## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Introduction to TPM</td>
<td>2</td>
</tr>
<tr>
<td>2.2</td>
<td>Authenticated Boot</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1</td>
<td>CRTM</td>
<td>3</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Authenticated Boot</td>
<td>3</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Authenticated vs. Secure Boot</td>
<td>4</td>
</tr>
<tr>
<td>2.2.4</td>
<td>TrustedGRUB</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>Secure Storage</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Theoretical Assignments (30 Points)</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>Practical Assignments</td>
<td>10</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Activating the TPM from BIOS</td>
<td>10</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Initializing the TPM</td>
<td>10</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Taking Ownership</td>
<td>10</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Reading the PCR</td>
<td>10</td>
</tr>
<tr>
<td>2.5.5</td>
<td>Using TrustedGRUB</td>
<td>11</td>
</tr>
<tr>
<td>2.5.6</td>
<td>Measurements</td>
<td>11</td>
</tr>
<tr>
<td>2.5.7</td>
<td>Using the Checkfile</td>
<td>12</td>
</tr>
<tr>
<td>2.5.8</td>
<td>Un-/Sealing</td>
<td>12</td>
</tr>
<tr>
<td>2.5.9</td>
<td>Clearing the TPM</td>
<td>13</td>
</tr>
</tbody>
</table>

### Motivation and goals:

Goal of this exercise is to give you basic practical skills for working with the Trusted Platform Module (TPM). This includes all important actions of the chip such as taking ownership or reading the Platform Configuration Registers (PCR). Later in the exercises you will learn to perform an authenticated boot with the open-source tool TrustedGRUB and how to (un-)seal data with the TPM Tools.

### Procedure:

The exercise is split into two parts: theoretical and practical. The theoretical exercises have to be solved before Friday, 3 June 2011, and be handed in at that date. The practical exercises will take place in the System Security Lab at CASED (Room 3.2.04). You will do practical exercises under supervision and questions will be solved together. Note that the questions discussed in the practical exercises are also relevant for the exam!
2.1 Introduction to TPM

The Trusted Platform Module (TPM) is a hardware chip specified and designed by the Trusted Computing Group industry consortium, that includes Intel, HP, Microsoft, IBM and others. It is a security microchip mounted on the motherboard of most contemporary PCs and laptops. The TPM stores cryptographic keys and other sensitive data in its shielded memory and provides ways to use these keys to achieve security requirements, such as secure storage. The current implementation consists of a cryptographic co-processor providing hardware-based random number generation and a small set of cryptographic functions (key generation, signing, encryption, hashing, MAC). In addition, the TPM offers three kinds of functionality: (i) Secure Storage where the user can store data that is encrypted by keys available only to the TPM; (ii) Platform Integrity Measurement and Reporting, which enables the platform able to create reports of its integrity and configuration state using the Platform Configuration Registers (PCR), that can be relied on by a remote verifier (Attestation); and (iii) Platform Authentication. Figure 2.1 gives an overview of the architecture of the TPM.

The TPM acts as the “Root of Trust” for the platform and must be trusted by all parties to behave correctly and not being compromised. For a more details we refer here to the lecture slides or the TCG Specification [CG].

Figure 2.1: TPM architecture

\[meaning: \text{ideally tamper-resistant}\]
2.2 Authenticated Boot

2.2.1 CRTM

Trusted Computing defined by the Trusted Computing Group (TCG) defines Storage and Platform Integrity Management and Reporting to be one of the main features. They are used to build a mechanism called Authenticated Boot.

In Trusted Computing, the Trusted Platform Module (TPM) acts as the Root of Trust. As the TPM is a passive component, measurement will not be initiated by the TPM, but instead is the task of each bootstrapping component. The TCG defined the so-called Core Root of Trust for Measurement (CRTM). The CRTM is a piece of executable code starting the measurement of bootstrapping components. It is an immutable portion of the host platform’s initialization code that executes upon platform reset.

It is important to note that trust in all subsequent measurements is based on the integrity of the CRTM. If the CRTM is tampered, all subsequent measurements cannot be trusted. Ideally, the CRTM is located inside the TPM, where it may be located in secure storage, but due to implementation decisions, the CRTM is mostly located in other firmware (e.g. BIOS boot block). It can be a part of the BIOS (e.g. BIOS is composed of a Boot Block and a POST BIOS) or the entire BIOS, where the latter choice would not be a good one, as it would include the whole BIOS into, and thus increase, the Trusted Computing Base.

2.2.2 Authenticated Boot

The overall flow of measurement and execution is depicted by figure 2.2. Upon system power-up, the TPM performs several initialization and self-test functions. In the following, it passes control to the CRTM which starts the chain of measurement by measuring and passing control to the BIOS. In general, every bootstrapping software $S_i$ performs the following steps, where $i$ is the position in the boot chain:

1. Measure $S_{i+1}$ by computing the hash of the executable code of $S_{i+1}$
2. Extend this measurement into the TPM’s Platform Configuration Register (PCR) by using the TPM command $PCR_{extend}(n, S_{i+1}) : PCR_{i+1}[n] = SHA1(PCR_i[n] || S_{i+1})$,
   where $n$ is the PCR index
3. Pass control to $S_{i+1}$

Additionally, each time a PCR is extended, a log entry, e.g. containing name, version and vendor of the software, is added to the TCG Event Log. This allows a challenger to see how the final PCR digests were generated.

The BIOS will perform the same steps before passing control to the bootloader. If the bootloader is TCG-enabled, like TrustedGRUB [Sir], it will continue by measuring important operating system files such as the kernel and kernel configuration files.

---

[2] for all TPM of version 1.2, the hash function is SHA-1
2.2.3 Authenticated vs. Secure Boot

After the operating system is booted, all measurements can be found in the corresponding PCRs. Usually, you have a set of PCR values describing the initial, authentic state of the system. With Authenticated Boot, you can now detect any changes to the platform by comparing authentic to current PCR values, e.g., taken immediately after system installation.

During Secure Boot, the authentic measurement values are compared to current measurements during the whole bootstrapping process and – if measurements differ – the boot process is cancelled and the system halted.

2.2.4 TrustedGRUB

TrustedGRUB [Sir] is an enhancement to the GNU GRUB [GNU] bootloader. It supports Authenticated Boot as well as Secure Boot using the Trusted Platform Module. TrustedGRUB, in contrast to the original GRUB bootloader, consists of two parts – called stages. This architecture of multiple stages is historically related. In the beginning of computer technology, the bootloader was designed to be placed at the first sector of the harddisk (head 0, track 0, sector 1) called Master Boot Record (MBR). The MBR is by convention only 512 bytes small. The bootloader must fit into the first 440 bytes of the MBR, followed by the partition table and additional disk signatures.

As the bootloaders added in more and more functionality, like support for network boot, developers were compelled to find a way circumventing the size limit of only 440 byte. Their solution was to split up the bootloader into several parts. A very small
part, stage1, is still located inside the Master Boot Record, whereas stage2 is located on secondary storage. Upon system boot, the CPU jumps to the MBR and executes stage1. Stage1 merely loads stage2, which holds the filesystem drivers, program code for the selection screen and the GRUB shell as well as the load routine for the operating system kernel. Stage2 would then load the operating system kernel and the bootstrapping process would continue.

Now for the whole bootloader, stage1 and stage2, to be measured by the BIOS, additional measurement functionality had to be added to stage1 of GRUB. When executed, stage1 would measure the first part of stage2 before passing control to it. The first part of stage2 would measure the rest of the stage2 and execute it. As the original stage1 did nearly fit the whole MBR, code enabling network boot had to be removed in favor of the measurement code. As a side note, the SHA-1 measurements in TrustedGRUB are not done by the TPM, as tests revealed that the TPM is very slow in measuring code of bigger size. Instead, the bootloader will compute the hash itself and extend it into the PCRs. This will not break the security model but only minimally enlarge the Trusted Computing Base, because the bootloader is already part of it.

**Extending Authenticated Boot to the Operating System**

Current TCG-enabled platforms feature a CRTM and a TCG-enabled BIOS. To extend the chain of measurements into the operating system, an enhanced bootloader that supports measurements is needed. TrustedGRUB is an enhanced bootloader that supports the measurement of the operating system kernel and arbitrary files on the disk.

**Measurements**

The measurements listed in figure 2.1 are performed by the TrustedGRUB bootloader and other bootstrapping components. $PCR[0] - PCR[7]$ are defined in TCG PC Client Specification [Gro], where all other PCRs can be used for additional measurements like, in our case, by TrustedGRUB.

**Checkfile**

TrustedGRUB measures important operating system files, such as the kernel to be booted, by default. Additionally, you can provide a list of files to be measured upon boot together with their authentic hash value. At the end of stage2, TrustedGRUB will measure the files and compare the measurements to the hash values given in the checkfile. If they do not match, which means the files must have been modified, the user will be warned and can decide whether to continue booting the system or not (Secure Boot). If the hashes match, stage2 will extend the measurements into $PCR[13]$. The syntax of the checkfile is as follows:

```
HASH (hdX,Y)/some/path/some.file
```

---

5Initial Program Loader, e.g., Bootloader
6Dynamic Root of Trust for Measurement
<table>
<thead>
<tr>
<th>PCR Index</th>
<th>PCR Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CRTM, BIOS and Platform Extensions</td>
</tr>
<tr>
<td>1</td>
<td>Platform Configuration</td>
</tr>
<tr>
<td>2</td>
<td>Option ROM Code</td>
</tr>
<tr>
<td>3</td>
<td>Option ROM Configuration and Data</td>
</tr>
<tr>
<td>4</td>
<td>IPL Code (MBR Information and Bootloader Stage1)</td>
</tr>
<tr>
<td>5</td>
<td>IPL Code and Configuration Data (for use by IPL Code)</td>
</tr>
<tr>
<td>6</td>
<td>State Transition and Wake Events</td>
</tr>
<tr>
<td>7</td>
<td>Reserved for future usage</td>
</tr>
<tr>
<td>8</td>
<td>Bootloader Stage2 Part 1</td>
</tr>
<tr>
<td>9</td>
<td>Bootloader Stage2 Part 2</td>
</tr>
<tr>
<td>10</td>
<td>Not in Use</td>
</tr>
<tr>
<td>11</td>
<td>Not in Use</td>
</tr>
<tr>
<td>12</td>
<td>Bootloader Commandline Arguments</td>
</tr>
<tr>
<td>13</td>
<td>Files checked via checkfile routine</td>
</tr>
<tr>
<td>14</td>
<td>Files which are actually loaded (e.g., Linux kernel, initrd, modules...)</td>
</tr>
<tr>
<td>15</td>
<td>Not in Use</td>
</tr>
<tr>
<td>16</td>
<td>Not in Use</td>
</tr>
<tr>
<td>17</td>
<td>DRTM</td>
</tr>
<tr>
<td>18-23</td>
<td>Not in Use</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of defined PCR Usage

The checkfile contains a list of entries that must not be larger than 8096 bytes altogether. The path of each entry (including the last one) must be succeeded by a newline (‘\n’).\footnote{\textsuperscript{5}}

In the last place, a checkfile entry must be added to each GRUB menu.lst-entry, giving the filesystem path of the checkfile. For an example entry, see listing 2.1.

Pay special attention that the drive parameter and the path are both correct, otherwise the system is not able to boot. In this case you can try again by editing the path temporarily or booting without the checkfile parameter (both via the edit functionality of Grub) and fix the problem.

```
1 title Debian 6 32 bit
2 root (hd0,0)
3 kernel /boot/vmlinuz-2.6.32-5-686 root=/dev/sda1 ro
4 initrd /boot/initrd.img-2.6.32-5-686
5 checkfile /boot/grub/checkfile
```

Listing 2.1: GRUB menu.lst example entry with checkfile option added

\footnote{Note: Some editors add a redundant newline at the end of the file, which must be removed (using, e.g., vbindiff or ghex tools).}
2.3 Secure Storage

Secure storage with the TPM is implemented as a hierarchy of keys. The root of this hierarchy is denoted Storage Root Key (SRK) – a TPM key of type storage key, whose private portion never leaves the TPM. The SRK is generated during the taking ownership of the TPM and deleted upon clearing the TPM.

Because of the size constraints of the internal memory of the TPM, newly created keys in the hierarchy have to be stored outside the TPM on persistent, untrusted storage. In order to protect the private portions of those keys from eavesdropping and modifications, this portion is "wrapped", i.e., encrypted with the public key of the parent key in the hierarchy. These parent keys must be of type storage key. Since the SRK never leaves the TPM and thus does not require wrapping, it does not have a parent key and forms the root of the hierarchy.

Storage keys can be further used to encrypt arbitrary data provided to the TPM. This procedure is denoted as sealing, since during encryption of the data a certain platform configuration can be declared, which defines the only state in which the sealed data can be decrypted (unsealed). The platform state is defined based on PCR values. Since storage keys cannot be migrated to other TPMs if used for sealing, the sealed data is bound to a specific state of a specific platform. Because the possible input size of the raw data for sealing is limited by the key size of the storage key (currently 2048 bits), usually a hybrid approach is taken to seal larger amounts of data: The data is encrypted symmetrically, e.g., using AES, and the symmetric key is sealed with the TPM. Thus, the data is indirectly sealed, since it can only be decrypted if the encryption key can be unsealed.

Another possibility to achieve binding of data to a particular platform state (but not necessary platform) is to use binding keys. Binding is equivalent to traditional asymmetric encryption and occurs outside the TPM. The TPM provides a special command to retrieve the public portion of a binding key outside the TPM, while the private portion is still only available inside the TPM in un-wrapped form. Thus, decryption of bound data is performed within the TPM, similar to sealing. However, binding keys can be migrated to other TPM and thus the data encrypted with a binding key can be migrated as well.


2.4 Theoretical Assignments (30 Points)

**Trust Computing Incentives (6 Points)**

1. The trusted computing base of an operating system consists of all hardware and software components that are necessary to define and enforce the security goals of the operating system. If a component of the TCB is compromised, the overall system security is affected. Software components of a computer platform running the Linux operating system are listed below. Identify the components that belong to the trusted computing base in the example below and justify your reasoning.

   - E-Mail client
   - Linux kernel modules
   - BIOS
   - Device drivers
   - Internet browser
   - Privileged applications (SUID)
   - Unprivileged applications
   - Boot loader
   - Linux kernel

2. Today’s modern operating systems have a very large and complex Trusted Computing Base. Explain why this is a disadvantage!

3. What are the desired primitives for Trustworthy Computing?

4. The Root of Trust for Measurement (RTM) is the root of the Chain of Trust. How can trust in the RTM be enforced?

**Integrity Measurement (8 Points)**

1. What is integrity measurement? How is integrity measurement realized in the TPM?

2. A PCR register is extended according to the following formula:

   \[
   PCR_{i+1}[j] \leftarrow SHA1(PCR_i[j]|data)
   \]

   where SHA-1 is a hash algorithm, \( data \) indicates the value (e.g., file, hash) to be measured and \( | \) means concatenation. What is the reason that PCR registers are extended by such a formula and that the designers have not used, for instance, the following formulas for the extension of the PCR register:
1. \(PCR_i[j] \leftarrow SHA_1(data)\)
2. \(PCR_i[j] \leftarrow SHA_1(PCR_i[j])\)
3. \(PCR_i[j] \leftarrow SHA_1(PCR_i[j] \oplus data)\)

3. Assume that the PCR registers could be reset by a command while the system is running. Which security problem arise if this would be possible?

4. Describe the meaning of the notions Secure Boot and Authenticated Boot! What are the differences between them?

**Authenticated Boot (8 Points)**

1. What is the CRTM? Which bootstrapping component can you still trust when the CRTM is tampered?
2. Where should the CRTM be located? Where is in current platforms usually implemented?
3. Should the TCG Event Log be protected, e.g., it’s integrity? Justify your answer!
4. TrustedGRUB measures important operating system files, such as the kernel, during the bootstrapping process. Can a kernel, such as the Linux kernel, change during runtime and which impact does this have on the measurements taken at boot time? Give an example!
5. The checkfile itself is not measured in any way. Think of the consequences! What could be done to protect the checkfile?

**Secure Storage (8 Points)**

1. What is the difference between binding and sealing?
2. If you use a non-migratable binding key for encrypting data, this data can be bound to a specific platform configuration (strictly speaking, the usage of the private portion of the key is bound to the platform configuration). Is this kind of binding with a non-migratable binding key equal to sealing?
3. How do you unseal data that was sealed with a storage key whose SRK was deleted during a TPM clearing?
4. What is an obvious drawback of sealing data to the measurement(s) (PCR values) of frequently changing data, e.g., the kernel.
2.5 Practical Assignments

2.5.1 Activating the TPM from BIOS

As the TPM is Opt-in, it is disabled by default. To use the Trusted Platform Module, it should be enabled and activated first.

1. Start the PC and press F2 to enter the BIOS.
2. In the “Security” tab you will find both “TPM Security” and the “TPM Activation” setting. Enable and activate the TPM!
3. The BIOS also allows you to clear the TPM. Should access to the BIOS be password-protected? Justify your answer!

2.5.2 Initializing the TPM

To manage the TPM using the TPM Manager software, you would want to start “TrouSerS” (see [Pro]) and the TPM Manager. TrouSerS is an open-source implementation of the Trusted Software Stack (TSS), used to provide a high-level interface for the TPM and TCG functionalities.

1. TrouSerS should already be running as a service in the background (tcsd). Verify this! (Hint: ps ax | grep tcsd)
   If not, start it as root: sudo tcsd
2. Initialize the TPM using the TPM Manager!

2.5.3 Taking Ownership

After initializing the TPM you should now also take ownership in order to use all features of the TPM, using the TPM Manager.

1. You will be asked to enter two passwords. Use “tpmtest” for the first. What is the name of this password and what is it used for in the Trusted Computing Model.
2. When asked, set the SRK password to “WELL_KNOWN_PASSWORD”, which corresponds to 20 bytes of zeros.

2.5.4 Reading the PCR

Authenticated boot is one of the main features in the Trusted Computing Model. The TPM acts as a root of trust for the bootstrapping process. The boot components establish a Chain of Trust, where each component being executed is measured by it’s

---

6The TPM is disabled by default, using it is not mandatory
predecessor before passing control to it and the measurement value will then be extended into one of the 24 PCRs (TPM v1.2). Measurement is done by computing a 160 bit SHA-1 cryptographic hash of the binary. The extend function ensures that registers cannot be overwritten by including the old PCR value in the new measurement value being written into the register.

1. Read the contents of the PCRs using the TPM manager!
   - Info → PCRs

2. Read the contents of the PCRs from the terminal!
   - `cat /sys/class/misc/tpm0/device/pcrs`

### 2.5.5 Using TrustedGRUB

TrustedGRUB is already installed on your system. It operates in “SHA-1 Mode”, showing the measured SHA-1 result for every measured file loaded on the screen while loading the operating system to boot.

1. You can find all the TrustedGRUB files in the `/boot/grub/` directory.

2. The `menu.lst` file holds a list of all operating systems managed by TrustedGRUB. List the operating systems managed by TrustedGRUB on your platform and note on which partition they have been installed. Use the GRUB notation `(hdX,Y)`!

### 2.5.6 Measurements

In the following task, we let TrustedGRUB perform the measurements of important operating system files and validate the PCR value using a command line tool afterwards.

TrustedGRUB performs measurements of important operating system files being loaded. This includes the kernel and the initrd-image\(^7\). These measurements all are extended into `PCR[14]`.

1. Check the measurements using the TrustedGRUB tool `verify_pcr`. Note both the PCR14 value and the output of `verify_pcr`.

   Usage: `verify_pcr <pcr initial value {NULL | 20 byte hex}>{filename-1 ... filename-n}`, see also appendix.

2. When measuring and extending more than one file into a PCR register, the order is very important. Explain why with regards to the order of the chain of trust!

\(^7\)initial ramdisk; temporary filesystem loaded by the kernel during boot, contains files needed for startup of the system. E.g., `/boot/vmlinuz-XY`, `/boot/initrd.img-XY` (XY: kernel version)

11
2.5.7 Using the Checkfile

In the following task we add two files to the TrustedGRUB checkfile, let TrustedGRUB perform measurements of these additional files and validate the corresponding PCR value using a command line tool afterwards.

1. Create the checkfile inside `/boot/grub/` and add the system files `/etc/passwd` and `/etc/shadow` to the checkfile. Use the syntax mentioned in the introduction and don’t forget to add a new line after each entry (including the last one)!

2. Compute the SHA-1 hash of both of these files and add it to the corresponding entry in the checkfile. You can either use the `create_sha1` tool of TrustedGRUB or the `openssl` package (see `man openssl`). Note both entries!

3. Now, reboot the system.

4. In which Platform Configuration Register (PCR) can you find the measurements of the files in your checkfile?

5. Now, verify the measurements of `/etc/passwd` and `/etc/shadow`. Use the `verify_pcr` tool provided by TrustedGRUB and compare the output with the corresponding PCR value. Note both values!

6. Make a change to either of these files, e.g., set a new password for user “laborratte” by executing `sudo passwd laborratte`. Now, reboot again. What happens? Again, verify the PCR and note both values.

7. Suppose you are an attacker that wants to tamper the system. Your previous modification of the `/etc/passwd` respectively `/etc/shadow` file could be such a scenario.
   a) Are your system modifications detected? Justify your answer!
   b) How could you hide your modifications, so they cannot be detected?

2.5.8 Un-/Sealing

1. Create a file with content that you easily recognize (e.g., `echo 'foobar' >> foo.file`)

2. Seal this file and bind the data to PCR15

3. Unseal the data into a new file, e.g., `foo.unseal1`, and verify that the unsealed data is correct

4. Extend PCR15 with the `measure_file` tool (pre-installed)

5. Unseal the data again. Why didn’t it work?

6. Would the sealing also work if you would take a bigger file as input, i.e., with more than 256 bytes size?
2.5.9 Clearing the TPM

You want to sell your computer, because you just bought a new model with double the speed. Of course, you want to erase all personal data on the hard drive first.

1. Should you also clear the TPM? Justify your answer!

2. Clear the TPM using the TPM Manager or execute `tpm_clear`!

3. What happens during clearing of the TPM?

Appendix

`create_sha1` - takes as an argument a filename and writes the result of the SHA-1 function of the file to `stdout`

`verify_pcr` - intended to check, if a PCR register is extended correctly with the given files. For example, if you have a checkfile containing 5 entries, all the files are hashed and extended into `PCR[13]`. The corresponding value of `PCR[13]` can be verified with this utility. Execute the command with the following parameters:

Usage: `verify_pcr <pcr initial value {NULL | 20 byte hex}> {filename-1 ... filename-n}`
Abbreviations

TPM  Trusted Platform Module
PCR  Platform Configuration Register
TCG  Trusted Computing Group
MAC  Message Authentication Code
TCB  Trusted Computing Base
RTM  Root of Trust for Measurement
BIOS Basic Input Output System
OS   Operating System
SRK  Storage Root Key
EK   Endorsement Key
TSS  Trusted Software Stack
API  Application Programming Interface
TSP  TSS Service Provider
Tspi TSP interface
TCS  TCG Core Service
TDDL TSS Device Driver Library
TDD  TPM Device Driver
CRTM Core Root of Trust for Measurement
DRTM Dynamic Root of Trust for Measurement
SRTM Static Root of Trust for Measurement
MBR  Master Boot Record
DAA  Direct Anonymous Attestation
RTR  Root of Trust for Reporting
AIK  Attestation Identity Key
TTP  Trusted Third Party
ZKP  Zero Knowledge Proof
NMK  Non-Migratable Key
MK   Migratable Key
CMK  Certified Migratable Key
MA   Migration Authority
MSK  Migration Selection Authority
KCM  Key Cache Manager
References

https://www.trustedcomputinggroup.org/specs/

[GNU] GNU. GNU GRUB website.
http://www.gnu.org/software/grub/

https://www.trustedcomputinggroup.org/specs/PCClient

http://trousers.sourceforge.net/

[Sir] Sirrix. Trustedgrub project page.
http://www.sirrix.de/content/pages/50553.htm