# Trusted Computing and Linux

Kylene Hall
*IBM*
kylene@us.ibm.com

Tom Lendacky
*IBM*
toml@us.ibm.com

Emily Ratliff
*IBM*
emilyr@us.ibm.com

Kent Yoder
*IBM*
yoder1@us.ibm.com

## Abstract

While Trusted Computing and Linux® may seem antithetical on the surface, Linux users can benefit from the security features, including system integrity and key confidentiality, provided by Trusted Computing. The purpose of this paper is to discuss the work that has been done to enable Linux users to make use of their Trusted Platform Module (TPM) in a non-evil manner. The paper describes the individual software components that are required to enable the use of the TPM, including the TPM device driver and TrouSerS, the Trusted Software Stack, and TPM management. Key concerns with Trusted Computing are highlighted along with what the Trusted Computing Group has done and what individual TPM owners can do to mitigate these concerns. Example beneficial uses for individuals and enterprises are discussed including eCryptfs and GnuPG usage of the TPM. There is a tremendous opportunity for enhanced security through enabling projects to use the TPM so there is a discussion on the most promising avenues.

## 1 Introduction

The Trusted Computing Group (TCG) released the first set of hardware and software specifications shortly after the creation of that group in 2003.[1] This year, a short two years later, 20 million computers will be sold containing a Trusted Platform Module (TPM) [Mohamed], which will largely go unused. Despite the controversy surrounding abuses potentially enabled by the TPM, Linux has the opportunity to build controls into the enablement of the Trusted Computing technology to help the end user control the TPM and take advantage of security gains that can be made by exercising the TPM properly. This paper will cover the pieces needed for a Linux user to begin to make use of the TPM.

This paper is organized into sections covering the goals of Trusted Computing, a brief introduction to Trusted Computing, the components required to make an operating system a trusted operating system from the TCG perspective, the current state of Trusted Computing, uses of the TPM, clarification of common technical misperceptions, and finally concludes with

[1] See [Fisher] and [TCGFAQ] for more history of the Trusted Computing Group.

a section on future work.

## 2 Goals of Trusted Computing

The Trusted Computing Group (TCG) has created the Trusted Computing specifications in response to growing security problems in the technology field.

"The purpose of TCG is to develop, define, and promote open, vendor-neutral industry specifications for trusted computing. These include hardware building block and software interface specifications across multiple platforms and operating environments. Implementation of these specifications will help manage data and digital identities more securely, protecting them from external software attack and physical theft. TCG specifications can also provide capabilities that can be used for more secure remote access by the user and enable the user's system to be used as a security token."[TCGBackground]

Fundamentally, the goal of the Trusted Computing Group's specifications is to increase assurance of trust by adding a level of verifiability beyond what is provided by the operating system. This does not reduce the requirement for a secure operating system.

## 3 Introduction to Trusted Computing

The Trusted Computing Group (TCG) has released specifications about the Trusted Platform Module (TPM), which is a "smartcard-like device," one per platform, typically realized in hardware that has a small amount of both volatile and non-volatile storage and cryptographic execution engines. Figure 1 shows

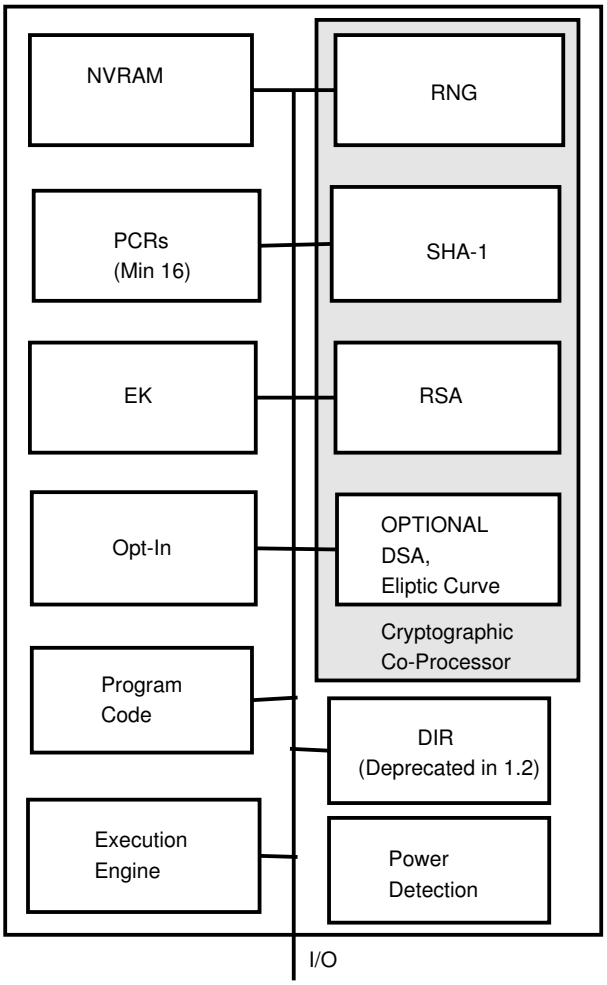


Figure 1: Trusted Platform Module

a logical view of a TPM. The TCG has also released a specification for APIs to allow programs to interact with the TPM. The next section details the components needed to create a completely enabled operating system. The interaction between the components is graphically shown in Figure 2.

For a rigorous treatment of Trusted Computing and how it compares to other hardware security designs, please read Sean W. Smith's "Trusted Computing Platforms Design and Applications" [Smith:2005].

### 3.1 Key Concepts

There are a few key concepts that are essential to understanding the Trusted Computing specifications.

#### 3.1.1 Measurement

A measurement is a SHA-1 hash that is then stored in a Platform Configuration Register (PCR) within the TPM. Storing a value in a PCR can only be done through what is known as an extend operation. The extend operation takes the SHA-1 hash currently stored in the PCR, concatenates the new SHA-1 value to it, and performs a SHA-1 hash on that concatenated string. The resulting value is then stored in the PCR.

#### 3.1.2 Roots of Trust

In the Trusted Computing Group's model, trusting the operating system is replaced by trusting the roots of trust. There are three roots of trust:

- root of trust for measurement
- root of trust for storage
- root of trust for reporting

The root of trust for measurement is the code that represents the "bottom turtle"[2]. The root of trust for measurement is not itself measured; it is expected to be very simple and immutable. It is the foundation of the chain of trust. It performs an initial PCR extend and then the performs the first measurement.

[2]This is an allusion to the folk knowledge of how the universe is supported. `http://en.wikipedia.org/wiki/Turtles_all_the_way_down`

The root of trust for storage is the area where the keys and platform measurements are stored. It is trusted to prevent tampering with this data.

The root of trust for reporting is the mechanism by which the measurements are reliably conveyed out of the root of trust for storage. This is the execution engine on the TPM.[TCGArch]

#### 3.1.3 Chain of Trust

The chain of trust is a concept used by trusted computing that encompasses the idea that no code other than the root of trust for measurement may execute without first being measured. This is also known as transitive trust or inductive trust.

#### 3.1.4 Attestation

Attestation is a mechanism for proving something about a system. The values of the PCRs are signed by an Attestation Identity Key and sent to the challenger along with the measurement log. To verify the results, the challenger must verify the signature, then verify the values of the PCRs by replaying the measurement log.

#### 3.1.5 Binding Data to a TPM

Bound data is data that has been encrypted by a TPM using a key that is part of the root of trust for storage. Since the root of trust of storage is different for every TPM, the data can only be decrypted by the TPM that originally encrypted the data. If the key used is a migratable key, however, then it can be migrated to the root of trust for storage of a different TPM allowing the data to be decrypted by a different TPM.

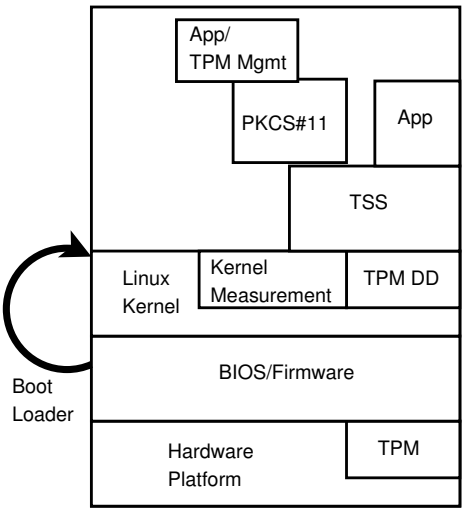


Figure 2: Trusted Computing Enabled Operating System

### 3.1.6 Sealing Data to a TPM

Sealed data is bound data that additionally records the values of selected PCRs at the time the data is encrypted. In addition to the restrictions associated with bound data, sealed data can only be decrypted when the selected PCRs have the same values they had at the time of encryption.

## 4 Components of Trusted Computing on Linux

Several components are required to enable an operating system to use the Trusted Computing concepts. These components are described in this section.

### 4.1 TPM

The Trusted Platform Module (TPM) is a hardware component that provides the ability to securely protect and store keys, certificates, passwords, and data in general. The TPM enables more secure storage of data through asymmetric key operations that include on-chip key generation (using a hardware random number generator), and public/private key pair encryption and signature operations. The TPM provides hardware-based protection of data because the private key used to protect the data is never exposed in the clear outside of the TPM. Additionally, the key is only valid on the TPM on which it was created unless created migratable and migrated by the user to a new TPM.

The TPM provides functionality to securely store hash values that represent platform configuration information. The secure reporting of these values, if authorized by the platform owner, enables verifiable attestation of a platform configuration. Data can also be protected under these values, requiring the platform to be in the same configuration to access the data as when the data was first protected.

The owner of the platform controls the TPM. There are initialization and management functions that allow the owner to turn on and off functionality, reset the TPM, and take ownership of the TPM. There are strong controls to protect the privacy of an owner and user.[3] The platform owner must opt-in. Any user, even if different from the owner, may opt-out.

Each TPM contains a unique Endorsement Key. This key can be used by a TPM owner to anonymously establish Attestation Identity Keys (AIKs). Since privacy concerns prevent the Endorsement Key from being used to sign data generated internally by the TPM, an AIK is used. An AIK is an alias to the Endorsement Key. The TPM owner controls the creation and activation of an AIK as well as the data associated with the AIK.[TCGMain],[TPM]

[3] See Section 7.1 for more details.

#### 4.1.1 A Software-based TPM Emulator for Linux

If you don't have a machine that has a TPM but you'd like to start experimenting with Trusted Computing and the TSS API, a software TPM emulator can provide a development environment in which to test your program. While a software TPM will provide you with a development environment, it can't provide you with the "trust" that a hardware TPM can provide.

The advantage of having the TPM be a hardware component is the ability to begin measuring a system almost immediately at boot time. This is the start of the "chain of trust." By measuring as early in the boot cycle as possible, you lessen the chance that an untrusted component (hardware or software) can be introduced without being noticed. There must be an initial "trusted" measurement established, known as the root of trust for measurement, and the measurement "chain" must not be interruped.

With a software TPM emulator, you have delayed the initial measurement long into the boot cycle of the system. Many measurements have not occurred and so the trust of the system can not be fully validated. So while you would not want to rely on a software TPM to validate the trust of your systemm, it does provide you with a development environment to begin preparing to take advantage of trusted computing.

Mario Sasser, a student at the Swiss Federal Institute of Technology has created a TPM emulator that runs as a kernel module.[Strasser] It is not a full implementation of the specification and it is still under development. It is available from `http://www.infsec.ethz.ch/people/psevinc/` or `https://developer.berlios.de/projects/tpm-emulator`.

### 4.2 TPM Device Driver

The TPM device driver is a driver for the Linux kernel to communicate TPM commands and their results between the TCG Software Stack (TSS) and the TPM device. Today's TPMs are connected to the LPC bus. The TPM hardware is located by the driver from the PCI device for the LPC bus and attempts to read manufacturer specific information at manufacturer specific offsets from the standard TPM address. Since the TPM device can only handle one command at a time and the result must be cleared before another command is issued, the TPM device driver takes special care to provide that only one command is in-flight at a time and that the data is returned to only the requester. Rather than tie up all system resources with an ioctl, the command is transmitted and the result gathered into a driver buffer on a write call. Then the result is copied to the same user on a subsequent read call. This coupling of write and read calls is enforced by locks, the file structure's private data pointer and timeouts. At the direction of the Trusted Computing Group Specification, the TSS is the only interface allowed to communicate with the TPM thus, the driver only allows one open at a time, which is done by the TSS at boot time. The driver allows canceling an in-flight command with its sysfs file `cancel`. Other sysfs files provided by the driver are `pcrs` for reading current pcr values, `caps` for reading some basic capability information about the TPM such as manufacturer and version and `pubek` for reading the public portion of the Endorsement Key if allowed by the device. The current driver supports the Atmel and National Semiconductor version 1.1 TPMs, which are polled to determine when the result is available. The common functionality of the driver is in the `tpm` kernel module, and the vendor specifics are in a separate module. The driver is available on Sourceforge at `http://sourceforge.`

`net/projects/tpmdd/` under the project name tpmdd and has been in Linux kernel versions since 2.6.12.

### 4.3 TSS

The TCG Software Stack (TSS) is the API that applications use to interface with the TPM.

#### 4.3.1 TSS Background

The TCG Software Stack (TSS)[TSS] is the set of software components that supports an application's use of a platform's TPM. The TSS is composed of a set of software modules and components that allow applications to communicate with a TPM.

The goals of the TSS are:

- Supply one entry point for applications to the TPM's functionality. (Provided by the TSS Service Provider Interface (The TSS API)).
- Provide synchronized access to the TPM. (Provided by the TSS Core Services Daemon(TCSD)).
- Hide issues such as byte ordering and alignment from the application. (Provided by the TSS Service Provider Interface (TSPI)).
- Manage TPM resources. (Provided by the TCSD).

All components of the TSS reside in user space, interfacing with the TPM through the TPM device driver.

TPM services provided through the TSS API are:

- RSA key pair generation
- RSA encryption and decryption using PKCS v1.5 and OAEP padding
- RSA sign/verify
- Extend data into the TPM's PCRs and log these events
- Seal data to arbitrary PCRs
- RNG
- RSA key storage

Applications will link with the TSP library, which provides them the TSS API and the underlying code necessary to connect to local and remote TCS daemons, which manage the resources of an individual TPM.

#### 4.3.2 The TrouSerS project

The TrouSerS project aims to release a fully TSS 1.1 specification compliant stack, following up with releases for each successive release of the TSS spec. TrouSerS is released under the terms of the Common Public License, with a full API compliance test suite and example code (both licensed under the GPL) and documentation. TrouSerS was tested against the Atmel TPM on i386 Linux and a software TPM on PPC64. TrouSerS is available in source tarball form and from CVS at `http://trousers.sf.net/`.

#### 4.3.3 Technical features not in the TSS specification

By utilizing udev.permissions for the TPM device file and creating a UID and GID just for the TSS, the TrouSerS TCS daemon runs without root owned resources.

For machines with no TPM support in the BIOS, TrouSerS supports an application level interface to the physical presence commands when the TCS daemon is executing in single user mode. This allows administrators to enable, disable, or reset their TPMs where a BIOS/firmware option is not available. This interface is automatically closed at the TCS level when the TCS daemon is not running in single user mode, or cannot determine the run level of the system.

In order to maintain logs of all PCR extend operations on a machine, TrouSerS supports a pluggable interface to retrieve event log data. Presumably, the log data would be provided by the Integrity Measurement Architecture (IMA) (see Section 4.6 below). As executable content is loaded and extended by the kernel, a log of each extend event is recorded and made available through sysfs. The data is then retrieved by the TCS Daemon on the next GetPcrEvent API call.

To maintain the integrity of BIOS and kernel controlled PCRs, TrouSerS supports configurable sets of PCRs that cannot be extended through the TSS. This is useful; for example in keeping users from extending BIOS controlled PCRs or for blocking access to an IMA controlled PCR.

TCP/IP sockets were chosen as the interface between TrouSerS' TSP and TCS daemon, for both local and remote access. This makes connecting to a TCSD locally and remotely essentially the same operation. Access control to the listening socket of the TCSD should be controlled with firewall rules. Access controls to the TCSD's internal functionality was implemented as a set of 'operations,' each of which enable a set of functions to be accessible to a remote user that will enable that user to accomplish the operation. For instance, enabling the seal operation allows a remote user to open and close a context, create authorization sessions, load a key, and seal data. By default, all functionality is available to local users and denied to remote users.

### 4.4 TPM Management

Some TPM management functionality was implemented in the tpm-mgmt package and the openCryptoki package. The tpm-mgmt package contains support for controlling the TPM (enabling, activating, and so on) and for initializing and utilizing the PKCS#11 support that is provided in the openCryptoki package.

#### 4.4.1 Controlling the TPM

The owner of the platform has full control of the TPM residing on that platform. A TPM maintains three discrete states: enabled or disabled, active or inactive, and owned or unowned. The platform owner controls setting these states. These states, when combined, form eight operational modes. Each operational mode dictates what commands are available.

Typically, a TPM is shipped disabled, inactive and unowned. In this operational mode, a very limited set of commands is available. This limited set of commands consists mainly of self-test functions, capability functions and non-volatile storage functions. In order to take full advantage of the TPM, the platform owner must enable, activate, and take ownership of the TPM. Enabling and activating the TPM is typically performed using the platform BIOS or firmware. If the BIOS or firmware does not provide this support, but the TPM allows for the establishment of physical presence through software, then TrouSerS can be used to establish physical presence and accomplish the task of enabling and activating the TPM. Taking

ownership of the TPM sets the owner password, which is required to execute certain commands.

The tpm-mgmt package contains the commands that are used to control the TPM as described above, as well as perform other tasks.

### 4.4.2 PKCS#11 Support

The PKCS#11 standard defines an API interface used to interact with cryptographic devices. Through this API, cryptographic devices are represented as tokens, which provide applications a common way of viewing and accessing the functionality of the device. Providing a PKCS#11 interface allows applications that support the PKCS#11 API to take advantage of the TPM immediately.

The TPM PKCS#11 interface is implemented in the openCryptoki package as the TPM token. Each user defined to the system has their own private TPM token data store that can hold both public and private PKCS#11 objects. All private PKCS#11 objects are protected by the TPM's root of trust for storage. A symmetric key is used to encrypt all private PKCS#11 objects. The symmetric key is protected by an asymmetric TPM key that uses the user's PKCS#11 user login PIN as the key's authorization data. A user must be able to successfully login to the PKCS#11 token in order to use a private PKCS#11 object. The TPM token provides key generation, encryption and signature operations through the RSA, AES, triple DES (3DES), and DES mechanisms.

The following RSA mechanisms are supported (as defined in the PKCS#11 Cryptographic Token Interface Standard[PKCS11]):

- PKCS#1 RSA key pair generation
- PKCS#1 RSA
- PKCS#1 RSA signature with SHA-1 or MD5

The following mechanisms are supported AES, 3DES, and DES (as defined in the PKCS#11 Cryptographic Token Interface Standard[PKCS11]):

- Key generation
- Encryption and decryption in ECB, CBC or CBC with PKCS padding modes

The RSA mechanisms utilize the TSS to perform the required operations. By utilizing the TSS, all RSA private key operations are performed securely in the TPM. The symmetric mechanisms are provided to allow for the protection of data through symmetric encryption. The symmetric key used to protect the data is created on the TPM token and is thus, protected by the TPM. Since the key is protected by the TPM, the data is protected by the TPM.

Before any PKCS#11 token is able to be used it must be initialized. Since each user has their own TPM token data store, each user must perform this intialization step. Once the data store is initialized it can be used by applications supporting the PKCS#11 API.

The tpm-mgmt package contains commands to initialize the TPM token data store as well as perform other tasks. Some of the other tasks are:

- Import X509 certificates and/or RSA key pairs

  Existing certificates and/or key pairs can be stored in the data store to be used by applications.

- List the PKCS#11 objects in the data store

  In addition to any objects that you import, applications may have created or generated objects in the data store. tpm-mgmt lets you get a list of all the PKCS#11 objects that exist in the data store.

- Protect data using the "User Data Protection Key"

  Protect data by encrypting it with a random 256-bit AES key. The key is created as a PKCS#11 secret key object with an label attribute of "User Data Protection Key." This label attribute is used to obtain a PKCS#11 handle to the key and perform encryption, or decryption operations on the data.

- Change the PKCS#11 PINs (Security Officer and User)

  PKCS#11 tokens have security officer and a user PINs associated with them. It may be necessary or desirable to change one or both of these PINs at some point in time.

### 4.5 Boot Loader

To preserve the chain of trust beyond the boot loader, the boot loader must be instrumented to measure the kernel before it passes over control. The root of trust for measurement measured the BIOS before it transferred control, the BIOS measured the boot loader. Now the boot loader must measure the kernel. Seiji Munetoh and Y. Yamashita of IBM's Tokyo Research Lab have instrumented Grub v.0.94 and v.0.96 to perform the required measurements. Since Grub is a multi-stage boot loader, each stage measures the next before it transfers control. This is a slight simplification. Stage 1 is measured by the BIOS. Stage 1 measures the first sector of stage 1.5, which then measures the rest of stage 1.5 and stage 2. The configuration file is measured early with additional measurements of files referred to in configuration files taken in sequence. If stage 1.5 is not loaded, stage 1 measures the first sector of stage 2 instead and that sector measures the rest of stage 2. The grub measurements are stored in PCR 4, the grub configuration file measurement is stored in PCR 5, and the kernel measurement is stored in PCR 8. The PCRs used are configurable but the defaults meet the requirements of the TCG PC Specific Implementation Specification Version 1.1[TCGPC].

Lilo has also been instrumented to take the measurements by the Dartmouth Enforcer team. (See more detail about this project in Section 6.1.1).

### 4.6 Kernel Measurement Architecture—IMA

Reiner Sailer, and others of the IBM T.J. Watson Research Center have extended the chain of trust to the Linux kernel by implementing a measurement architecture for the kernel as a LSM.[SailerIMA] (Note: To effectively preserve the chain of trust, the LSM must be compiled into the kernel rather than dynamically loaded.) The `file_mmap` hook is used to perform the measurement on anything that is mapped executable before it is loaded into virtual memory. Kernel modules are measured just before they are loaded. The measurement is used to extend one of the PCRs numbered between 7-16, as configured in the kernel to allow for somewhat flexible PCR use. PCR 9

is the default. Other files that are read and interpreted, such as bash scripts or Apache configuration files, require application modifications to measure these files. Measurements are cached to reduce performance impact. Performance, usability, and bypass-protection are all addressed in the Sailer, et al. report. Enforcement is not part of this architecture. The measurements (and measurement log) are intended to be used by the challenger during remote attestation to determine the integrity of the system, rather than by the system to enforce a security policy.

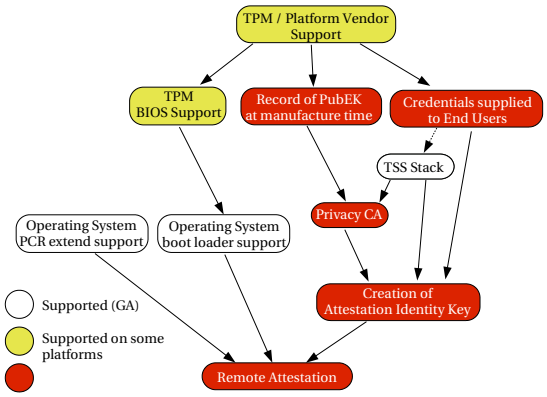


Figure 3: Dependencies for Full Trusted Computing Deployment

## 5 Trusted Computing on Linux Now and in the Future

Although version 1.1 TPMs provide many features usable today, significant hurdles exist to deploying the full capabilities of Trusted Computing outside a structured or corporate environment. Software that exists today basically enables the use of a TPM as one would use a smartcard. Other features, such as remote attestation, have more extensive requirements. The components required to implement remote attesation can be seen in Figure 3.

In order to implement remote attestation, TPM and Platform Vendor Support is required to:

- Put TPM support in the BIOS of shipping platforms (currently shipping).
- Record the Public Endorsement Key in some way (such as make a cryptographic hash) in order to identify whether a platform has a true TPM.
- Create and ship the TPM credential and the platform credential.

As long as the platform vendor has included TPM support in the BIOS, a corporate environment can work around the lack of the other elements by recording the PubEK as machines are deployed and maintaining a PKI internally. However, in order to enable remote attestation for general use by the public, a new infrastructure among hardware and software vendors must be created. This infrastructure would provide the credentials and a hash of the PubEK of shipping systems to Privacy CAs. The Privacy CAs differ from existing CAs in the key creation, certificate application, and certificate delivery mechanism, so new CAs are needed or existing CAs must implement the required software and procedures. At best, shipping platforms that fully support remote attestation are years away. Because of the lack of this infrastructure, no currently shipping platforms will have the capability to provide remote attestation for general use.

To make use of the more advanced features the TPM can provide, in addition to the infrastructure element listed above, a Linux distro would need to:

- Incorporate the measurement architecture

into the kernel.

- Ship measurement support for the boot loader.
- Include TrouSerS or some TCG Software Stack.
- Include attestation software.
- Include software for safe handling of the TPM and Platform credentials.

When this level of TPM hardware support is achieved, the groundwork will be laid to enable the software that will be used for attestation. Ideas for an interoperable attestation interface include a stand-alone attestation daemon and a modified TLS protocol that includes attestation. Until one of these solutions is specified and implemented, attestation solutions are ad hoc at best.

Finally, before general purpose remote attestation can be widely used, tools and best practice guidelines are needed to help define valid policies and maintain policy currency. Depending on the measurement architectures implemented by various operating systems, the policy becomes quite complex very quickly.

# 6 Example Uses of the TPM

Given the passive nature of the TPM device, the decision about its usefulness rests almost entirely on how one will use the device. Many of the doomsday scenarios surrounding the TPM device are based on scenarios involving software that Linux users will never agree to run on their hardware. In this section, some of the most promising uses of the TPM device are addressed. See also "Interesting Uses of Trusted Computing"[Anonymous] and "The Role of TPM in Enterprise Security"[Sailer] for more discussion around this topic. Anonymous notes in the first article "Before wide-scale use of TC for DRM, it will be necessary for the manufacturers, software vendors and content providers to get past a few tiny details, like setting up a global, universal, widely trusted and secure PKI. Hopefully readers . . . will understand that this is not exactly a trivial problem." The uses discussed below do not depend on full deployment of a complete Trusted Computing infrastructure but only on existing capabilities.

## 6.1 Beyond Measurement –Enforcement

A couple of examples of how the Trusted Computing measurements can be used to enforce a security policy exist and are described in this section.

### 6.1.1 Dartmouth's Enforcer

Enforcer is an LSM that measures each file as it is opened.[MacDonald] The measurement is compared against a database of previous measurements. File attributes (mtime, inode number, and so on) are also inspected. If the file has changed, the system will either log the condition, deny access to the file, panic the kernel, or "lock" the TPM (by extending the PCR used by Enforcer with random data, which makes decrypting data sealed to this PCR fail) based on the setting selected by the administrator. Enforcer does not require a TPM, but can optionally use the TPM to protect the database and configuration files. Enforcer also provides `helper`, which allows encrypting a loopback file system with a key protected by the TPM. Enforcer is available at `http://sourceforge.net/projects/enforcer/`.

### 6.1.2 Trusted Linux Client

The IMA kernel measurement architecture described previously provides no direct enforcement mechanisms. Dave Safford of IBM's T.J. Watson Research Center has proposed an extension to the concept that includes enforcement. The idea is to provide a series of LSMs that provide authenticated boot, encrypted home directories and file attribute checking. The first module validates the integrity of initrd and the kernel, and releases a TPM based kernel symmetric key. The key is used to derive keys for encrypted home directories via loopback file system and authenticated file attribute checking. The next module deals with extended attributes that are applied to every file including a file hash, MAC label, and others. The derived symmetric key is used to HMAC these attributes, and the value is checked and cached once at open/execute. A final module provides LOMAC style mandatory access control. See the presentation 'Putting Trust into Computing: Where does it Fit? —RSA Conference 2005' for an overview of this concept.[TCGRSA]

## 6.2 Enterprise Uses

Since the Trusted Computing Group is an industry led standards organization it is no surprise that compelling use cases exist for the enterprise.

### 6.2.1 Network attach

Enterprise networks are often described as 'hard and crunchy on the outside, but soft and chewy on the inside' reflecting the fact that they typically have very good perimeter defenses, but are less well protected from the inside. This poses a problem for enterprises that allow their employees to take mobile computing devices on the road and connect to non-protected networks. Viruses very often use unprotected mobile devices as a gateway device through which to invade corporate networks. To defeat viruses and worms that come in this way, more internal firewalls and choke points are architected into the network. A few vendors are now offering compliance checking software that challenge mobile devices when they attempt to reattach to the internal network; this is to prove that they meet corporate guidelines before allowing them to attach. This is typically done through agent software running on the mobile device. The agent software becomes the logical attack point.

Trusted Computing can make this compliance checking stronger. The Trusted Network Connect (TNC) subgroup of the Trusted Computing Group has released a specification[TNC] for client and server APIs that allow development of plugins for existing network attach products to do client integrity measurement and server-side verification of client integrity. The plugins add remote attestation capabilities to existing network attach products. The products continue to operate in their normal manner with the assurance that the client agents have not been subverted.

### 6.2.2 Systems management

Remote attestation is extremely useful when combined with systems management software. System integrity of the managed system is verified through remote attestation periodically, or on demand. Tied into the intrusion detection system, systematic integrity checking ensures that compromises can be quickly detected.

### 6.2.3 Common Criteria Compliance

Common Criteria evaluations are based on a well-defined and usually strict Security Target. Installing new software may cause the system to flip to an unevaluated mode. Remote attestation can be employed to confirm that all systems that are required to be Common Criteria compliant, retain adherence to the Security Target. This is one of the simpler uses of remote attestation since the policy to which the client must adhere is so static, strict, and well-defined that it eliminates the need for much policy management.

## 6.3 Uses by Individuals

The TPM can also be used to secure individual's computer and data.

### 6.3.1 TPM Keyring

The TPM Keyring application illustrates usage of the TSS API, and some of the properties of keys created with a TPM. TPM Keyring is licensed under the GPL and contains examples of how to wrap a software generated key with a TPM key, connect to local and remote TCS daemons, store and retrieve keys and encrypt and decrypt data using the TSS API. The source is available from CVS at `http://trousers.sf.net/`. TPM Keyring will wrap a software generated OpenSSL key with the Storage Root Key (SRK) of an arbitrary number of users. Once each user has a copy of this wrapped key, all users of the keyring can send secure messages to one another, but no user can give the key to anyone else, except the owner of the original OpenSSL key. Scripts are also provided to easily encrypt a symmetric key and use OpenSSL to encrypt large files.

You can imagine using `tpm_keyring` itself, or the concepts presented by `tpm_keyring` to create private and secure peer-to-peer networks.

**Creating a New Keyring** `tpm_keyring` generates a plaintext RSA key pair in memory and wraps the private key of that key with the public key of your TPM's root key. The plaintext RSA key is then encrypted with a password (that you are prompted for), and written to disk. Once the new key ring is created, you should move the encrypted software generated key to a safe place off your machine.

**Adding Members to a Keyring** After you've created a keyring, you'll probably want to add members so that you can start exchanging data. You'll need to bring your encrypted key file out of retirement from off-site backup in order to wrap it with your friend's TPM's root key. Contact this person and ask for their IP address or hostname. The public portion of your friend's root key will be pulled out of their TCS daemon's persistent storage and used to wrap your plaintext key. The resulting encrypted key is stored in your friend's persistent storage, with a UUID generated by hashing the name of the keyring you created. Let your friend know the name you gave the keyring so that they can import the key.

**Importing a Key** Once a friend has stored their key in your persistent store, you can import it so that `tpm_keyring` can use it. Run the import command with the same name of the key ring that your friend created and some hostname and alias pair to help you remember the friend who's keyring you're joining.

#### 6.3.2 eCryptfs

The need for disk encryption is often overlooked but well motivated by security events in the news; for example, this article at `http://sanjose.bizjournals.com/sanjose/stories/2005/04/04/daily47.html` about a physical theft compromising 185,000 patients' medical records. eCryptfs [Halcrow:2005], being presented at OLS 2005 by Michael Halcrow, offers as an option, file encryption using TPM keys. In the case mentioned, if the information on disk had been encrypted with a TPM key, the data would not have been recoverable by the thief. Hot swappable drives and mobile storage being so easy to remove, in particular, benefit from encryption tied to a TPM key.

#### 6.3.3 mod_ssl

Another use for the TPM is to provide secure storage for SSL private keys. Many system administrators face a problem of securely protecting the SSL private key and still being able to restart a web server as needed without human interaction. With the TPM, the private key can be bound, or optionally, sealed to a certain set of PCRs allowing it to be unsealed as necessary for starting SSL in a trusted environment on the expected platform.

#### 6.3.4 GnuPG

Project Aegypten (`http://www.gnupg.org/aegypten/`) has extended GnuPG and other related projects so that GnuPG can use keys stored on smartcards. This can be extended to enable GnuPG to use keys stored on the TPM.

#### 6.3.5 OpenSSH

Similarly, as the mod_ssl use case mentioned above, the TPM can be used to provide secure storage for SSH keys. In addition to the server key being protected, individuals can use their own TPM key to protect their SSH keys.

## 7 Pros and Cons of Trusted Computing

So much emotionally charged material has been written about Trusted Computing that it is difficult to separate the wheat from the chaff.

The seminal anti-TCG commentary is available from [Anderson], [RMS], [Schoen:2003], and [Moglen] with the seminal pro-TCG commentary available from [Safford].

Seth Schoen has written an excellent paper "EFF Comments on TCG Design Implementation and Usage Principles 0.95" with thoughtful, informed criticism of Trusted Computing. This paper makes the point "Many [criticisms] depend on what platform or operating system vendors do." [Schoen:2004]

Catherine Flick has written a comprehensive survey of the criticisms of Trusted Computing in her honor's thesis entitled "The Controversy over Trusted Computing." [Flick:2004]

This paper will address and attempt to clarify only the few technical issues that seem to come up repeatedly.

### 7.1 Privacy

There were many valid privacy concerns surrounding the 1.1 version of the TPM specification requiring 'trusted third parties' (PKI vendors) to issue AIKs. The concern was that the

trusted third party is able to link all pseudonymous AIKs back to a single Platform Credential. To address this concern, v. 1.2 now provides a new way for requesting AIKs called Direct Anonymous Attestation. DAA is beyond the scope of this paper, more information can be found in [Brickell]. In the v.1.1 timeframe, privacy concerns are mitigated by the fact that no manufacturer records a hash of the EK before shipping the TPM.

The measurement log, as maintained by the IMA kernel measurement architecture contains an entry about every executable that has run since boot. Like systems management data, this measurment log data may be considered sensitive data that should not be shared beyond the confines of the system, or perhaps the local network. A couple of solutions have been proposed for this problem. One very interesting solution calls for a compact verifier which verifies the targeted system and reports the results back to the challenger without leaking data. The verifier is a stock small entity with no private attributes. In this solution, the verifier would ideally be a small neighboring partition or part of the hypervisor[Garfinkel:2003]. Another solution calls for attestation based on abstract properties rather than complete knowledge of the system attributes. See [SadStu:2004] and [**?**].

### 7.2 TPM Malfunction

What happens to my encrypted data if the TPM on my motherboard dies? This depends on how the data was encrypted and what type of key was used to encrypt the data. When TPM keys are created, you have the option of making the key migratable. This implies a trade-off between security and availability so you are encouraged to consider their goal for each individual key. If the key was created migratable and the data is bound but not sealed to the TPM, you can import the key on a new TPM, restore the encrypted data from a backup, and use the key on the new TPM to access the data. If the key was not created as a migratable key or the data was sealed to the TPM, then the data will be lost. [4] Note that a backup of the migratable key must be made and stored in a safe place.

### 7.3 Secure Boot

Will trusted computing help me be able to perform secure boot as described by Arbaugh, and others[Arbaugh:1997] Arbaugh and others described “A Secure and Reliable Bootstrap Architecture” that is widely believed to be an inspiration for Trusted Computing. This paper describes the AEGIS architecture for establishing a chain of trust, driving the trust to lower levels of the system, and, based on those elements, secure boot. Trusted Computing supplies some elements of this architecture, but the TPM cannot completely replace the PROM board described in the paper. Commercially available TPMs currently do not have enough storage to contain the secure recovery code. Additionally, the infrastructural [5] and procedural hurdles, described in Section 5, would still have to be overcome. Trusted Computing enhanced BIOSes do not currently perform the verfication described in the paper, so the secure recovery has to be added to the BIOS implementation or enforced at a higher level than described in the paper.

[4] This is a slight simplification. If the PCR(s) selected for the seal operation on the new machine are exactly identical to the ones that the data was sealed to, then you can migrate the sealed data. Depending on the PCR chosen, to have the PCRs be the same the system, kernel, boot loader, and patch level of the two systems would have to be identical.

[5] The assumption of “the existence of a cryptographic certificate authority infrastructure” and the assumption that “some trusted source exists for recovery purposes.”

### 7.4 Kernel Lock-out

Does trusted computing lock me out of being able to boot my custom kernel? No, this functionality does not exist. The technical, infrastructural, and procedural hurdles, described in Section 5, would have to be overcome to enforce this technology on a global basis. Will this technology ever exist? There are cultural and political forces barring adoption of technology that takes aways the individual's right to run their operating system of choice on their general purpose computer. There is economic disincentive to forcibly limiting general purpose computing. The realization of this scenario depends more on political factors than technical capabilities.

### 7.5 Free and Open Source BIOS

Will I still be able to replace my computer's BIOS with a free BIOS? Trusted Computing does not prevent you from replacing your system BIOS with one of the free BIOS replacements, however, doing so currently violates the chain of trust. The TPM on the system can be used as a smartcard, but attestation would be broken. Free BIOS replacements can implement the relevant measurement architecture and maintain the chain of trust, if the boot block remains immutable and measures the new BIOS before it takes control of the system. The challenger during attestation would see that a different BIOS is loaded and can choose to trust the system, or not, based on their level of trust in the free BIOS.

### 7.6 Specific Additional Concerns

Is the Trusted Computing Group taking comments about specific concerns? The Trusted Computing Group has interacted with many people and organizations who have expressed concern with the group's specifications. Several of the concerns have resulted in changes to the TPM specification, for example, the introduction of Direct Anonymous Attestation, which solves many of the privacy problems that the BSI and individuals expressed. The Trusted Computing Group invites serious comments to be sent to admin@trustedcomputinggroup.org, or entered into their comment web page at `https://www.trustedcomputinggroup.org/about/contact_us/`.

## 8 Conclusion

This paper has quickly covered a great deal of ground from Trusted Computing definitions and components to uses and common concerns; no discussion about Trusted Computing and Linux is complete without citing Linus Torvald's famous email 'Flame Linus to a crisp!' proclaiming 'DRM is perfectly ok with Linux.' [6] Even though Linus considers DRM okay, the hope is that this paper makes clear that the uses of Trusted Computing are not limited to DRM, and that individual Linux users can use the TPM to improve their security. The Trusted Computing Group has shown itself willing to work with serious critiques and the Linux community is capabable of defending itself from abusive technologies being adopted. With estimates that more than 20 million computers have been sold containing a TPM, and the existence of open source drivers and libraries, let's put this technology to productive use in ways that are compatible with free and open source philosophies. While the infrastructure and software for complete support are future work items, that does not prevent users from

[6] `http://marc.theaimsgroup.com/?l=linux-kernel&m=105115686114064&w=2`

utilizing their TPM to gain secure storage for their personal keys and data through projects already available or proposed by this and other papers.

## 9 Legal Statement



## References

[Anderson] Ross Anderson *'Trusted Computing' Frequently Asked Questions*, 2003, `http://www.cl.cam.ac.uk/users/rja14/tcpa-faq.html`

[Anonymous] Anonymous *Interesting Uses of Trusted Computing*, 2004, `http://invisiblog.com/1c801df4aee49232/article/0df117d5d9b32aea8bc23194ecc270ec`

[Arbaugh:1997] William A. Arbaugh, David J. Farber, and Jonathan M. Smith *A Secure and Reliable Bootstrap Architecture*, Proceedings of the IEEE Symposium on Security and Privacy, May 1997

[Brickell] E. Brickell, J. Camenisch, and L. Chen *Direct Anonymous Attestation*, Proceedings of 11th ACM Conference on Computer and Communications Security, 2004

[Fisher] Dennis Fisher, *Trusted Computing Group Forms*, eWeek, April 8, 2003, `http://www.eweek.com/article2/0,1759,1657467,00.asp`

[Flick:2004] Catherine Flick *The Controversy over Trusted Computing*, The University of Sydney, 2004, `http://luddite.cst.usyd.edu.au/~liedra/misc/Controversy_Over_Trusted_Computing.pdf`

[Garfinkel:2003] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh *Terra: A Virtual Machine-Based Platform for Trusted Computing*, Proceedings of the 19th Symposium on Operating System Principles(SOSP 2003), October 2003

[Halcrow:2005] Michael Austin Halcrow *eCryptfs: An Enterprise-class Cryptographic Filesystem for Linux*, Ottawa Linux Symposium, 2005

[Haldar:2004] Vivek Haldar, Deepak Chandra, and Michael Franz *Semantic*

*Remote attestation: A Virtual Machine Directed Approach to Trusted Computing*, USENIX Virtual Machine Research and Technology Symposium, 2004, http://gandalf.ics.uci.edu/~haldar/pubs/trustedvm-tr.pdf

[MacDonald] Rich MacDonald, Sean Smith, John Marchesini, and Omen Wild *Bear: An Open-Source Virtual Secure Coprocessor based on TCPA*, Dartmouth College, August 2003, http://www.cs.dartmouth.edu/~sws/papers/msmw03.pdf

[Moglen] Eben Moglen *Free Software Matters: Untrustworthy Computing*, 2002, http://emoglen.law.columbia.edu/publications/lu-22.html

[Mohamed] Arif Mohamed, *Who Can You Trust?*, ComputerWeekly.com, April 26, 2005, http://www.computerweekly.com/articles/article.asp?liArticleID=138102&liArticleTypeID=20&liCategoryID=2&liChannelID=22&liFlavourID=1&sSearch=&nPage=1

[RMS] Richard Stallman *Can you trust your computer?*, 2002, http://www.gnu.org/philosophy/can-you-trust.html

[PKCS11] RSA Laboratories PKCS #11 v2.20: Cryptographic Token Interface Standard, 28 June 2004, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf

[SadStu:2004] Ahmad-Reza Sadeghi and Christian Stueble *Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms*, 20th Annual Computer Security Applications Conference, December 2004, http://www.prosec.rub.de/Publications/SadStu2004.pdf

[Safford] David Safford *TCPA Misinformation Rebuttal*, IBM T.J. Watson Research Center, 2002, http://www.research.ibm.com/gsal/tcpa/tcpa_rebuttal.pdf

[SailerIMA] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn, *Design and Implementation of a TCG-based Integrity Measurement Architecture*, 13th Usenix Security Symposium, August 2004

[Sailer] Reiner Sailer, Leendert van Doorn, and James P. Ward *The Role of TPM in Enterprise Security*, September 2004, https://www.trustedcomputinggroup.org/press/news_articles/rc23363.pdf

[Schoen:2003] Seth Schoen *Trusted Computing Promise and Risk*, EFF, 2003, http://www.eff.org/Infrastructure/trusted_computing/20031001_tc.pdf

[Schoen:2004] Seth Schoen *EFF Comments on TCG Design, Implementation and Usage Principles 0.95*, EFF, 2004, http://www.eff.org/Infrastructure/trusted_computing/20041004_eff_comments_tcg_principles.pdf

[Smith:2005] Sean W. Smith *Trusted Computing Platforms Design and*

*Applicatons*, Springer, 2005, ISBN 0-387-23916-2

[Strasser] Mario Strasser *A Software-based TPM Emulator for Linux*, Swiss Federal Institute of Technology, 2004, http://www.infsec.ethz.ch/people/psevinc/TPMEmulatorReport.pdf

[TCGArch] TCG *Trusted Computing Group Specification Architectural Overview, Revision 1.2*, April 28, 2004, http://www.trustedcomputinggroup.org/downloads/TCG_1_0_Architecture_Overview.pdf

[TCGBackground] TCG *Trusted Computing Group Backgrounder*, January 2005, http://www.trustedcomputinggroup.org/downloads/background_docs/TCGBackgrounder_revised_012605.pdf

[TCGFAQ] TCG *Trusted Computing Group Fact Sheet*, 2005, http://www.trustedcomputinggroup.org/downloads/background_docs/FACTSHEET_revised_020105.pdf

[TCGMain] TCG *TCG Main Specification Version 1.1b*, 2003, http://www.trustedcomputinggroup.org/downloads/specifications/TCPA_Main_Architecture_v1_1b.zip

[TCGPC] TCG *TCG PC Specific Implementation Specification Version 1.1*, August 18, 2003, http://www.trustedcomputinggroup.org/downloads/TCG_PCSpecification_v1_1.pdf

[TCGRSA] TCG *Putting Trust into Computing: Where does it Fit? - RSA Conference 2005*, https://www.trustedcomputinggroup.org/downloads/Putting_Trust_Into_Computing_Where_Does_It_Fit_021405.pdf

[TNC] TCG *TCG TNC Architecture Version 1.0*, 2005, http://www.trustedcomputinggroup.org/downloads/specifications/TNC_Architecture_v1_0_r4.pdf

[TPM] TCG *TCG TPM Specification Version 1.2*, 2004, http://www.trustedcomputinggroup.org/downloads/specifications/mainP1DP_rev85.zip

[TSS] TCG *TCG Software Stack Specification Version 1.1*, 2003, http://www.trustedcomputinggroup.org/downloads/TSS_Version__1.1.pdf

# Proceedings of the Linux Symposium

Volume Two

July 20nd–23th, 2005
Ottawa, Ontario
Canada

## Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*
Stephanie Donovan, *Linux Symposium*

## Review Committee

Gerrit Huizenga, *IBM*
Matthew Wilcox, *HP*
Dirk Hohndel, *Intel*
Val Henson, *Sun Microsystems*
Jamal Hadi Salimi, *Znyx*
Matt Domsch, *Dell*
Andrew Hutton, *Steamballoon, Inc.*

## Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*