Die Technik hinter dem Trusted Computing der TCPA/TCG - Chance oder Bedrohung für Linux?

LinuxTag, Karlsruhe, 23. Juni 2005

Wilhelm Dolle, Director Information Technology
interActive Systems GmbH

Agenda

- Einführung
- Trusted Computing Konzepte der TCG / TCPA
- Funktionen des Trusted Platform Module (TPM)
- Status Quo der Unterstützung durch Hard- und Software
- Chancen und Risiken
- Linux und das TPM
- Fazit

Trusted Computing Platform Alliance (TCPA)

- **1999 von Microsoft, Intel, IBM, Compaq und HP gegründetes Hesteller-konsortium**
- **2003 über 200 Mitglieder (u.a. Infineon, Siemens, RSA, Nokia)**
- **Einstimmige Entscheidungsfindung**
- **Erste Veröffentlichung der Spezifikationen in Version 0.9 im August 2000**
- **Ziel: Hard- und Softwarestandards zu spezifizieren, um der Vertrauen (Trust) in Computerplattformen zu erhöhen**
  - Plattform: Motherbord, CPU, weitere Geräte und Chips
  - Vertrauen: Komponenten agieren so wie erwartet

Trusted Computing Group (TCG)

- **Von AMD, IBM, Intel und Microsoft gegründet**
- **Seit April 2003 Rechtsnachfolger der TCPA**
- **Nicht ganz so basisdemokratisch wie die TCPA**
  - Kein Veto für Mitglieder mehr (2/3 Mehrheit)
  - Verschiedene Mitgliedsstufen
    - Promotor (50.000 $/Jahr)
    - Contributor (15.000 $/Jahr)
    - Adopter (7.500 bzw. für kleine Firmen 1.000 $/Jahr, kein Stimmrecht)
    - Seit Mitte 2004 “Industry Liasison Program” (keine Kosten, kein Stimmrecht, NDA erforderlich)
- **Juni 2005: 7 Promotor, 70 Contributor, 30 Adopter**
- **Ziel: Entwicklung uns Support von offenen Industriestandards für “Trusted Computing” aus verschiedenen Plattformen**
  - Plattform: PCs, Server, Laptops, Mobiltelefone und PDAs

TCG-Architektur

- **Hardware**
  - Trusted Platform Module (TPM)
- **Software**
  - Trusted Software Stack (TSS)
- **Hier wichtige Spezifikationen**
  - TCG TPM Main Specification (alte Version 1.1b)
  - TCG TPM Specification Version 1.2 (November 2003)
  - TCG Software Stack Specification Version 1.1 (September 2003)

Vertrauenswurzeln (Roots of Trust) des TCG-Subsystems

- **Root of Trust for Measurement (RTM)**
  - Bestimmt beim Booten die Integrität einer Konfiguration
  - Wird vom TPM (Hash-Funktion, PCRs) und Teilen des BIOS erledigt

- **Root of Trust for Storage (RTS)**
  - Schützt Schlüssel und Daten, denen das TPM vertrauen muss
  - Auch für Daten ausserhalb des TPM
  - Wird vom TPM erledigt

- **Root of Trust for Reporting (RTR)**
  - Erstellt auf vertrauenswürdige Weise Bescheinigungen über die Integrität von Daten die RTS verwaltet
  - Beispielsweise Attestierungen
  - Wird vom TPM erledigt

- **TCG-Subsystem aus TPM und TSS stellt Betriebssystemen vertrauenswürdige Dienste und Mechanismen zur Verfügung**

Designziele der TCG-Architektur

- **Authentifizierung der Systemkomponenten und deren Konfiguration (sicheres Booten)**

- **Sicheres Generieren und Schutz kryptographischer Schlüssel (Speichern in Hardware)**

- **Remote Platform Attestation**
  - Plattformbenutzer fragt eine Dienstleistung bei einem Anbieter an
  - Dienstleister stellt eine Attestierunganfrage an das TPM
  - Root of Trust for Reporting (RTR) als Teil des TPM erstellt eine entsprechende Attestierung, die den Integritätszustand der Konfiguration beschreibt und signiert diese (mit dem passenden AIK)
  - Diensteanbieter erfragt die Gültigkeit der Signatur bei einer CA
  - Diensteanbieter überprüft die Konfiguration

- **Sealing**
  - Systemkonfiguration wird beim Booten bestimmt
  - Über einen Hash-Wert aus der Systemkonfiguration werden Daten und Applikationen an diese Konfiguration “gebunden”
  - Ver- und Entschlüsselung funktioniert nur anhand dieser Konfiguration

Trusted Platform Module (TPM)

- **Nach US-Senator Fritz Hollings, der sich sehr für DRM stark macht, auch “Fritz Chip” genannt**
- **Chip auf Motherboard (geplant ist aber auch eine Integration in CPUs und andere Bausteine)**
- **Kryptographische Funktionseinheiten**
  - Random Number Generator (RNG)
  - Hash-Einheit (SHA-1)
  - HMAC (Keyed Hashing for Message Authentication)
  - Generator für RSA-Schlüssel mit bis zu 2.048 Bit
  - RSA-Einheit zum Erzeugen von Signaturen (nicht prüfen), sowie zum Ver- und Entschlüsseln
- **Enthält einen vertrauenswürdigen Zeitgeber (timer)**
- **Führt beim Start einen Selbsttest auf Manipulation durch und vermerkt das Ergebnis (deaktiviert sich aber nicht selbst)**

Blick ins TPM

Funktionale Einheit	Nicht flüchtiger Speicher	Flüchtiger Speicher
Random Number Generator	Endorsement Key (2048 Bit)	RSA Key Slot-0 … RSA Key Slot-9
Hash (SHA-1)	Storage Root Key (2048 Bit)	PCR-0 … PCR-15
HMAC	Owner Auth Secret (160 Bit)	Key Handle
RSA Key Generation		Auth Session Handle
RSA Encrypt/Decrypt		

TPM – Nicht flüchtiger Speicher I

- **Endorsement Key (EK)**
  - 2.048 Bit RSA-Schlüsselpaar
  - Beim Herstellungsprozess im TPM generiert oder hinein geschrieben
  - Hersteller signiert den EK
  - Nicht lösch- oder änderbar (ab TPM 1.2 ist Löschen möglich)
  - Privater Teil des EKs verlässt das TPM nie
  - Öffentlicher Teil aus Sicht der Privatsphäre kritisch (siehe Seriennummern von Intel-Prozessoren) -> AIKs
  - Öffentlicher Teil Basis zur “Attestation”
  - Öffentlicher Teil zur Verschlüsselung von sensitiven Daten die an den Chip gesendet werden (zum Beispiel beim “Besitz übernehmen”)

- **Attestation Identity Keys (AIK)**
  - Mit EK signierte pseudonyme Schlüssel (beliebig viele)
  - Bestätigt Vorhandensein und Konfiguration des TPM (zum Beispiel PCRs) ohne den öffentlichen Teil des EK selbst herauszugeben
  - Signierte AIKs werden mit EK-Zertifikat nur an vertrauenswürdige Zertifizierungsstellen (Privacy CA) herausgegeben
  - AIKs können auch verschlüsselt ausserhalb des TPM aufbewahrt werden

TPM – Nicht flüchtiger Speicher II

- **Storage Root Key (SRK)**
  - 2.048 Bit RSA Schlüsselpaar
  - Initial ist der Speicherplatz leer
  - Wird beim “Besitz übernehmen” generiert
  - Privater Teil verlässt den Chip nie
  - Kann vom Systembesitzer gelöscht werden
  - Bildet die Wurzel einer Schlüsselhierarchie
  - Dient zum Verschlüsseln (wrap) von anderen Schlüsseln der ersten Hierarchiestufe die außerhalb des Chips gespeichert werden, sowie beim Entschlüsseln dieser Schlüssel wenn sie wieder in den Chip geladen werden

- **Owner Authorization**
  - 160 Bit Schlüssel den der Besitzer mit dem Chip teilt
  - SHA-1 Hash des angegebenen Passworts mit EK verschlüsselt
  - Das Passwort selber kann 256 Byte lang sein
  - Wird beim “Besitz übernehmen” im Chip erzeugt
  - Autorisierung von sensitiven Benutzerbefehlen

TPM – Flüchtiger Speicher I

- **Key Handles**
  - Um temporär geladenen Schlüsseln Namen zur weiteren Verwendung zuzuweisen
  - Werden gelöscht wenn der Schlüssel aus dem Chip geworfen wird

- **Authorization Session Handle**
  - Wird genutzt um den Status der Autorisation für mehrere hintereinander abfolgende Befehle beizubehalten

- **Ab TPM 1.2 zusätzlich mindestens 160 Bit große Speicherplätze (Data Integrity Register)**

TPM – Flüchtiger Speicher II

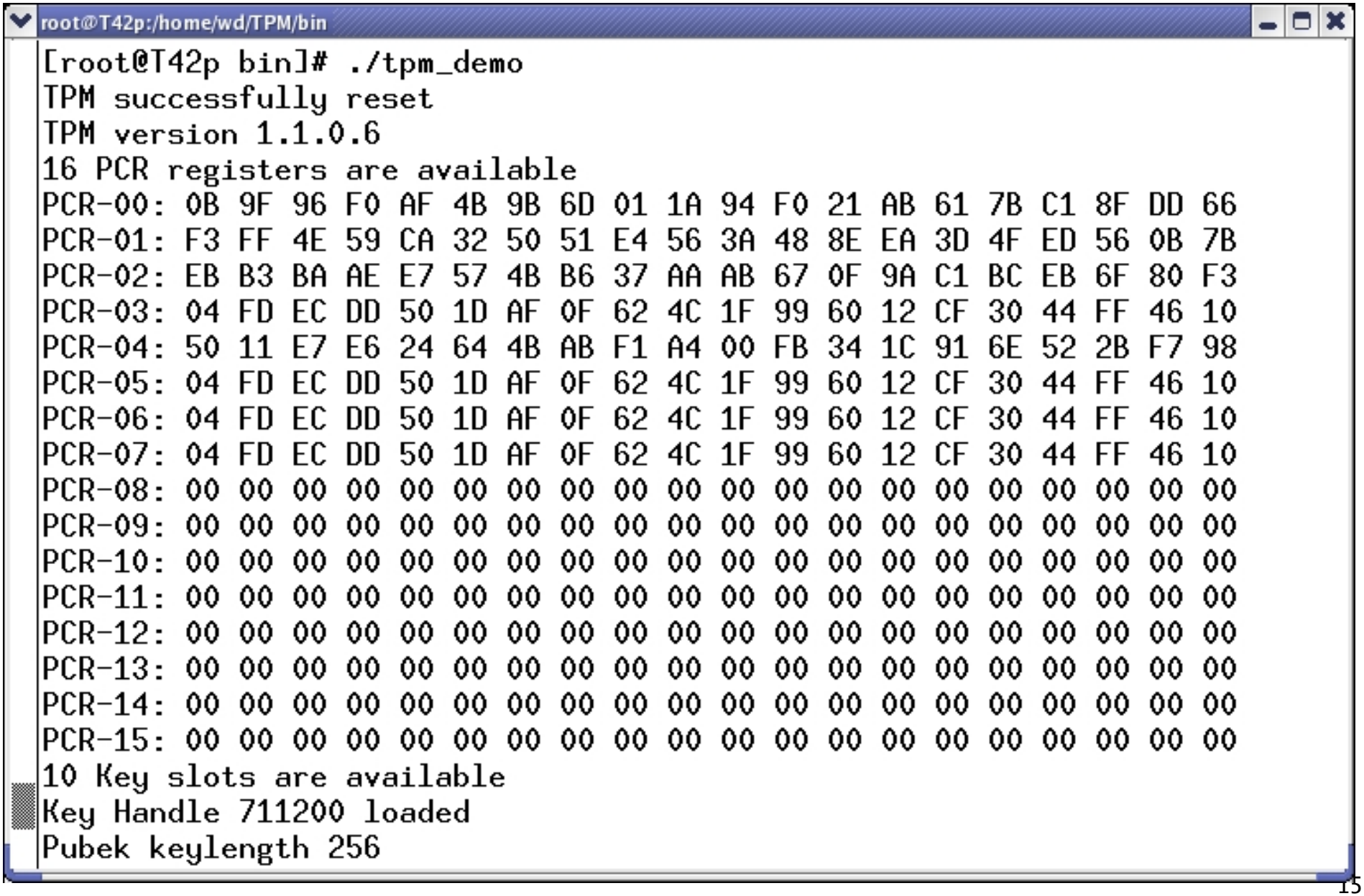
- **10 Slots für RSA-Schlüssel**
  - Extern gespeicherte Schlüssel können hier, nach Eingabe des Passworts, in den Chip geladen und genutzt werden
  - Können aus dem Slot geworfen (evicted) werden um den Platz frei zu geben

- **16 Slots für Platform Configuration Register (PCRs)**
  - 160 Bit für ermittelte Hash-Werte der Integritätsmessungen
  - Folge von integritätswerten möglich: $PCR_i = HASH(PCR_{i-1}, Wert)$
  - Zugriff nur im Rahmen von Sicherheitsdiensten
  - Beim Booten können zum Beispiel Messungen vom BIOS, erweitertem BIOS, MBR und anderen Daten (z.B. Kernel), aber auch von Hardware die dies unterstützt erzeugt und hier gespeichert werden
  - Ab TPM Version 1.2 sind für die PC-Plattform 24 Register vorgesehen

“sicherer” Bootprozess (Beispiel)

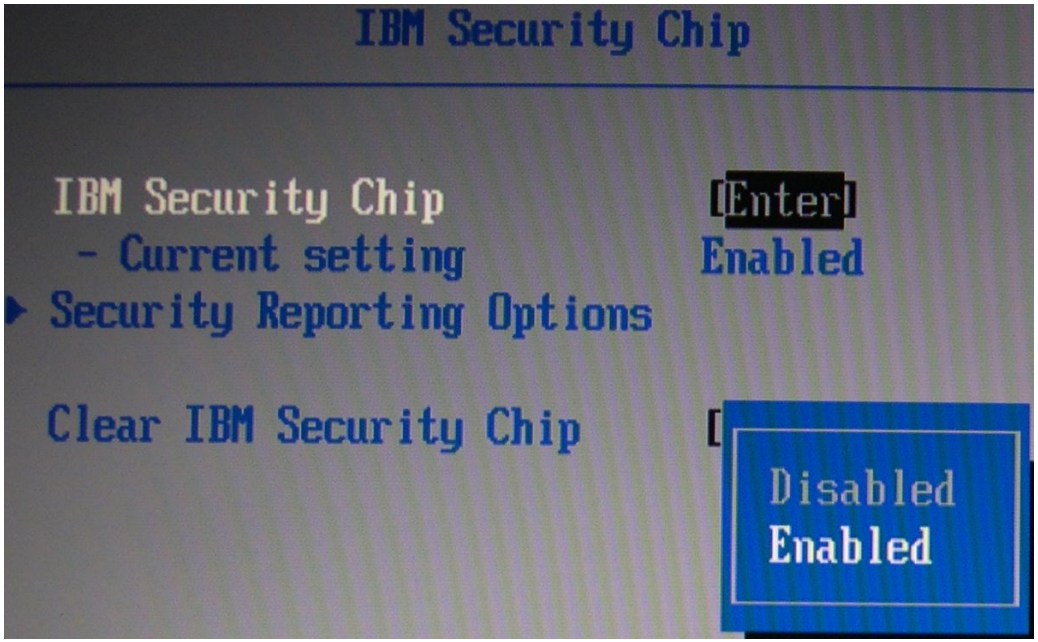
- **Aufbau einer Vertrauenskette**
  - Core Root of Trust Measurement (CRTM) – entweder Bestandteil des TPM oder eine BIOS-Erweiterung
  - CRTM bildet einen Hash-Wert über sich selber, speichert diesen in PCR[0] und läd erst dann den CRTM-Code in den Speicher
  - Konfiguration des Motherboards und angeschlossener Hardware wird gehasht und in PCR[1] geschrieben
  - Option ROMs des BIOS werden gehast in PCR[2] und PCR[3] gespeichert bevor sie geladen und ausgeführt werden
  - MBR gehasht in PCR[4]
  - Daten aus dem MBR (z.B. Plattenlayout) werden in PCR[5] gespeichert
  - Kontrollübergänge werden in PCR[6] gehasht
  - Schrittweises hashen des Bootloaders (jeweils vor Ausführung) in PCR[4]
  - Hash-Wert des geladenen OS-Kerns ebenfalls in PCR[4] abgelegt
- **Kette wird hier nach dem Booten unterbrochen**
- **Die Integritätswerte müssen ausgewertet werden (Betriebssystem)**

Inhalt eines TPM-Chips (PCRs) unter Linux



Aktivieren und Löschen des TPMs durchs BIOS

- **BIOS gibt TPM beim Einschalten des Rechners ein Startkommando (drei Möglichkeiten)**
  - TPM deaktivieren (kann bis zum Einschalten nicht mehr aktiviert werden)
  - TPM starten und Reset der PCRs, Inhalte der PCRs beim neu Booten berechnen
  - TPM starten und PCRs wieder herstellen falls sie vorher gespeichert wurden (Resume-Modus)



- **BIOS kann TPM “komplett” resetten (ForceClear)**
  - Benötigt Beweis der physikalischen Präsenz (Fn beim Systemstart gedrückt halten und mit F1 ins BIOS wechseln)
  - Wirft alle geladenen Schlüssel und Handles raus und löscht den SRK sowie das Owner Authorization Secret

Chancen

- **Sicherer Hardwarespeicher**
  - Verhindert unter anderem off-line-Angriffe auf Geheimnisse
- **Sicheres Booten**
  - TPM als sicherer Hardwareanker (Root of Trust)
  - Chain of Trust beim Boot-Prozess
- **Digital Rights Management (DRM)**
  - Mit Hardwareanker sicherer und zuverlässiger als ausschließlich in Software (bzw. Betriebssystem)
- **Remote Attestation**
  - GRID-Computing
  - Peer-to-Peer Computing
- **Vergleich von TPM zu Smartcards**
  - TPM authentifiziert Plattform
  - Smartcard authentifiziert Benutzer

Exkurs: Digital Rights Management (DRM)

- **Ziel von DRM-Systemen**
  - Abgrenzung zum reinen Kopierschutz (lediglich das Verhindern von illegaler Vervielfältigung)
  - DRMS erlauben dem Rechteinhaber digitaler Inhalte zusätzlich ein Rechtemodell gegenüber dem Rechteverwerter durchzusetzen

- **Rechtemodelle können beinhalten**
  - Wiedergaberechte (Anhören, Ansehen, Ausdrucken, ...
  - Transportrechte (Kopieren, Vermieten, Weitergeben, ...)
  - Derivativrechte (Extrahieren, Editieren, Einbinden, ...)
  - Dienstrechte (Sicherung, Caching, Integritätssicherung, ...)

- **Geschäftsmodelle (Kombinationen möglich)**
  - Zeitlich befristete Nutzung
  - Mengenabhängige Nutzung (n Wiedergaben / Aufrufe)
  - Geräteabhängige Nutzung
  - Gebrauchsorientierte Nutzung (n Minuten wiedergegeben)

Risiken

- **Remote Platform Attestation**
  - Wirklich (komplette) Hard- und Softwareumgebung preisgeben?
- **Sealing, Zensur, DRM, ... (TCG ist “policy neutral”)**
- **Open Source Software / Patente**
- **Gefahr von (nicht entdeckbaren) Hintertüren**
  - Ron Rivest: “... renting out a part of your PC to people you may not trust.”
- **Migrations / Backup der Schlüssel (bzw. Daten)?**
- **“ungeeignete” Kryptographie**

“ungeeignete” Kryptographie

- **Februar 2005: erfolgreicher Angriff auf SHA-1**
  - SHA-1 bildet Hash-Werte mit 160 Bit
  - Für Kollisionen “nur” noch 2^{69} statt 2^{80} Nachrichten überprüfen (Faktor 2.048, 2^{11})
  - Existierende signierte Nachrichten und selbst erstellte Dokumente sind sicher

- **2004er Empfehlungen des BSI / der RegTP (Regulierungsbehörde für Telekommunikation und Post)**
  - SHA-1 und RIPEMD-160 sind bis 2009 geeignet
  - SHA-256, SHA-384 und SHA-512 gewähren ein langfristiges Sicherheits-niveau (mindestens bis 2009)
  - Bei RSA für langfristiges Sicherheitsniveau 2.048 Bit empfohlen (Mindestwert bis Ende 2009 sind 1.536 Bit)

Status Quo Hardware

- **Über 16 Millionen Motherboards mit TPM (1.1b) sind ausgeliefert**
  - STMicroelectronics und Atmel fertigt bereits 1.2er TPMs
  - Infineon kündigt 1.2er TPMs für Juli 2005 an

- **Verfügbare Komponenten (Beispiele)**
  - Hauptsächlich Komplettrechner oder Motherboards für Unternehmenskunden (u.a. von IBM, HP und Fujitsu-Siemens)
  - Bei IBM ist das TPM 1.1b innerhalb des Embedded Security Subsystems 2.0 verbaut
  - Gigabit-Ethernet-Controller von Broadcom (1.1b)
  - Von Intel bereits Chipsätze und Motherboards mit TPM 1.2 erhältlich

- **Angekündigt unter anderem**
  - TPM integriert in I/O-Chip von National Semiconductor mit Ports für Tastatur, Maus, Drucker, Floppy, RS-232
  - Trusted Mode Keyboard Controller (Intel, Microsoft)
  - USB-Security-Extension (Intel, Microsoft)
  - TPM in CPU (Intel – LaGrande Technology)

Microsofts Paladium bzw. Next Generation Secure Computing Base (NGSCB)

- **Vorschlag 2003**
  - Windows in Quadranten Aufteilen
  - Left Hand Site (LHS) – ungesicherte Windowsumgebung
  - Right Hand Site (RHS) – Anwendungen im Trusted Mode
  - LHS und RHS haben jeweils Benutzer- und Kernel-Modus
  - Nexus im Kernel-Modus in der RHS

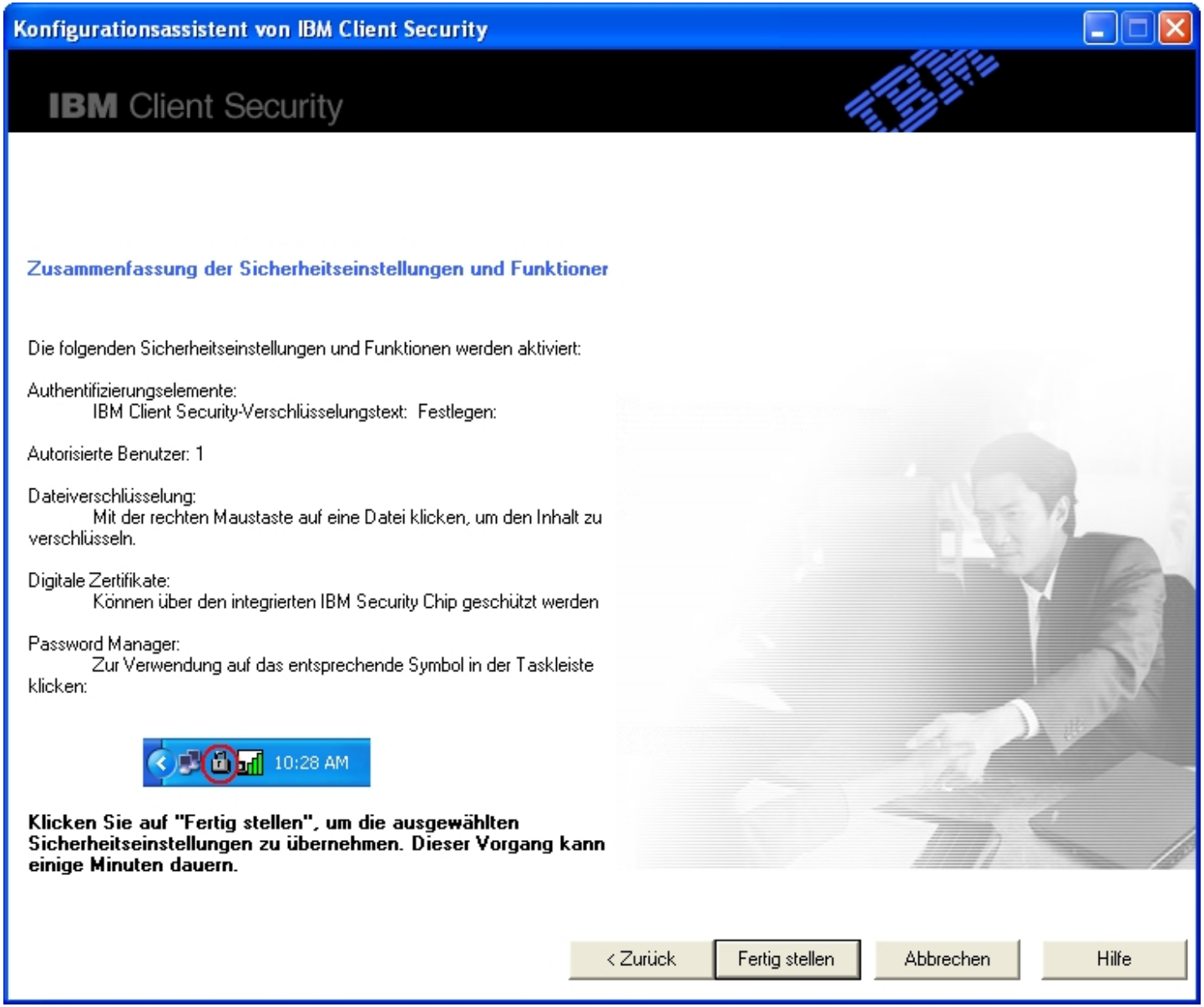
- **Ankündigungen auf der WinHEC 2004**
  - LHS nahezu unverändert
  - RHS wird komplett überarbeitet
  - Compartmens – abgeschottete virtuelle Systeme parallel zum Hauptbetriebs-system

- **Ankündigungen auf der WinHEC 2005**
  - Secure Startup
  - Full Volume Encryption in Longhorn
  - Unterstützung 1.2er TPM
  - Compartments in Longhorn Server erst frühestens 2007

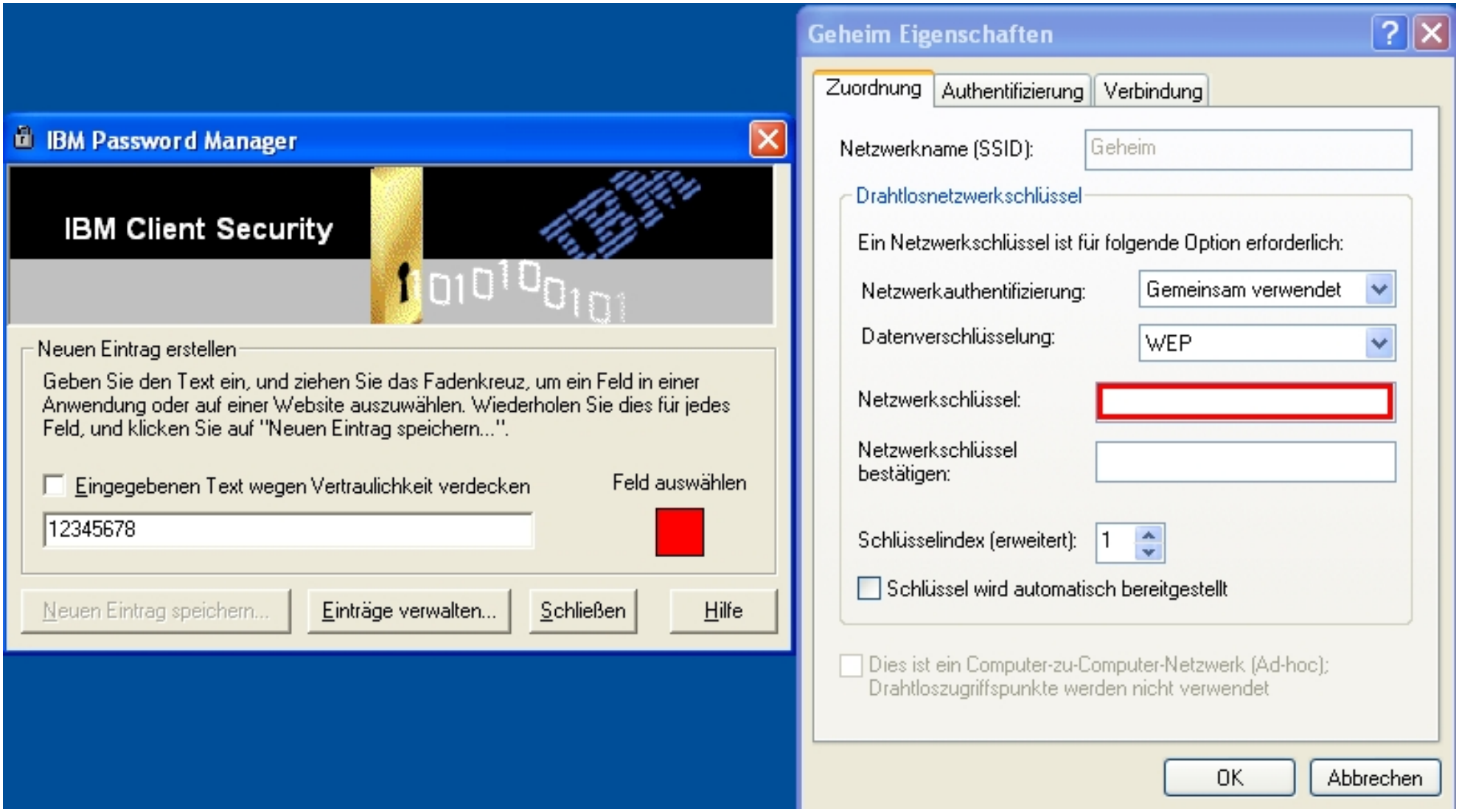
Status Quo Software

- **Kommerzielle Applikationen mit TPM-Support (Beispiele)**
  - Utimaco SafeGuard (Laufwerksverschlüsselung)
  - Check Point VPN-1 SecureClient
  - Adobe Acrobat 6.0 (Verschlüsselung bzw. Zugriffskontrolle von PDF-Dokumenten – DRM-Lösung)
- **TPM-Unterstützung im Linux Kernel ab 2.6.12**
- **Software von IBM**
  - Windows: verändertes Login, rudimentäre Verschlüsselungswerkzeuge
  - Linux-Testpaket (samt Quellen) mit Kernel-Modul, Bibliothek, API und Beispielprogrammen
  - tcgLinux: TPM-based Linux Run-time Attesttation
- **Forschungsprojekte**
  - Enforcer Linux Security Module
  - PERSEUS
  - European Multilateral Secure Computing Base (EMSCB)

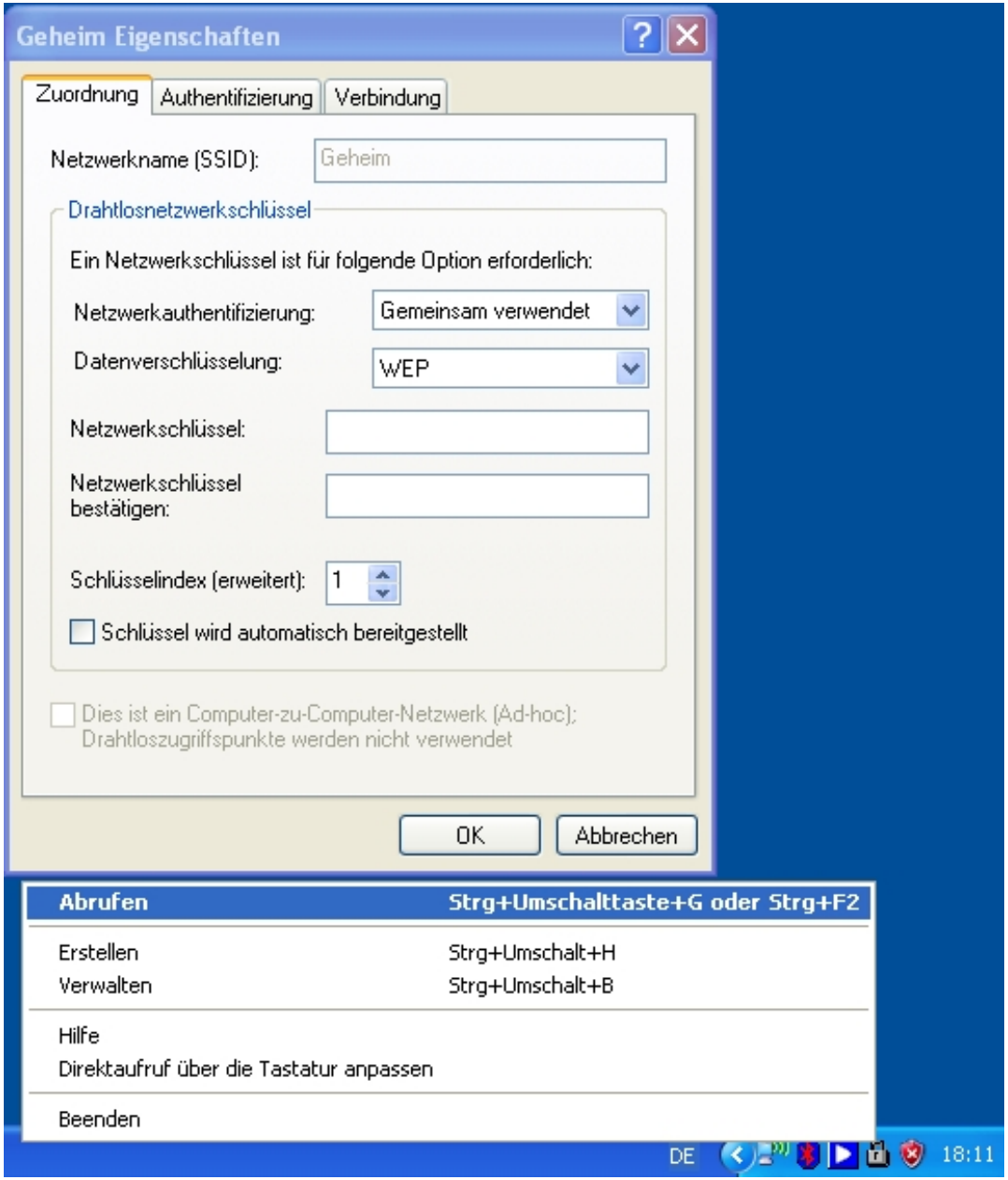
Windows – IBM Client Security



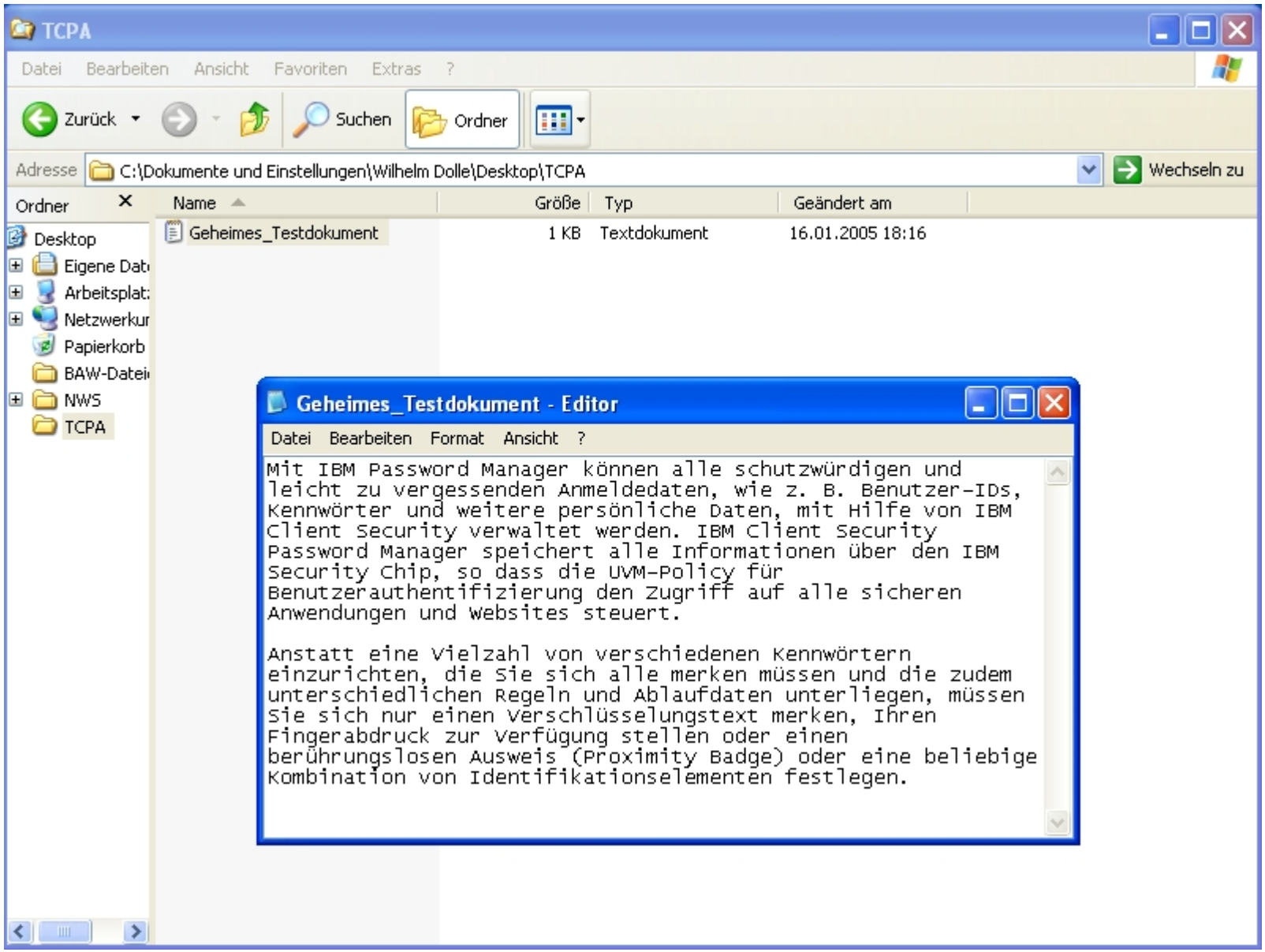
Windows – IBM Password Manager (Capture)



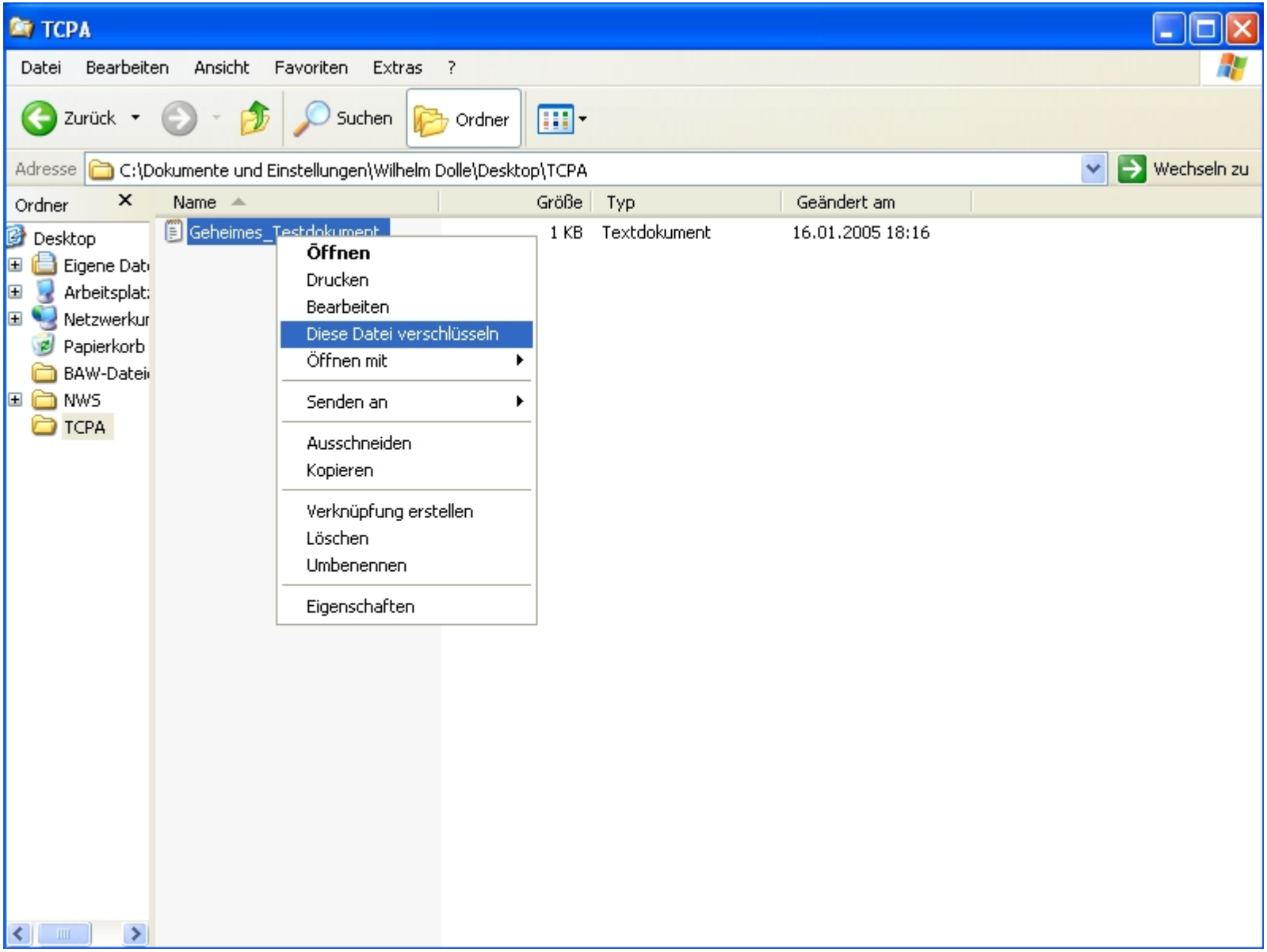
Windows – IBM Password Manager (Paste)



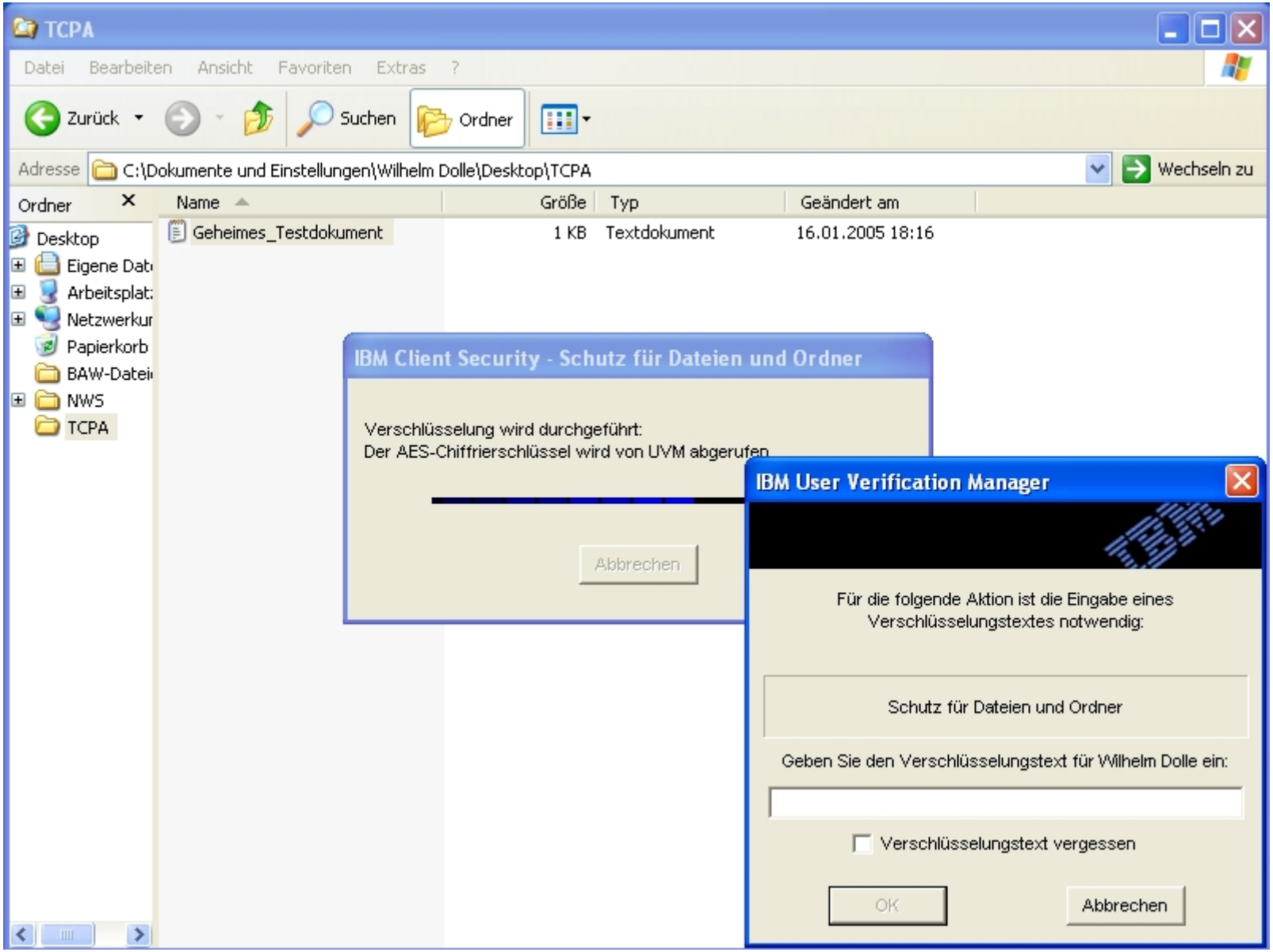
Windows – IBM Client Security (Dateiverschlüsselung)



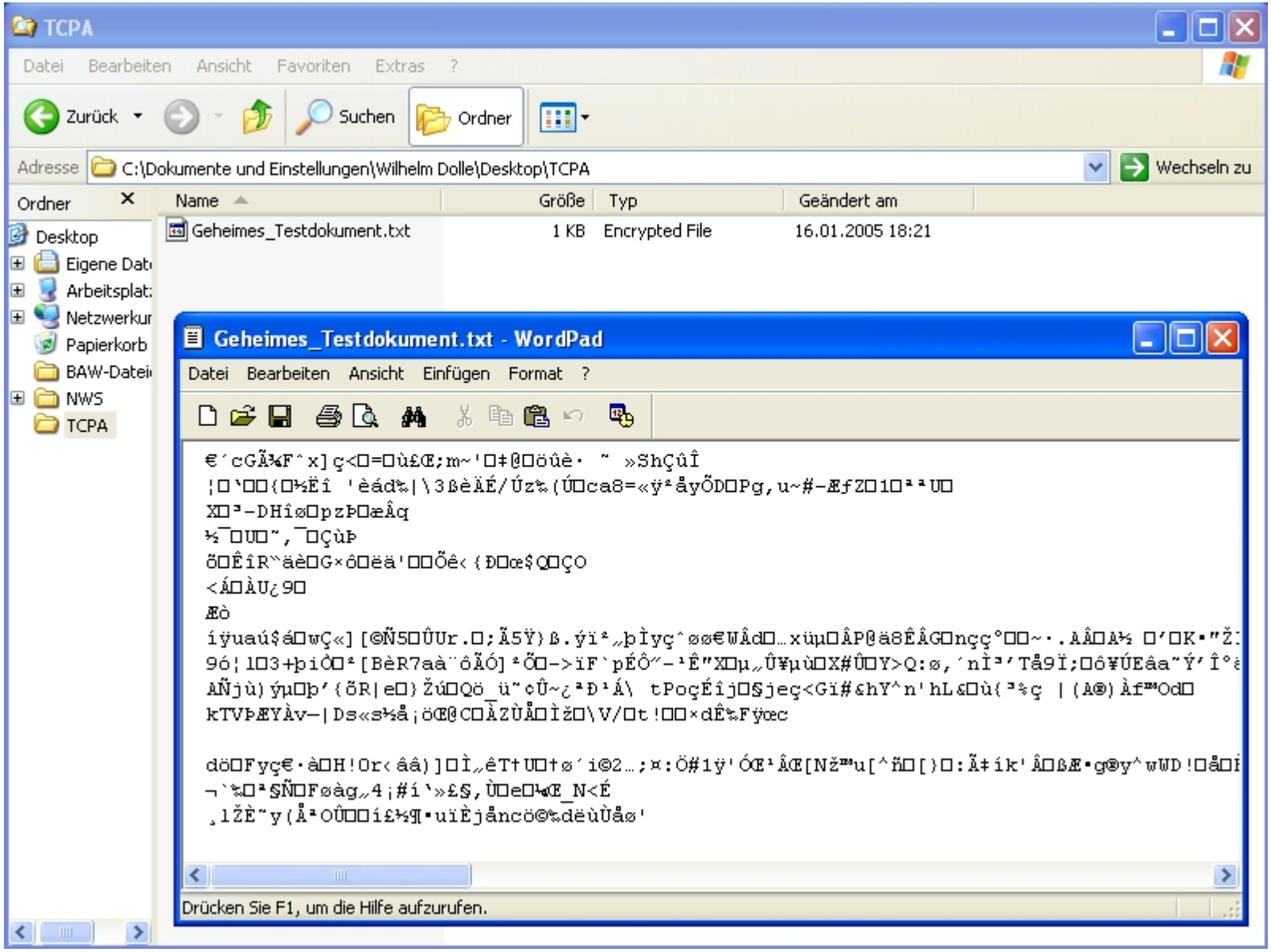
Windows – IBM Client Security (Dateiverschlüsselung)



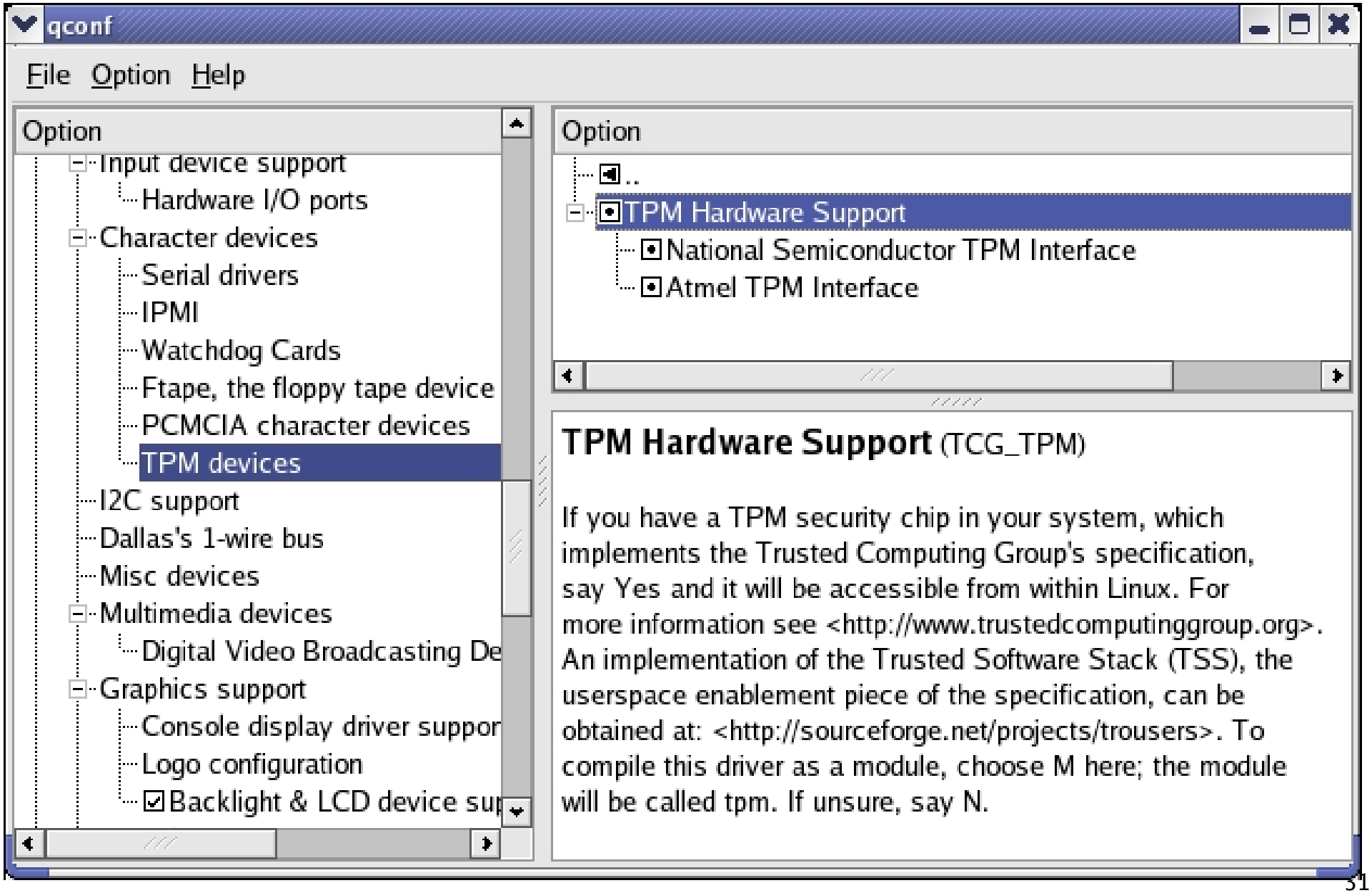
Windows – IBM Client Security (Dateiverschlüsselung)



Windows – IBM Client Security (Dateiverschlüsselung)



TPM-Unterstützung direkt im Linux Kernel 2.6.12



TPM Kernel Module

- **Module direkt im 2.6.12er Linux Kernel**

```
root@T42p:~
[root@T42p ~]# ls -l /lib/modules/2.6.12/kernel/drivers/char/tpm/
total 318
-rw-r--r--  1 root root  97532 Jun 21 01:07 tpm_atmel.ko
-rw-r--r--  1 root root 119780 Jun 21 01:07 tpm.ko
-rw-r--r--  1 root root 103571 Jun 21 01:07 tpm_nsc.ko
[root@T42p ~]#
```

- **Modul aus dem IBM TPM Device Driver Projekt (http://www.research.ibm.com/gsal/tcpa/)**

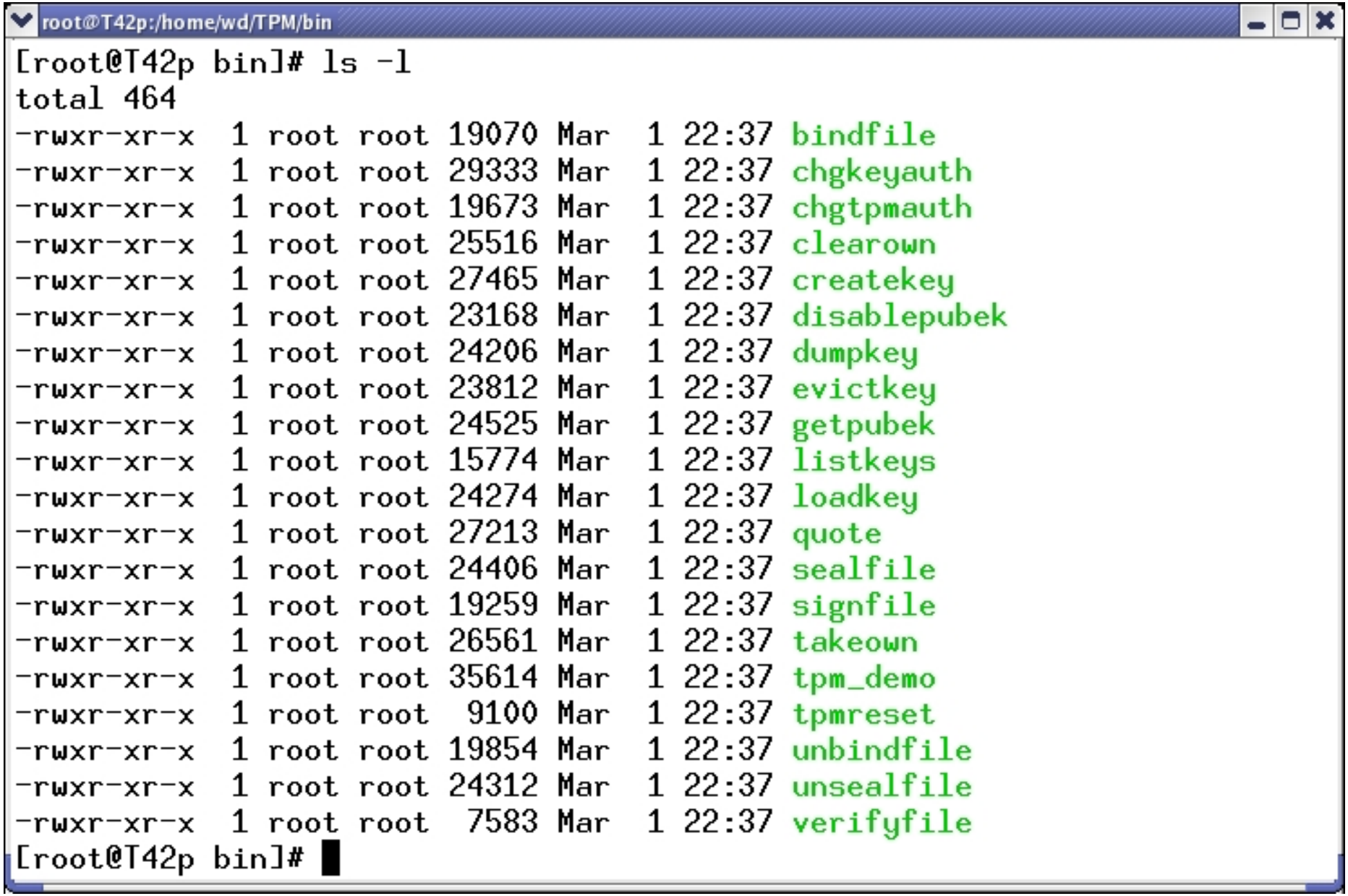
```
root@T42p:/home/wd/Software/tpm-2.0
[root@T42p tpm-2.0]# ls -l
total 356
-rw-r--r--  1 wd   wd    18010 Oct 18  2004 COPYING
-rw-r--r--  1 wd   wd      202 Oct 18  2004 Makefile
-rw-r--r--  1 wd   wd      576 Oct 18  2004 Makefile-2.4
-rw-r--r--  1 wd   wd    43084 Oct 18  2004 tpm.c
-rw-r--r--  1 wd   wd      909 Oct 18  2004 tpm.h
-rw-r--r--  1 root root 133789 Mar  1 22:39 tpm.ko
-rw-r--r--  1 root root   1779 Mar  1 22:39 tpm.mod.c
-rw-r--r--  1 root root  30816 Mar  1 22:39 tpm.mod.o
-rw-r--r--  1 root root 104214 Mar  1 22:39 tpm.o
[root@T42p tpm-2.0]#
```

IBM Device Driver (libtpm)

```
root@T42p:/home/wd/Software/libtpm-2.0
[root@T42p libtpm-2.0]# ls
aclocal.m4  config.status  depcomp     lib          missing        utils
AUTHORS     configure      doc         Makefile     mkinstalldirs
ChangeLog   configure.ac   INSTALL     Makefile.am  NEWS
config.log  COPYING        install-sh  Makefile.in  README
[root@T42p libtpm-2.0]#
```

```
root@T42p:/home/wd/Software/libtpm-2.0/doc
[root@T42p doc]# ls
bindfile.1       signfile.1           TPM_setlog.3
chgkeyauth.1     takeown.1            TPM_Sign.3
chgtpmauth.1     TPM_BuildKey.3       TPM_TakeOwnership.3
clearown.1       TPM_ChangeAuth.3     TPM_Unbind.3
createkey.1      TPM_CreateWrapKey.3  TPM_Unseal.3
disablepubek.1   TPM_EvictKey.3       TSS_Bind.3
dumpkey.1        TPM_GetErrMsg.3      TSS_convpubkey.3
evictkey.1       TPM_GetPubKey.3      TSS_GenPCRInfo.3
getpubek.1       tpmlib.3             TSS_Key2Pub.3
listkeys.1       TPM_LoadKey.3        TSS_KeyExtract.3
loadkey.1        TPM_OwnerClear.3     TSS_PubKeyExtract.3
Makefile         TPM_PcrRead.3        TSS_sha1.3
Makefile.am      TPM_Quote.3          unbindfile.1
Makefile.in      TPM_Seal.3           unsealfile.1
sealfile.1       TPM_SealCurrPCR.3    verifyfile.1
[root@T42p doc]#
```

IBM Device Driver (utils)

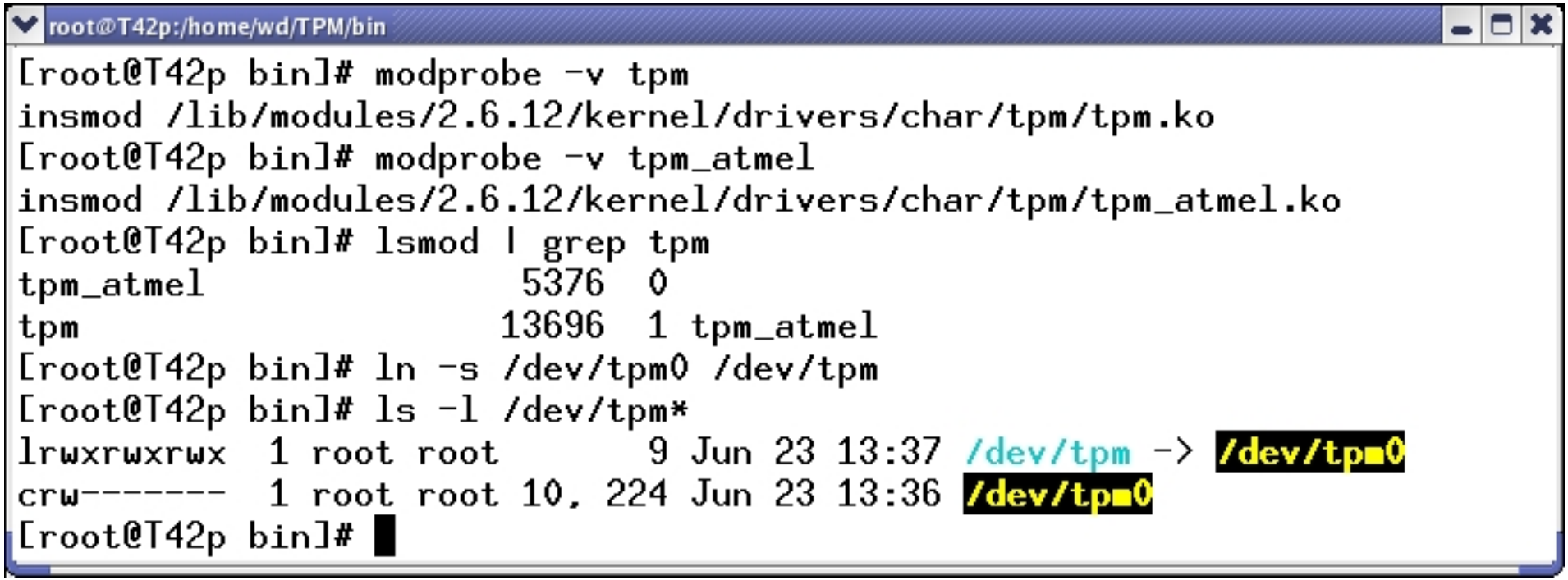


TrouSerS Projekt

- **Quelloffener Unter der CPL (Common Public License) veröffentlichter Trusted Computing Software Stack (TSS)**
- **Compliant mit der TSS 1.1b Spezifikation**
- **TCS Daemon (TSCD)**
  - User-Space-Dienst der (gemäß der Spezifikationen) der einzige Zugang zum TPM ist
  - Wird beim Booten gestartet und ab da müssen alle Anfragen an das TPM über diesen Dienst laufen
  - Behandelt lokale und entfernte (z.B. Remote Attestation) Anfragen
- **TSP (TCG Service Provider) shared Library**
  - Stellt dem TSCD und Anwendungen Ressourcen zur Verfügung und verwaltet diese
- **Persitant Storage Files**
  - User Persistant Storage – über die Lebensdauer einer Applikation
  - System Persistant Storage – Lebenszyklus des Systems oder des TSCDs

“Arbeiten” mit dem TPM

- **Module (manuell) starten**
- **Gerätedatei (Character Device) wird angelegt (/dev/tpm1)**
- **Link erzeugen (/dev/tpm)**



Besitzerkonzept

- TPM wird immer deaktiviert ausgeliefert
- Vor dem Gebrauch muss der Besitzer der TCG-Plattform das TPM explizit aktivieren
- Ohne die Aktivierung sind keinerlei Dienste des TPM verfügbar
- Explizite Besitzübernahme notwendig (Take_Ownership-Kommando)
- Über die Kenntnis eines Passworts kann dann der Zugriff kontrolliert werden
- Beim direkten Zugang zur Plattform kann das TPM auch ohne Wissen des Besitzers aktiviert bzw. deaktiviert werden (siehe BIOS, oder per Jumper)

Besitz Übernehmen

- **takeown <owner password> [<srk password>]**
  - Installation des angegebenen Owner Authorization Secret im TPM
  - Erzeugen des Storage Root Key (SRK-Schlüsselpaar) im TPM
  - Das SRK Authorization Secret wird auf den Schlüssel angewandt
  - Ausgabe des öffentlichen Teils des SKR Schlüsselpaares

```
root@T42p:/home/wd/TPM/bin
[root@T42p bin]# time ./takeown benutzer_pw srk_pw

real    0m14.927s
user    0m0.004s
sys     0m0.073s
[root@T42p bin]# ls -l srk.pem
-rw-r--r--  1 root root 451 Jun 23 14:28 srk.pem
[root@T42p bin]# cat srk.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAu1ZFbyNHq5mRLy2Bc0uU
PL2ZK9hJ6HSM0LX64TxZj5SBBXp5VeWWUnICAGVMZOqKX8NhahXrwCrabrut9hdF
s500TrqNhb049tYX2yra0rq8jJRJ5HMJHBay4N7N7KTLBWMBMqmNFqGrU6BMnukq
pRmNdkjBtpkA31IKj65hDR2rybEfixGewKFeN3CufNEmM+6IgywyKu949PV1aZmu
+2bvjChj0jmDZc/DfR6xVC4Zm6IRoj80XM0ef3LpHT0dxCH7e6pJ3AliL4hSUy0X
eTJuwilgPM4NEPN7WYhK5bI41KWLYqELWZzBoPNjzcFPNqeCa2maE7uUVNt9inxx
xQIDAQAB
-----END PUBLIC KEY-----
[root@T42p bin]#
```

Schlüssel erstellen

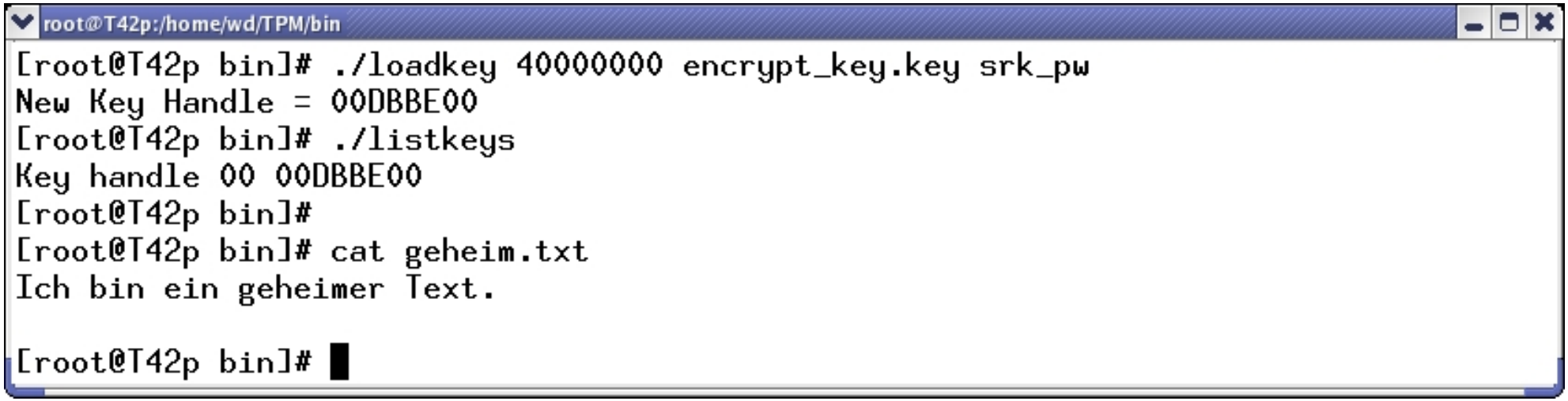
- **createkey [options] <keyname> <parent key handle>**
  - Erzeugt je nach Option (-t) einen Schlüssel zum signieren (s), verschlüsseln (e), binding (b) oder legacy (l)
  - Es kann eine Passphrase für den neuen Schlüssel angegeben werden
  - Benutzt der Parentschlüssel eine solche muss sie ebenfalls angegeben werden

Schlüssel erstellen

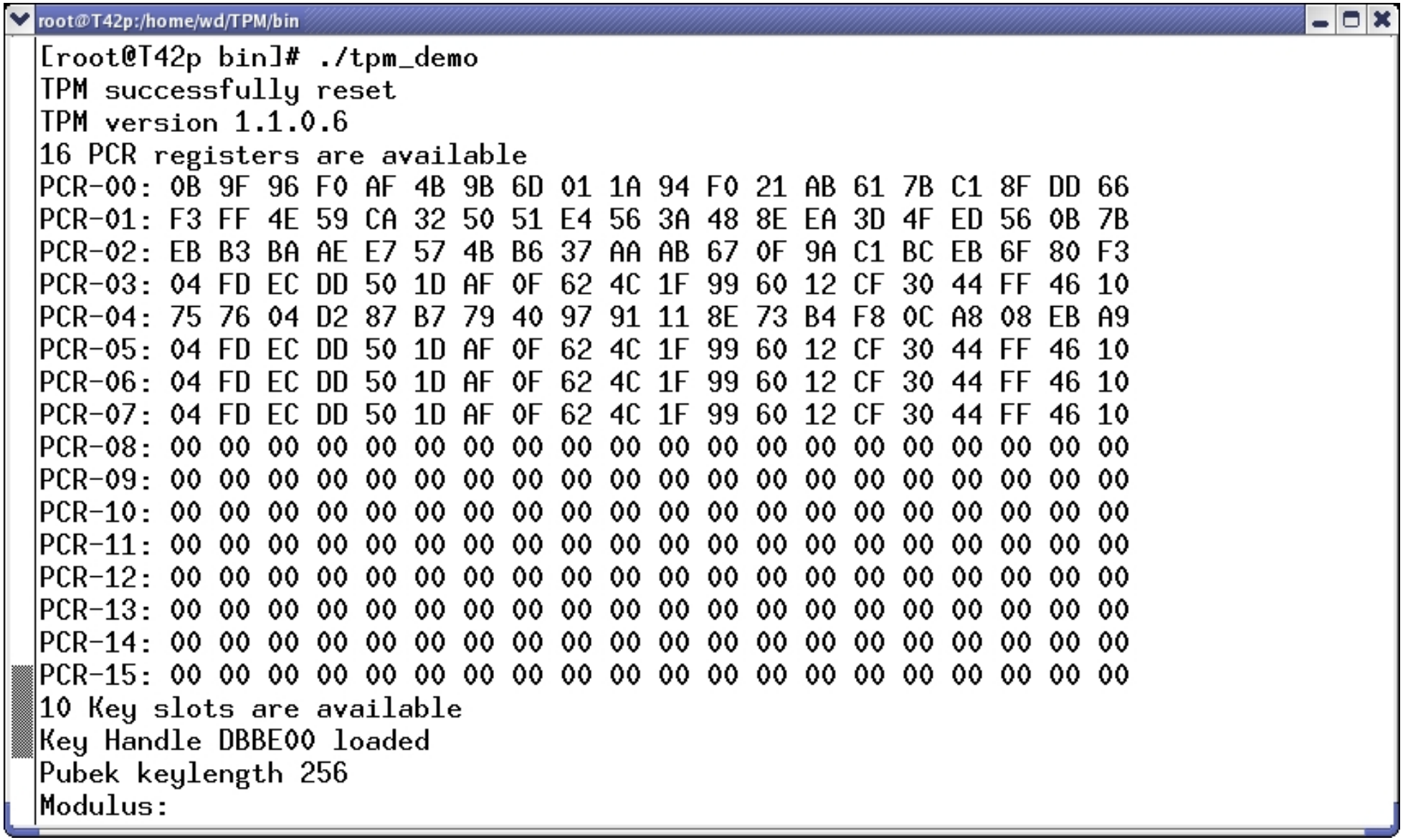
root@T42p:/home/wd/TPM/bin

```
[root@T42p bin]# time ./createkey -t e -k key_pw -p srk_pw encrypt_key 40000000

real    0m11.280s
user    0m0.002s
sys     0m0.038s
[root@T42p bin]# ls -l encrypt_key.*
-rw-r--r--  1 root root 559 Jun 23 18:26 encrypt_key.key
-rw-r--r--  1 root root 451 Jun 23 18:26 encrypt_key.pem
[root@T42p bin]#
[root@T42p bin]# cat encrypt_key.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAx1Qduzv34Ai3+WttR0Sk
S0egw6lpk1nqi9xpshiOOGOH7DYGaMk2PNywusQKXTY+IXJPXbx79+jBWtSz1cCa
+c+M88JRzcFIN+wSH9XxjZebtalwRXkJyIuvjIACaaYDq6+x8vbFJzSg6SbBYHcB
/RCYNqjnUsmpqUOfCnAZdnxHmGdlWbUbz+GOPQ/zhXRmWao1TmKvuxgU2SXQCQ1d
vJy5gjrddmT+LVhkg1AKbBzNCxGAtnFU2Q7co01JMD5MnhEVetSDLdpcsupk8e/u
LOQs0ay88B/KGuPR/gUxxe9UcbMIlWPL7BPd5+CjwVNYKKwo9HXgHgJ9YYWDuI5L
vQIDAQAB
-----END PUBLIC KEY-----
[root@T42p bin]# cat encrypt_key.key

□□;□□mGD□H□□i□Y□i□□8c□□h□K□□
                                ]6>!r0]□{□□%□□□□□Q□H7□□□□□□□pEy      i□□□i□□□□□□4□□□w□□6
□□ı□C□
pvlG□geY□□=□fY□5Nb□□□□  ]□□□□:□d□-Xd□P
l□□□qT□□MI0>L□zd-□□□□□□□9□□□□□□□1□Tq□c□□□SX(□(□□}a□□□□K□□j2XB□□□□□□#n□n□□{_z9□□x
                                                                                □□□□?Q
&O□□□Y□□□□□□□.$)F□□□Z□□S□□t□□□(k=□□k□z□□\_□]□Gq□βM□rLty/□S
                                                                    □gO(□□□Y□□□□OOA□^□l□□(>□□□□
b□□<mvl□~GZ□□□*□□nkM
[root@T42p bin]#
```

Schlüssel laden



Schlüssel geladen

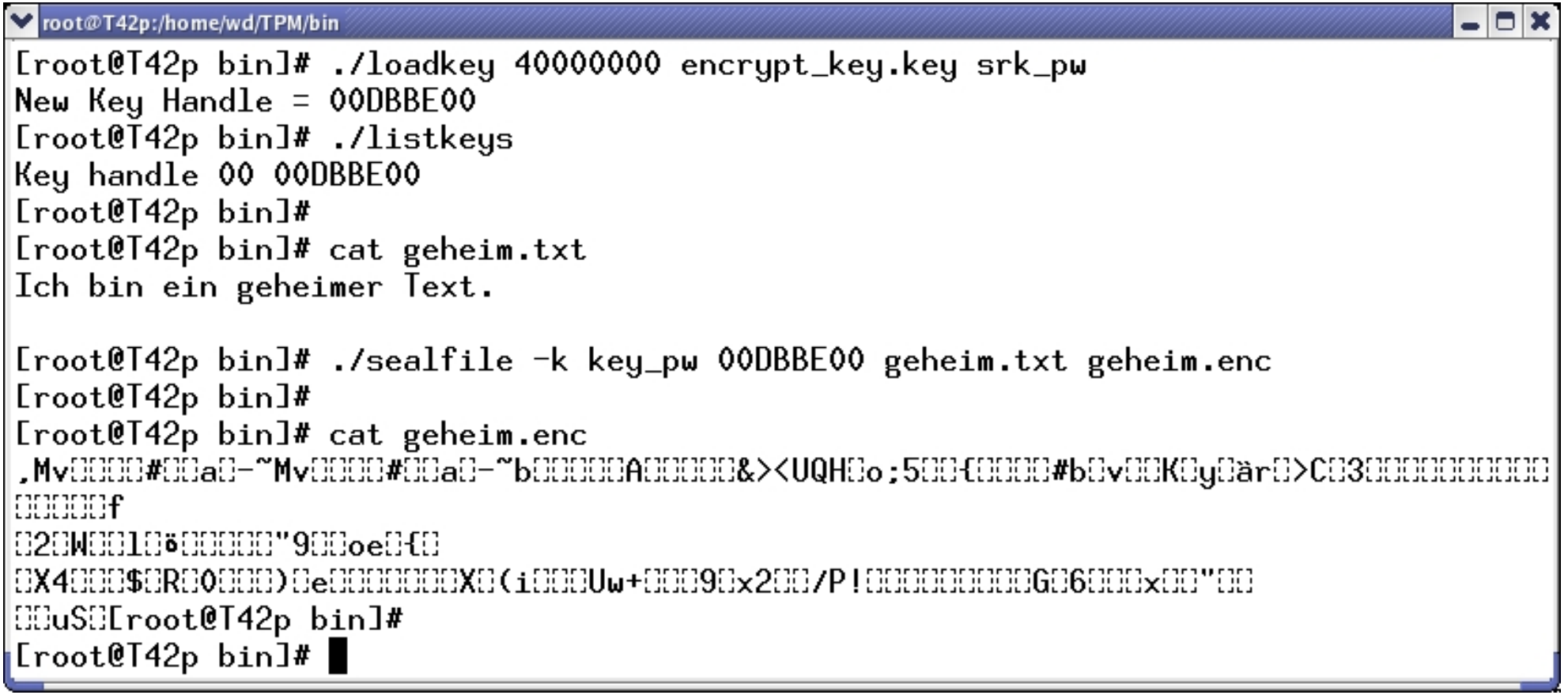


Sealing

- **Bedeutung der PCRs beim Sealing**
  - TPM_Seal_CurrPCR() arbeitet mit aktuellen PCR-Werten
  - TPM_Seal() kann mit beliebigen PCR-Werten arbeiten

- **sealfile [options] <key handle> <input file> <output file>**
  - Als Optionen können die Passphrasen des benutzten Schlüssels sowie ein Passwort für das Datenfile angegeben werden

Sealing


root@T42p:/home/wd/TPM/bin
[root@T42p bin]# ./loadkey 40000000 encrypt_key.key srk_pw
New Key Handle = 00DBBE00
[root@T42p bin]# ./listkeys
Key handle 00 00DBBE00
[root@T42p bin]#
[root@T42p bin]# cat geheim.txt
Ich bin ein geheimer Text.
[root@T42p bin]# ./sealfile -k key_pw 00DBBE00 geheim.txt geheim.enc
[root@T42p bin]#
[root@T42p bin]# cat geheim.enc
[root@T42p bin]#

Weitere Projekte

- **IBM tcgLinux – TPM-based Linux Run-time Attesttation**
  - Erweitert Integritätsprüfung vom Boot-Prozess auf alle geladenen Programme bzw. Konfigurationsdateien
  - Anfragendes System benötigt Hash-Wert-Datenbank

- **Enforcer Linux Security Module**
  - Linux Security Module (LSM)
  - Baut auf den IBM-Treibern für Linux auf
  - Gleicht beim Lesen von sensiblen Daten Hash-Werte mit Datenbank ab
  - Datenbank signiert und versiegelt
  - Modifizierter LILO überprüft Kernel Image und Master Boot Record

- **PERSEUS**
  - Security-Softwareschicht kontrolliert zu Schutz von sensiblen Anwendungen und Daten kritische Hardware-Ressourcen (auch das TPM)
  - Baut auf L4-Microkernel auf -> Codebasis PERSEUS max. 100.000 Zeilen

- **European Multilateral Secure Computing Base (EMSCB)**
  - Vorschlag für eine offene Computing Platform
  - Soll PERSEUS, TPM und herkömmliche Betriebssysteme kombinieren

Forderungen von Kritikern

- **Endorsement Key austauschbar**
  - Bereits in TPM Spezifikationen 1.2 enthalten
  - Für kleine Organisationen nicht sinnvoll einsetzbar
- **Direct Anonymous Attestation**
  - Bereits in TPM Spezifikationen 1.2 enthalten
  - Beliebig viele anonyme Zertifikate
  - Unlinkbarkeit
- **Vollständige Kontrolle über alle TPM-Schlüssel (CCC)**
- **Owner Override (EFF)**
- **Internationale und unabhängige Kontrolle des TPMs muss möglich sein (CPU-Integration?)**

Fazit

- Rechteinhaber können über wirksames (hardware-basierendes) DRM Inhalte in digitaler Form veröffentlichen (und vermarkten)
- Anwender werden DRM nur einsetzen (und dafür bezahlen) wenn es einfach und sicher zu benutzen ist
- Interessen des Anwenders vs. Interessen der Industrie (wer hat die Kontrolle über Inhalte und Hardware?)
- Die Verbreitungsfreiheit von Wissen könnte durch DRM (bzw. die TCG-Konzepte) eingeschränkt werden
- Es lohnt sich die Entwicklung bezüglich TCG / DRM im Auge zu behalten

Fragen?

Vielen Dank für die Aufmerksamkeit

Folien unter: http://www.dolle.net

Wilhelm Dolle, CISA, CISSP, BSI IT-Grundschutz-Auditor
Director Information Technology

iAS interActive Systems
Dieffenbachstrasse 33c
D-10967 Berlin

phone +49-(0)30-69004-100
fax +49-(0)30-69004-101
mail wilhelm.dolle@interActive-Systems.de
web http://www.interActive-Systems.de