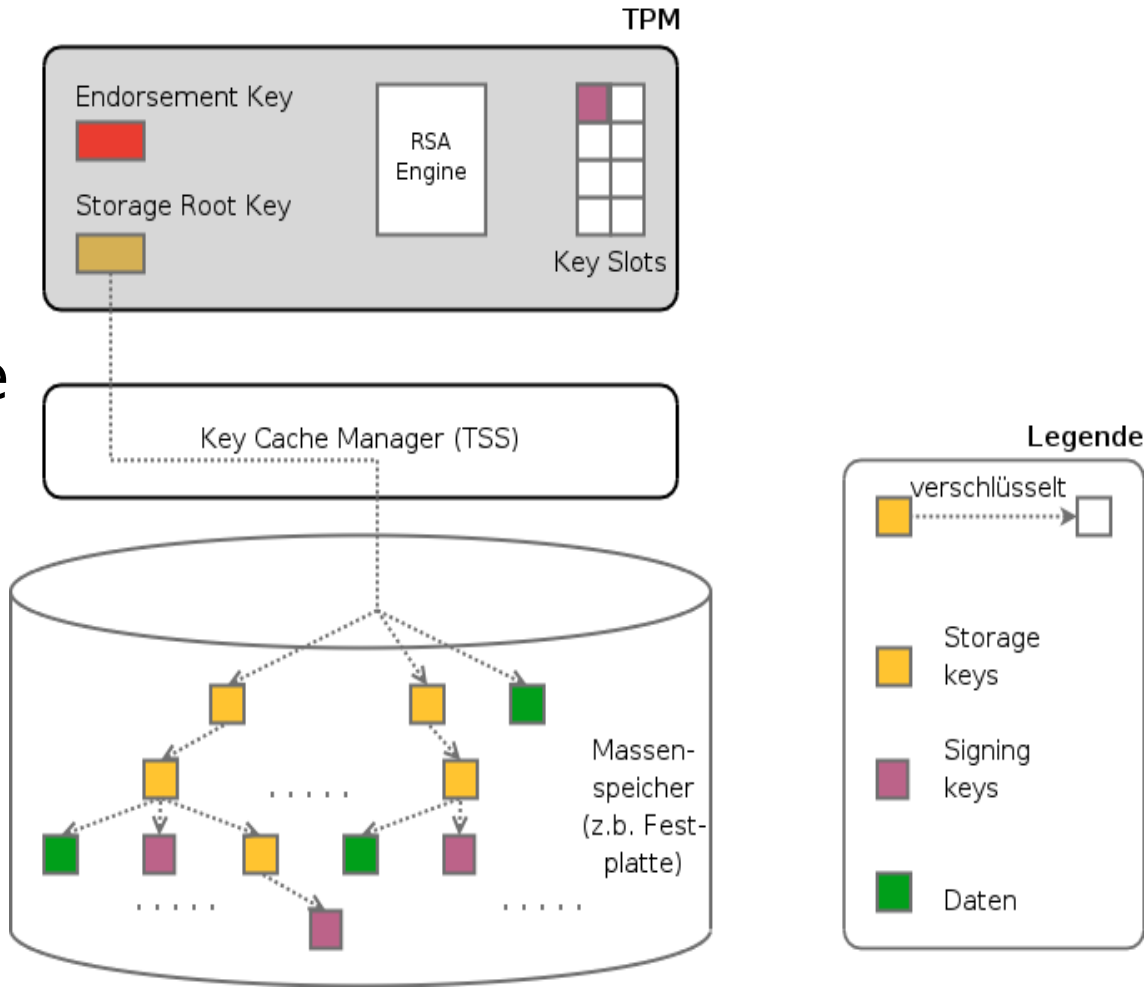


Kryptographische Schlüssel

- Endorsement Key (EK)
  - untrennbar mit dem TPM verbunden (~Identitätsschlüssel), EK Erzeugung ist i.d.R. Teil des Herstellungsprozesses
- Storage Root Key (SRK)
  - steht an der Spitze einer Schlüssel-Hierarchie
- Storage Keys und Signing Keys
- migrierbare und nicht migrierbare Schlüssel
- TPM hat nur beschränkten internen Speicher
  - Keys außer EK und SRK werden nicht dauerhaft im TPM gespeichert, sondern in einem vom TSS verwalteten (Key Cache Manager) externen Speicher
  - die Schlüssel verlassen das TPM nur in verschlüsseltem Zustand

Schlüsselbaum

- 2 fixe Schlüsselspeicher
- dynamischer Speicher
- System Support Software koordiniert swapping der Schlüsselspeicher
- Schlüsselbaum, alle Schlüssel abhängig von einem SRK



Identitäten

- Endorsement Key Problematik
  - identifiziert das TPM und damit oft zugleich den Benutzer
  - bleibt über die gesamte TPM Lebenszeit gleich
- zum Schutz der Privatsphäre: Erzeugung mehrerer, verschiedener "abgeleiteter" Identitäten
- AIK – Attestation Identity Key
  - TPM erzeugtes, nicht migrierbares RSA Schlüsselpaar
  - hat keinen direkten Bezug mehr zum EK
  - ein Benutzer kann "beliebig" viele AIKs haben – damit sind seine Aktivitäten nicht mehr einfach auf ihn zurückführbar

Identitätserzeugung

- Privacy Certificate Authority (Privacy CA)
  - Antragsteller muss mit Zertifikaten (TPM Chip, Plattform) beweisen, dass er eine TPM Plattform besitzt
  - wenn OK: Privacy CA stellt AIK Zertifikat aus
- AIK Zertifikat
  - bescheinigt, dass hinter einem AIK Schlüsselpaar tatsächlich ein TPM steht
  - spätere Nachprüfbarkeit (Langzeitdokument)
- Privacy CA Kritik
  - Vertrauen in Privacy CA, Skalierbarkeit
- Alternative ab TPM 1.2: Direct Anonymous Attestation
  - Zero-Knowledge-Proof basierend

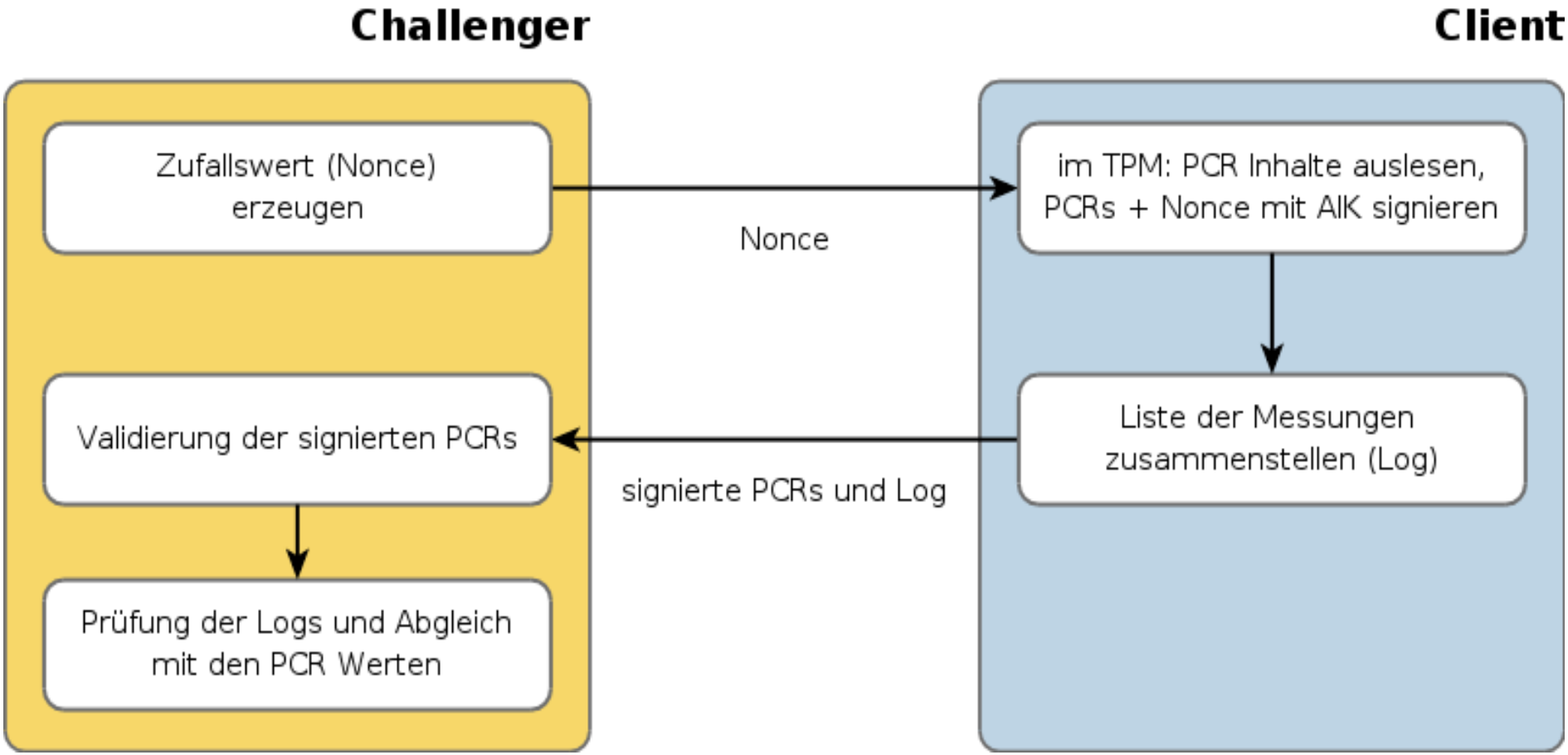
Attestierung (1/3)

- Ziel: eine Gegenstelle (z.B. eine Bank) will Auskunft über den Systemzustand, dass auf dem Client keine unerwünschten Programme laufen (Keylogger, etc.)
- Plattform darf jeden Zustand annehmen (d.h. der Benutzer darf tun und lassen was er will) aber man soll Systemzustand nicht leugnen können (z.B. dass bestimmte Applikation läuft/gelaufen ist)
- PCRs bilden den Zustand der Plattform ab
  PCR[i] <- SHA1(i | PCR[i] | new data)
- PCRs sind im TPM und können nicht manipuliert werden (werden nur bei (Re-)Boot der Plattform zurückgesetzt)

Attestierung (2/3)

	Measurement Value (fingerprint == SHA1)	Measurement Hook	File Name	
#000	9797EDF8D0EED36B1CF92547816051C8AF4E45EE	ima-init	boot-aggregate	Aggregate
#001	F7A0BF5A67CE98BC06316F77CA1F404A2D447534	mmap-file	init	Executable
#002	38C5D31E5DAD3F1B012FDD35B4E011E783CE6FD8	mmap-file	ld-2.3.2.so	Library
#003	42F79603219922016713BB8AAFC9E37F6936B226	mmap-file	libc-2.3.2.so	Library
#004	A4DC5EDF06698646CD76916F16E95C37E55DC12B	mmap-file	bash	Executable
#005	F4F6CB0ACC2F1BEE13D60330011DF926D24E5688	mmap-file	libtermcap.so.2.0.8	Library
#006	AE1BC1746AFD2AC1ECD1D9EEEAEBD125A6A9EB8D	mmap-file	libdl-2.3.2.so	Library
#007	CFBC7EC3302145AB78A307C0D41DBB9A4251377B	mmap-file	libnss_files-2.3.2.so	Library
#008	805572455CF5BF50A7EE42E3CC6B0EDA65AF17A4	mmap-file	initlog	Executable
#009	C95CBC5625719649103E0D1C3595967474842F7B	mmap-file	hostname	Executable
#010	0CAA342424F420FF29B7FB2FCF278F973600681B	mmap-file	mount	Executable
#011	5E45D898530F31BADEF5E247EBCF4AB57A795366	bash-source	functions	Bash Sourc
#012	A253AF3AB981711A13AE45D6B46462386E628076	mmap-file	consoletype	Executable
#013	2E37B839BC4EC1B6BE1BDF5BACD1E7B56567D8D9	bash-source	i18n	Bash Sourc
#014	C9D1B3E2CD0995E16AE6DD98B388FD873324740D	bash-source	init	Bash Sourc
#015	590F75EE97E0FC560F07FCB07A8646FADEC88C2A	mmap-file	uname	Executable
#016	5E851EFA4601B3AFCA9EAE75ED53688606630BFA	mmap-file	grep	Executable
#017	32798F58C4F1B4CD017B09BCAAF2A22D345E7E4F	mmap-file	sed	Executable
#018	CE516DE1DF0CD230F4A1D34EFC89491CAF3D50E4	mmap-file	libpcre.so.0.0.1	Library
#019	22EAF1B6009B23150367F465694AC63314866558	bash-script	setsysfont	Bash Comma
#020	8B15F3556E892176B03D775E590F8ADF9DA727C5	bash-script	unicode_start	Bash Comma
#021	A4C5F9D457DA16E47768423A68F135259F7180D7	mmap-file	kbd_mode	Executable
#022	497ED7F80C33AF25307DFC80970571C51006CE6A	mmap-file	dumpkeys	Executable
#023	04A0599405EBD306CEF2447679C8F4B5159A55C7	mmap-file	loadkeys	Executable
#024	AE327AD27D02BF2DE96557A1B4053D02129B1394	mmap-file	setfont	Executable
#025	7334B75FDF47213FF94708D2862978D0FF36D682	mmap-file	gzip	Executable
#026	93D65AB85CF5EE1ACD9E6BE5057D622D80AB5E10	mmap-file	dmesg	Executable
#027	B6E90C3A25B69C3B1D3B643DB7D9504FBC36C1D1	mmap-file	minilogd	Executable

Attestierung (3/3)



Zertifikate + PKI

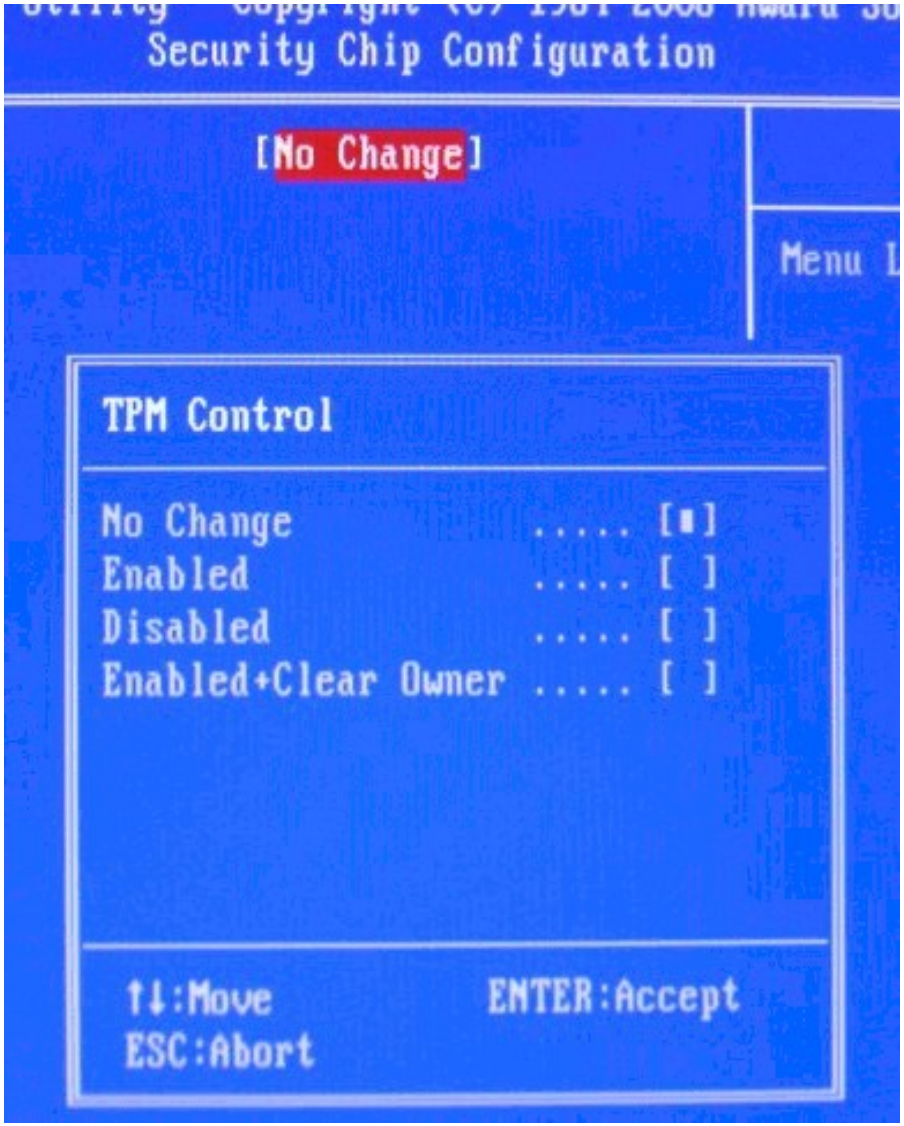
- Zertifikat = Datenblock mit
  - Aussteller, Inhaber, public key, Gültigkeitsdauer, Verwendung, Signatur, ...
- TC spezifische Zertifikatserweiterungen
  - für TPMs
  - für Plattformen
  - für Identitäten
- Problem: Zertifikat alleine ist nur eine „Behauptung“
  - Gültigkeitsüberprüfung benötigt Infrastruktur („PKI“)
  - TC PKI (noch) nicht existent

Software - Schichten

- ein TPM benötigt mehrere Software Komponenten um sinnvoll genutzt werden zu können
- bereits heute sind mehrere Open Source Komponenten für Trusted Computing verfügbar
- treibende Kräfte:
  Universitäre Forschung und einige Firmen (z.B. IBM)

BIOS

- Trusted Computing unterstützendes BIOS
  - Kontrollfunktionen für Nutzer
    - benötigt meist "physical presence" (Taste halten bei Boot oder Jumper am Mainboard)
    - TPM (De)Aktivierung
    - TPM forced clear
  - Core Root of Trust for Measurement (CRTM)
    - Messung Bootloader
    - TPM Servicefunktionen für nächste Stufen



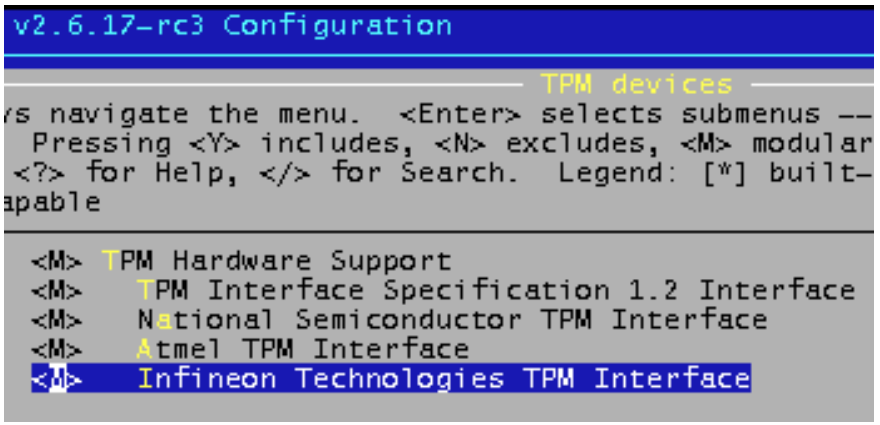
Bootloader

- Trusted Grub
  - modifizierte GRUB Version um Chain of Trust zu realisieren
  - stützt sich auf BIOS um TPM anzusprechen
  - misst nächste Stufe (Kernel) in PCR
  - http://sourceforge.net/projects/trustedgrub
- OSLO (Open Secure LOader)
  - unterstützt SKINIT Instruktion neuerer AMD64 Prozessoren
  - http://os.inf.tu-dresden.de/~kauer/oslo/

Kernel TPM Treiber

- in aktuellen 2.6er Kernels enthalten:
  - Infineon
  - Atmel
  - National Semiconductors
  - TIS Treiber – generischer Treiber für 1.2 TPMs (ab Ver. 2.6.17)

```
*************************************************
*    TPM found at 0x4700 (Intel ICH4 LPC)       *
*    Chip    ID 0006                            *
*    Vendor  ID 15d1 (Infineon)                 *
*    Product ID 0006 (SLD 9630 TT 1.1)          *
*************************************************
```



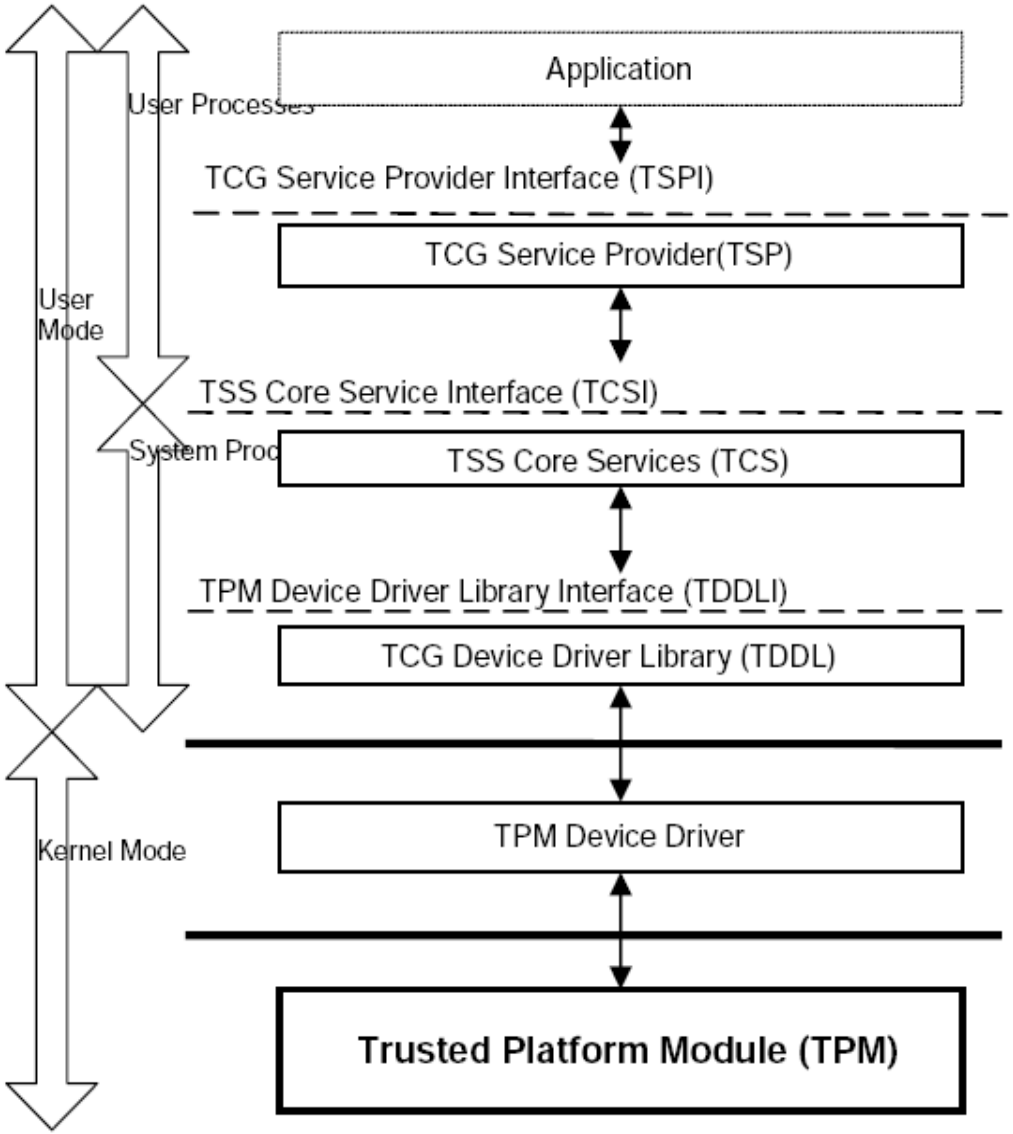
- Alternative wenn man kein TPM hat: TPM Emulator
  - http://tpm-emulator.berlios.de
  - nicht fehlerfrei, aber genug Funktionalität (1.1b) sodass man selbst experimentieren und testen kann

Software - Middleware

- libTpm
  - direkter Zugriff auf /dev/tpm
  - erlaubt direkt Befehle (Byte Sequenzen) an das TPM zu schicken
- TrouSerS – TCG Software Stack (TSS) von IBM
  - http://trousers.sourceforge.net/
  - implementiert offizielle C API nach TSS Spec 1.1b
  - 1.2 branch existiert (http://trousers.sourceforge.net/trousers12support.html)
  - noch mit „Kinderkrankheiten“
  - sowohl mit TPM Emulator als auch Hardware TPM verwendbar

TCG Software Stack

- Applikationen verwenden oberste TSS Schicht (TSPI) für Zugriff auf TSS
- API auf C Ebene von TCG spezifiziert
- TCS stellt TPM Funktionalität als Service (Daemon) bereit
- verwaltet parallelen Zugriff durch Applikationen auf das TPM
- TPM Treiber (Kernel)



TCG Specification: Architecture Overview, p. 26

Infineon + Linux

- TSS von Infineon für Linux
  - neuester Spec (1.2) entsprechend
  - interne SOAP Kommunikation zwischen TCS und TSP
  - erstes Release als .rpm für OpenSuse via OpenTC
- Support Software
  - TPM Management GUI
  - TSS backup / archive / recovery tools
  - Identity Management
- coming soon ...lassen wir uns überraschen :-)

Java Software

- Trusted Java (http://trustedjava.sourceforge.net/)
- Projekt um Trusted Computing Funktionalitäten in Java verfügbar zu machen
- Hauptkomponenten
  - jTSS
    - kompletter TSS Stack, geschrieben in 100% Java (derzeit Linux und Windows Vista kompatibel)
  - jTSS Wrapper
    - gleiches TSS Stack API, per JNI auf TrouSerS aufbauend
  - TCcert
    - Bibliothek zur Verarbeitung TC spezifischer Zertifikate
  - JTpmTools
    - Demonstrationscode zur Verwendung der Einzelteile

Software - Anwendungen

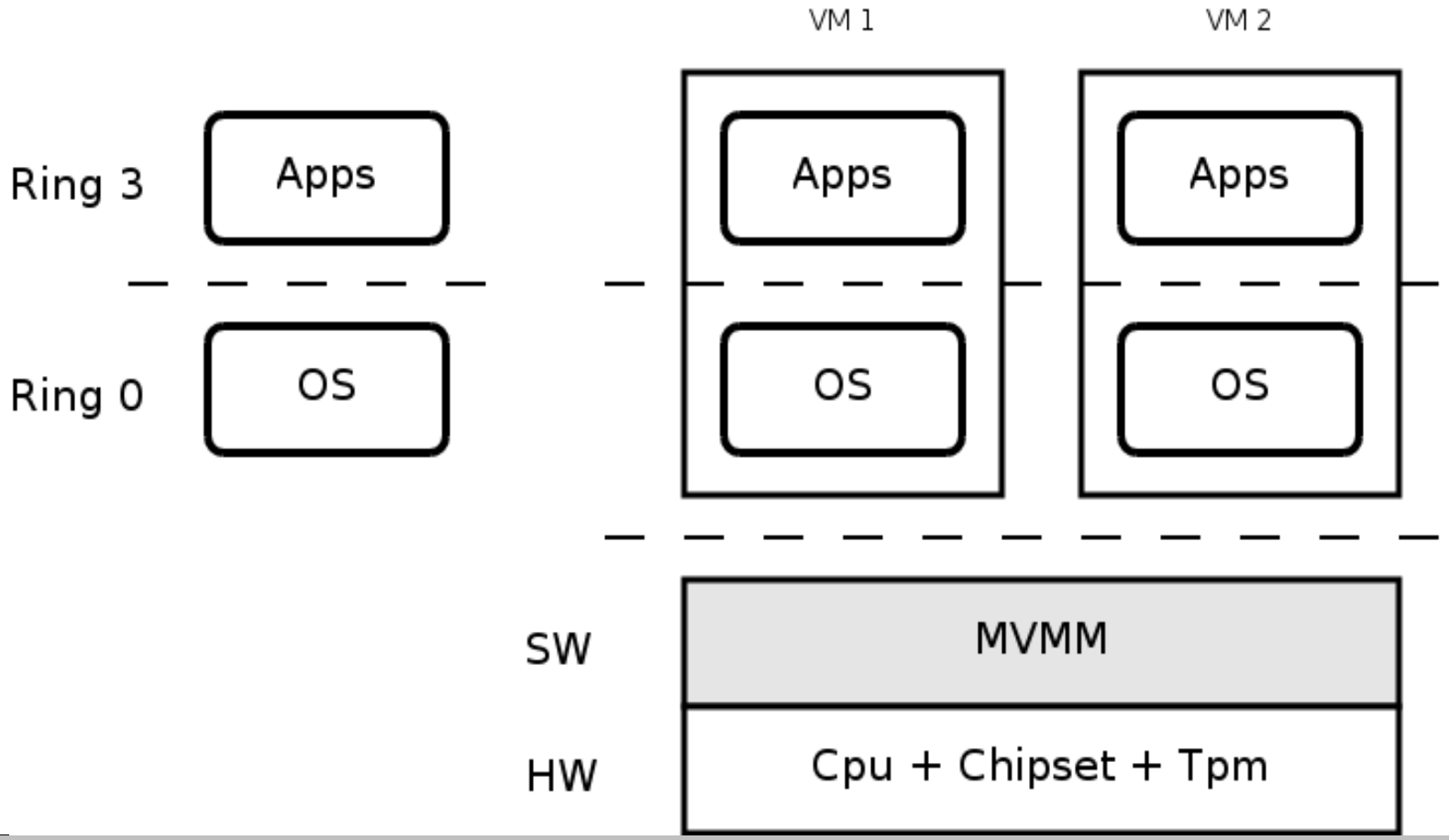
- Beispielanwendungen (von TrouSerS Homepage verlinkt)
  - OpenSSL TPM Engine
    - TPM Schlüssel in Zertifikate
    - TPM Schlüssel für SSL Verbindungen
  - TPM keyring
    - Schutz "normaler" Schlüssel mit TPM verankertem Hauptschlüssel
  - PKCS#11 Anbindung (openCryptoki)
  - IBMs Integrity Measurement Architecture (IMA)
    - kernel patch
    - http://www.research.ibm.com/ssd_ima/

Virtualisierung

- Messung eines gesamten "General Purpose" OS ist zu aufwändig
- Idee: mehrere spezialisierte, kleine Systeme, die auf einer Virtualisierungs-Schicht („Hypervisor“) laufen
- "compartments", "partitions"
- leichter messbar
- sichere und unsichere Systeme gleichzeitig nebeneinander, Abschottung durchgesetzt von Hypervisor
- Ansätze basierend auf XEN und dem L4 Microkernel

Virtualisierung

- neueste CPU+Chipset Generation ("LaGrande", "Pacifica",...) bietet Hardware Unterstützung für Virtualisierung



Virtualisierung

- bisher:
  - CPU Ringe 0-3
- neu:
  - Einführung eines "Superring", unter Ring 0
  - Möglichkeit auch privilegierte Instruktionen aus Ring 0 zu kontrollieren bzw. modifizieren
  - Hypervisor teilt Resourcen für VMs zu
  - Hypervisor unantastbar (No-DMA Memory etc.)
  - Hypervisor ist für Ring 0 transparent
    - Ausführung beliebiger OS ohne Modifikation möglich

Virtualisierung

- SKINIT / SENTER
  - dynamic root of trust for measurement (DRTM)
  - beliebig "späte" Umschaltung Prozessors in ein "trusted" System
  - 64kb Secure Loader (SL) initialisiert System, Messung System/Loader nach PCR17
  - Ausführung MVMM / Hypervisor
- MVMM = "measured virtual machine monitor"
  - Speicherzuteilung
  - VM Verwaltung
  - Kommunikationskanäle

Forschungsgebiete

- Was soll man messen um „Trust“ zu quantifizieren?
  - Hash über Programmimage einfach.
    - Problem: xy Programmversionen/kombinationen
    - Logkomplexität
  - Verhalten zur Laufzeit bestimmbar?
    - z.B. Java VM: Programm "gut" solange bestimmte Klassen nicht verwendet werden?
  - Abschottung von Programm, genaue Inspektion der Kommunikation
- Vergleichswerte
  - kann man lokalen Vergleichswerten je trauen?

Forschungsgebiete

- Reduktion Komplexität Hypervisor
  - formale Analyse des Datenflußes, Aussage über Verhalten/Sicherheit möglich?
- TPM Virtualisierung / Multiplexing
  - PCR Registersatz swap bei VM switch
  - Vertrauenskette Software vTPM in VM bis zum Hardware TPM
- TPM Zukunft
  - SHA1 ersetzen durch ... SHA256?
  - RSA ersetzen durch ... Eliptische Kurven?
- Konflikt Privacy vs. Identität Kommunikationspartner
  - Zero-Knowledge-Proof Konzepte

OpenTC Projekt

- EU Projekt OpenTC
- Ziel: Schaffung einer offenen Trusted Computing Infrastruktur auf Linux Basis
- Vorteil: Jeder kann die Technologie studieren und sich eine eigene Meinung über Trusted Computing bilden
- über OpenTC:
  - http://www.opentc.net
  - europaweit über 20 Projektpartner (Firmen und Unis)
  - 3,5 Jahre Laufzeit
  - offene Auseinandersetzung mit Trusted Computing

Zusammenfassung

- Trusted Computing kommt (bzw. ist schon da)
- ignorieren ist unproduktiv und hilft nicht weiter
- Auseinandersetzung mit der Technologie
- mit freien Softwarebausteinen Erfahrung sammeln
- Anwendungsszenarien ausloten
- Kontakte knüpfen, Dialog führen
- selbst Code schreiben und veröffentlichen (Sourceforge Projekte...)

... Entwicklung mitgestalten, nicht nur Beifahrer sein!

Weiterführende Literatur

- Trusted Computing Group:
  http://www.trustedcomputinggroup.org

  speziell für Einstieg/Überblick: Architecture Overview
  https://www.trustedcomputinggroup.org/specs/IWG/TCG_1_0_Architecture_Overview.pdf

- Grawrock, David
  "The Intel Safer Computing Initiative"
  Intel Press, 2006

- Google...

Letzte Folie!

- Kontakt:

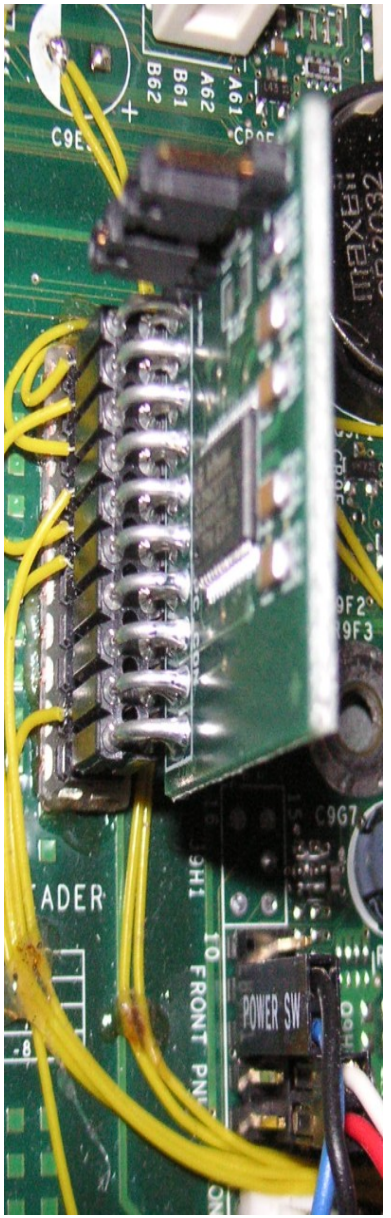
  Trusted Computing @ IAIK TU Graz

  web: http://iaik.tugraz.at/

  mail: iaik_open_tc@iaik.tugraz.at

  TU Graz, LV im SS: "AK IT-Sicherheit 1"

  Projekte / Diplomarbeiten



Vielen Dank für die Aufmerksamkeit!

OpenTC Projekt Details

- The Open_TC project is co-financed by the EC.
  contract no: IST-027635

- If you need further information, please visit our website www.opentc.net or contact the coordinator:

  Technikon Forschungs- und Planungsgesellschaft mbH
  Richard-Wagner-Strasse 7, 9500 Villach, AUSTRIA
  Tel. +43 4242 23355 0
  Fax. +43 4242 23355 77
  Email coordination@opentc.net