Mutual Remote Attestation: Enabling System Cloning for TPM based Platforms

Ulrich Greveler, Benjamin Justus, and Dennis Loehr

Computer Security Lab
Münster University of Applied Sciences
D-48565 Steinfurt, Germany
{greveler|benjamin.justus|loehr}@fh-muenster.de

Abstract. We describe a concept of mutual remote attestation for two identically configured trusted (TPM based) systems. We provide a cryptographic protocol to achieve the goal of deriving a common session key for two systems that have verified each other to be a clone of themselves. The mutual attestation can be applied to backup procedures without providing data access to administrators, i. e. one trusted systems exports its database to another identical trusted system via a secure channel after mutual attestation is completed.
Another application is dynamically parallelizing trusted systems in order to increase the performance of a trusted server platform.
We present details of our proposed architecture and show results from extensive hardware tests. These tests show that there are some unresolved issues with TPM-BIOS settings currently distributed by PC hardware manufacturers since the specification regarding measurement of extended platform BIOS configuration is either not met or the usage of undocumented options is required.

Keywords. Mutual Attestation, Trusted Computing, Data Cloning, Key Exchange Protocol

1 Introduction

Recent developments related to the legal and social aspects of privacy issues call for technical measures enforcing strict restrictions and requirements on the collection, use and disclosure of personal data. Trusted systems can be used for secure storage of sensitive data.

Once a system state is defined as a trusted state and the system is set up to this state, its security characteristics can be transferred to a system clone that is composed of identical software (boot chain components, operating system, and applications) and matching hardware. A system clone can be generated via methods such as copying the contents of one system hard disk image to another disk, or automatic installation using an install-script. The execution of these procedures will result into a run-capable system clone (depending on operating system characteristics). Mutual attestation is the key functionality to *verify* the secure cloning of trusted platforms.

Applications using cloned trusted platforms include

- Database synchronization: database management systems that offer a restricted access to its databases. Further, the database can be synchronized and backed up without the need of low-level table access for administrators.
- Parallel computing: clustering synchronous trusted servers increase the output performance and reduce the response time compared to stand-alone servers.
- Enforcing restrictions expressed through rights expression language (RELs) across systems: a REL description might require the system to restrict access and maintain a state (i. e. a maximum of n queries are permitted on a database in order to avoid illegitimate database duplication). This state needs to be distributed across physical systems in a way that one logical system stays consistent (i. e. set up 2 physical systems that allow $n/2$ requests each until the next synchronization takes place).

The implementation of any of the above projects requires an efficient and reliable remote attestation scheme. The ideal architecture should possess the following components: **1** systematic integrity measurement and automated integrity measurement verification procedures; **2** a secure key exchange protocol that allows both systems to possess a common session key. Existing Trusted Computing Standards provide extensive architectures and opensource components for the purpose of carrying out trust based applications. To our dismay, we have not found any off-the-shelf protocol or software which may provide a ready implementation of our trust-cloning project. A search of web shows that there are (section 2.2) some available opensource software tools that are designed to carry out functionalities of attestation for TPM based systems. These tools are mostly experimental, and do not take into account the implementation architecture of mutual attestation as an whole. A tailor-designed and ready-to-implement mutual attestation architecture is desired.

Contributions: Our contributions in this paper are as follows

1. We propose a mutual remote attestation protocol for two identically configured TPM hardware and provide implementation details.
2. We describe in details the hardware test results and discuss to what extent *system cloning* is possible for TPM based hardware.

Our mutual remote attestation scheme has particular merit in a corporate setting where database synchronization and backup are constantly required between available servers. The protocol is easy to implement and requires no intervention of a third party, such as a trusted certification authority once the attestation procedure has started. Our mutual attestation scheme can be viewed as a small step in the development of much needed peer-to-peer (P2P) attestation techniques.

The plan of the paper is as follows. Section 2 presents the related work on remote attestation, its applications and the available tools to perform the attestation. Our *Mutual Attestation Scheme* is explained in section 3. Section 4

provides a discussion and conclusion on the scheme. The appendices A and B give a detailed design of our proposed protocol and some test values respectively.

2 Background and Related Work

2.1 Remote Attestation

Specifying the notion of trust in computing platforms has been a goal of computer science research for decades. The use of secure operating system environments were proposed in the 1970s [23]. The premise of a *secure system* is built upon the philosophy that any system is only as secure as the foundation upon which it is built.

The Trusted Computing Group (TCG) is an industry standards body formed to develop and promote specifications for trusted computing and security technologies. The TCG proposed a trust model where each device is equipped with a hardware root-of-trust associated with the platform that can measure integrity metrics and may confirm these metrics to other parties. Regarding PCs this hardware is a chip called the Trusted Platform Module (TPM) and the process of reporting the integrity of a platform is known as *remote attestation*. When the TPM reports the values of the integrity metrics that it has stored, the TPM signs those values using a TPM identity.

To achieve the goals of *remote attestation*, TCG has introduced in version 1.1 specifications the concept of privacy certification authority (Privacy CA) [14]. It works briefly as follows. Each TPM is equipped with a RSA key pair called an Endorsement Key (EK). The Privacy CA is assumed to know the Endorsement Keys of all valid TPMs. Now, when TPM needs to authenticate itself to a verifier, it generates a second pair of RSA key called an Attestation Identity Key (AIK), it sends the AIK public key to the Privacy CA, and authenticates this public key w.r.t the EK. The Privacy CA will check whether it finds the EK in its list and, if so, issues a certificate to the TPM's AIK key. The TPM can then forward this certificate to the verifier and authenticate itself w.r.t. this AIK.

As discussed by Brickell, Camenisch and Chen [4], version 1.2 of the TCG specifications incorporate the Direct Anonymous Attestation (DAA) protocol. This protocol is designed to address anonymity issues of remote attestation. The DAA scheme is rather sophisticated [4, 5] whose implementation requires novel techniques and methods [7, 9, 10]. Many applications related to privacy preserving and privacy enhancing are built upon the concept of DAA [2, 6, 18, 20].

There have been several proposal in the literature to combine DAA with key exchange protocols. Balfe et al. [3] proposed anonymous authentication protocol in peer-to-peer networks by embedding DAA with TLS and IPSec. Cesena et al. [8] proposed an anonymous authentication protocol based on TLS and DAA including a reference implementation. Recently, Li and Walker [26] incorporated DAA scheme into a key exchange protocol. Further, they introduced a security model for key exchange with anonymous authentication, and provided rigorous security proof under the proposed model.

2.2 Available Remote Attestation Opensource Tools

The implementation of our trusted system cloning applications as described in the introduction requires the initiation of a mutual attestation protocol. Even though anonymity of the systems is not required during the attestation, we do require that the session keys to be authenticated with respect to some certification authorities. The successful authentication of critical keys assures a system that a remote platform who is trying to access the database is truly operating on trusted hardware modules.

As stated in the previous section, the certification of AIK keys ideally calls for the interaction with a Privacy CA. Theoretically, a Privacy CA should hold a list of all valid EK certificates delivered from TPM manufacturers. However, this kind of trust chain infrastructure at present is still lacking. To the authors' knowledge, currently only Infineon [27] is providing TPMs with Endorsement certificates.

Despite the lack of certification authorities, there exists several opensource tools that allow users to carry out experimentally the steps of mutual attestation. The *TPM Quote Tools* [25] contain a collection of programs that provides functionalities such as AIK key generation, TPM quote operations, and TPM quote verification operations.

Another ongoing project is *Trusted Computing for the Java Platform* [13]. The project is developed and maintained at the Institute for Applied Information Processing and Communication, Graz University of Technology. The package at present stage includes a basic implementation of a Privacy CA Server.

In the next section, we shall describe a mutual attestation protocol for two identically configured TPM based systems. During the protocol, AIK keys are generated, and we do make the assumption that all genuinely generated AIK keys are certifiable by some means. This assumption should be reasonable as Trusted Computing technologies and its related infrastructure are ever growing at present.

3 Mutual Attestation Scheme

3.1 High-Level Description

Our scheme allows mutual remote attestation between two identically configured TPM based systems, and provides a common session key for both systems at the end of the protocol. The scheme is an integrity based attestation. The reader can find the details of the protocol in appendix A. Figure 1 gives a pictorial representation of the protocol. We shall in this section explain the underlying ideas and discuss some of the implementation issues.

Though anonymity is not required during the attestation, we do require on-site systematic integrity measurement and automated measurement verification procedure. In fact, our mutual attestation scheme is very much driven by the the Cloning-Applications at hand, whereas schemes proposed in [26, 17] are much more theoretical and are not implementable at present. The security proof of our

protocol can be derived along the line as described in [26]. The protocol is of the challenge-and-response type. Both TPM based systems during attestation issue a sequence of challenges. The systems then mutually attest towards each other by demonstrating that they satisfy the specified attestation criterion. The attestation criteria are: system integrity (PCR values) and integrity of the AIK keys.

Initially, *system I* generates an AIK key AIK_I and obtains a certificate $Cert_I$ from a certification authority (e.g. Privacy CA). The purpose of the certificate is to verify the integrity of AIK_I. Similarly, *system II* generates an AIK key AIK_{II} and obtains a certificate $Cert_{II}$. Both systems now exchange and then verify each others' certificates.

Assuming the certificates are valid, both systems start a Diffie-Hellman key exchange protocol. This is achieved as follows. From a list of agreed-upon primes, *system I* selects a prime with required security parameter and a primitive root $g \bmod p$. Next, *system I* selects an secret integer a and computes its public Diffie-Hellman parameter ($A = g^a$). Similarly, *system II* generates its public Diffie-Hellman parameter ($B = g^b$) where b is *system II*'s secret parameter.

Next, *system I* and *system II* exchange their public Diffie-Hellman parameters. This allows each system to compute the shared Diffie-Hellman key. For example, *system I* computes its key as $\mathbf{skey_I} = B^a$ where B is *system II*'s public DH parameter. And *system II* computes its key as $\mathbf{skey_{II}} = A^b$ where A is *system I*'s public DH parameter. Notice, we have not assumed $\mathbf{skey_I} = \mathbf{skey_{II}}$ at this point of the protocol. The keys $\mathbf{skey_I}$ and $\mathbf{skey_{II}}$ are to be compared at the next step of integrity check. This assumption is needed to prevent man-in-the-middle type of attacks.

System integrity check is the next step. To prove system integrity, *system I* uses the TPM Quote utilities to sign a set of PCR values using its AIK key. If the AIK key is genuine and controlled by the TPM, it will only sign true and correct PCR values, which may therefore be taken to accurately represent the state of the signing system. Also to keep the quote fresh, the quote also includes a hashed $\mathbf{skey_I}$. *System I* sends $sign(\mathbf{PCR} \,||\, \mathbf{hash}(\mathbf{skey_I}))$ to *system II*. After verifying the signature, *system II* checks the integrity of *system I* by comparing *system I*'s PCR values with its own. *System II* also checks that $\mathbf{skey_I}$ is correctly formed. This is achieved by comparing $\mathbf{hash}(\mathbf{skey_I})$ with his own key $\mathbf{hash}(\mathbf{skey_{II}})$. The hashed exchange of **skey** is pivotal here as eavesdroppers will not have access to **skey** (one-wayness of the cryptographic hash function) and the fresh key is linked to the trusted system state.

This completes the steps of mutual attestation. And since $\mathbf{hash}(\mathbf{skey_I}) = \mathbf{hash}(\mathbf{skey_{II}})$, both systems at this point possess the common session key $\mathbf{skey_I} = \mathbf{skey_{II}} =: \mathbf{skey}$.

3.2 Diffie-Hellman Key Exchange

The key **skey** is computed by both systems following the Diffie-Hellman key exchange protocol [11]. Since the finite field Diffie-Hellman algorithms has roughly the same key strength as RSA for the same key size, we have chosen the DH

parameter to be of the size 2048 bits. The key length is reckoned sufficient until the end of 2016 [12].

We fix a group generator g, and find a safe 2048-bit prime whose group generator is g. This can be easily implemented using for example the open source software OpenSSL. The private key for *system I* (w.r.p. *system II*) is then a randomly generated integer a in the interval $[2, p-1]$. *System I*'s public Diffie-Hellman parameter is then computed as

$$A = g^a \bmod p.$$

And *system II*'s public DH parameter is computed as

$$B = g^b \bmod p$$

where b is *system II*'s private key.

In practice, a list of such safe primes is pre-generated and stored on both systems. At the beginning of each session of mutual attestation, a agreed-upon prime p will be selected from the list.

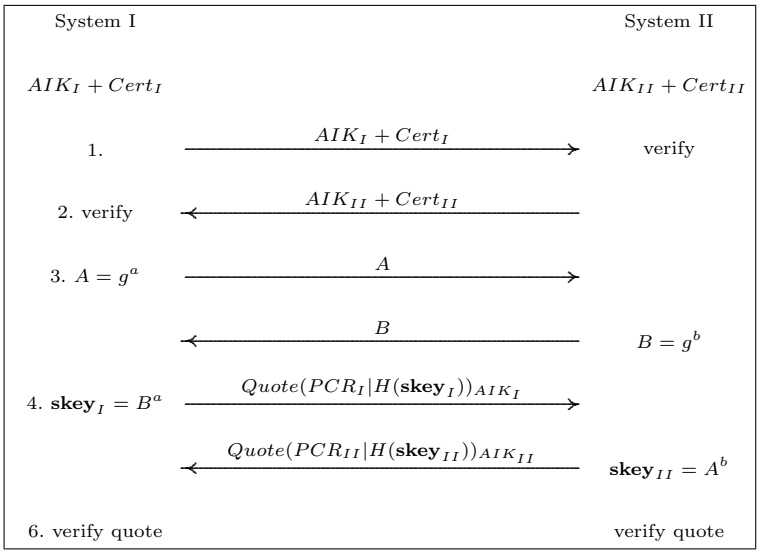

Fig. 1. Mutual Attestation Scheme for Identically Configured TPM Systems

3.3 TPM Quote and Verify

The most important part of the attestation is the system integrity check. TPM has a set of special volatile registers called *platform configuration registers* (PCRs).

These 160-bit long registers are used to keeping track of the integrity information during a bootstrap process. The TCPA specification defines a set of functions for reporting PCR values [14, 15]. The TPM Quote operation is able to sign a specified set of PCRs. The input of the Quote function also includes a 160 bit challenge file. By including this value in the Quote signature, the verifier knows that the Quote is fresh and is not an old replay of an old quote. In our protocol, the challenge file is a SHA-1 hash on **skey**. Each system can verify the integrity of the other system by comparing received PCR quote with its own PCR values.

3.4 Hardware Test Results

System name	Machine Model	BIOS Version	TPM Manufacturer & Chip Version
Lenovo T510 System 1	4384-GEG	1.35 (6MET75WW)	STM 1.2.8.16
Lenovo T510 System 2	4384-GEG	1.35 (6MET75WW)	STM 1.2.8.16
Lenovo T60	1951-WWA	2.20 (79ETE0WW)	ATML 1.2.11.5
Lenovo T61	8889-ABG	2.26 (7LETC6WW)	ATML 1.2.13.9
Lenovo M58p	9965-A5G	(5CKT61AUS)	WEC 1.2.2.16

Table 1. Test Environment

No	System(s)	Description	Result
1	T510 System 1	reboot system	same PCR 0–15 values
2	T510 System 2	**1.** Boot order changed, **2.** dynamic selection of different bootmedia	PCRs 1,4 are changed for **1.** as well as **2.**
3	T510 System 1	booting two different OSs	PCRs 4,5 differ as well as OS specific PCRs 8–15
4	T510 System 1	without extended reporting switch on/off Ultrabay switch on/off Firewire	same PCR 0–15 values in all 4 subcases
5	T510 System 1	activated *CMOS Reporting* switching Ultrabay on/off	PCR 1 differs → on/off is detected
6	T510 System 1	BIOS default settings plus SMBIOS extended reporting switch on/off Ultrabay	same PCR 0–15 values → on/off is undetected
7	T510 System 1	BIOS default settings plus NVRAM extended reporting switch on/off Ultrabay	PCR 1 differs → on/off is detected
8	T510 System 1	BIOS default settings plus ESCD extended reporting switch on/off Ultrabay	PCR 1 differs → on/off is detected

Table 2. Hardware Test cases for a Single System

No	System(s)	Description	Result
1	T510 System 1, T61,M58p	measure different hardware configurations with same boot chain without extended reporting[a]	differences in PCRs 0,1,2,4,6 (please note next test case!)
2	T510 System 1, T60	measure different hardware configurations with same boot chain without extended reporting	PCR 1 is equal on both systems
3	T510 System 1, T510 System 2	BIOS default settings plus maximum extended reporting (BIOS ROM String and ESCD[b] and CMOS and NVRAM and SMBIOS)	PCR 1 differs between (identical hardware) systems
4	T510 System 1, T510 System 2	BIOS default settings plus CMOS extended reporting	PCR 1 differs between (identical hardware) systems
5	T510 System 1, T510 System 2	BIOS default settings plus NVRAM extended reporting	PCR 1 differs between (identical hardware) systems
6	T510 System 1, T510 System 2	BIOS default settings plus ESCD extended reporting	PCR 1 is equal on both (identical hardware) systems

[a] *without extended reporting* does refer to the BIOS menu *Security Reporting Options* settings: BIOS ROM String and ESCD and CMOS and NVRAM and SMBIOS are in the state *Disabled*.

[b] ESCD (Extended System Configuration Data) is a subset of the nonvolatile BIOS memory (still named CMOS in the BIOS settings.)

Table 3. Hardware Test cases Multiple Systems

The root of trust in the mutual attestation protocol lies at the fact that two identically configured TPM based hardware have the same boot-up values in certain platform configuration registers. This is a claim laid out in the relevant TCG specifications [16] which we have rigorously tested in the lab. Together with the BIOS CRTM, the TPM forms a root of Trust: the TPM allows a secure storage and the reporting of relevant security metrics into PCRs. These metrics can be used to detect changes to previous configurations from which it can easily be deduced whether a system clone is comparable in its security metrics or not. In our tests, we compared extensively the boot-up PCR values among different TPM hardware.

Despite the fact that there exists a *TCG Generic Server Specification* we could not supply our test bed environment with ready-to-use server hardware since TPM-based servers are still a shortage with respect to the IT hardware market. Thus, for our test results we limited ourselves to the testing of TPM equipped notebook and desktop hardware. Table 1 shows the hardware test environment. The tested hardware include : IBM Lenovo T510, IBM Lenovo T61, IBM Lenovo T60, and IBM Lenovo M58p (Desktop computer). All the platforms have a TPM 1.2 chip on main-board[1].

There are similar hardware test results in the literature. Sadeghi et al. [1, 22] tested a core set of TPM functionalities on TPM chips from different vendors.

[1] TPM hardware details: Atmel TPM 97SC3203 (on T60, T61), Chipset integrated TPM (on T510), 9965-A5G TPM 1.2 Winbond (on the M58p Desktop)

The compliance test results show that there exist discrepancies in the behaviors among different TPM chips. Several TPMs show non-compliant behavior with respect to the TCG specification and errors occur sometimes in the runtime library [24].

For single system testings (Table 2), we fix a TPM platform and record the boot-up PCR values for the various system settings. For instance, we have booted up two different Linux OS systems (No. 3 in Table 2). The boot-up PCR values are recorded in Figure 3.4. We have found resulted differences in PCRs 4, 5 and 8 - 15. Changing boot order or dynamically selecting a different bootmedia (No. 2 in Table 2) results differences in PCRs 1 and 4 (Figure 2.3). We have also tested the effects of on/off DMA activation on PCRs. Our test results show that the vendor default configuration does not include Extended Security Reporting Options (in BIOS submenu) in the PCRs measurement. And the activation of BIOS DMA features (No. 4 in Table 2) results in no differences in PCRs. The activation of BIOS DMA features is detected only after we switch on the CMOS, NVRAM and ESCD reporting in the Extended Security Reporting Options.

The significance of the DMA tests is the following: DMA allows devices to transfer data without supervision by the CPU. An attacker with physical access to the trusted system may activate DMA options in the BIOS and subsequently connect a hardware to the system to access its memory [21, 19]. To prevent such a security flaw, DMA can be disabled in the BIOS settings and any change of this setting should be reflected in the configuration register values. A system with a changed DMA setting will then not be able to qualify as a clone of a trusted system .

Among different hardware platforms (Table 3), there exists expected PCRs discrepancies (No. 1 in Table 3). We also extensively tested among the Extended Security Report Options (No. 3 - 6 in Table 3) between two similar hardware. While the activation of most of the Extended Reporting features resulted differences in PCRs 1, the only exception is being the BIOS ESCD extended reporting feature whose activation has produced the same PCRs 1 on both platforms. This system behavior is not documented in the system documentation. Our testing on the T510s show that the ESCD reporting option is the only feature fulfilling the double requirements:

1. same PCR 1 value return after identical system hardware (here T510 vs. T510) measurement
2. a change of DMA related BIOS options (stored in the non-volatile BIOS memory) is detected and results in a changed PCR 1 value (see test cases 8 in Table 2 and 6 in Table 3)

The other available reporting option (CMOS, NVRAM, SMBIOS) do not meet the specified TCG requirement [16]: *platform configuration information being either unique (e.g. serial numbers) or automatically updated (e.g. clock registers) must not be measured into PCR 1.* The activation of any one of the three extended security reporting options above on two identical systems results in different PCRs 1 (see test cases 3–5 in Table 3).

4 Discussion and Conclusion

Our contribution in this paper is the proposal of a mutual attestation protocol for identical TPM based platforms. We also provide source code and bootable prototypes on our project website[2].

Trusted Platform Modules are deployed in many PC clients (especially laptop computers) since 2006 and they can be therefore viewed as commodity goods. However, software applications using the TPM attestation functions are still rare and to our knowledge limited to project prototypes.

While the ability of attesting a remote platform is supposed to be one of the main functionalities of the Trusted Platform Module, TPM based remote attestation is still no ready-to-use technology. Real-world attestation applications require not only that the system architecture to have a ready-to-implement TCG Software Stack, but it must also have compatible hardware to support the relevant TPM operations.

The hardware issues we have identified in section 3.4 require us to use two identical hardware for the purpose of cloning TPM based systems, taking into account the fact that the BIOS machine code needs to be part of the trusted boot chain. Though hardware equivalence is rather a strong requirement for the cloning procedures, it is still insufficient in the following sense: We were unable to add security relevant BIOS settings to the verifiable state of the system in an appropriate way. The activation of the extended reporting options results into different PCR values for identical systems. Only by rigorously testing the undocumented options in the BIOS setup submenu, we were able to derive a BIOS configuration from which our mutual attestation scheme can be carried out: i.e. the cloned system has the same PCR values and a change of security relevant BIOS variables (e.g. DMA activation) is detected.

Our results show that the specified requirement [16] that "platform configuration information being unique or automatically updated must not be measured" is apparently violated. The full activation of extended security reporting options results in different values on identical systems.

Note that the situation for *TPM-Sealing* is quite different from attestation since there are ready-to-use software libraries and only one TPM platform is involved per sealing or de-sealing procedure. An application architecture making use of this TPM-based function would run on any compatible hardware since sealed files are not to be migrated to different platforms in any case.

The purpose of Trusted Computing is to enable each endpoint to make a trusted decision about the other endpoint, regardless of hardware background and software configurations. Indeed in reality, it is hard to expect a homogeneous enterprise with identical hardware, and completely synchronized BIOS settings, and globally verified Service Packs installed. Future research in trusted computing should focus on more robust and flexible mechanism for trust establishment and infrastructure. In the meantime, we will require from the system vendors a well documented TPM platform together with a full disclosure of BIOS internal

[2] http://www.daprim.de/

integrity checks regarding the extended security reporting options. The present situation that the platform owner is required to test undocumented options, and to find out which of these options being in line with the TCG specifications is not acceptable.

Acknowledgements

Many thanks go to the anonymous reviewers who provided detailed and insightful commentary on this paper.

References

1. Sirrix AG. *TPM Compliance Test Results.* Available at: http://www.sirrix.com/content/pages/test_results_en.htm, 2006.
2. Frederik Armknecht, Liqun Chen, Ahmad-Reza Sadeghi, and Christian Wachsmann. Anonymous Authentication for RFID Systems. In *RFIDSec*, pages 158–175, 2010.
3. Shane Balfe, Amit D. Lakhani, and Kenneth G. Paterson. Trusted Computing: Providing Security for Peer-to-Peer Networks. In *Peer-to-Peer Computing*, pages 117–124, 2005.
4. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct Anonymous Attestation. In *ACM Conference on Computer and Communications Security*, pages 132–145, 2004.
5. Ernie Brickell, Liqun Chen, and Jiangtao Li. A New Direct Anonymous Attestation Scheme from Bilinear Maps. In *TRUST*, pages 166–178, 2008.
6. Ernie Brickell and Jiangtao Li. Enhanced privacy id: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In *WPES*, pages 21–30, 2007.
7. Ernie Brickell and Jiangtao Li. A Pairing-Based DAA Scheme Further Reducing TPM Resources. In *TRUST*, pages 181–195, 2010.
8. Emanuele Cesena, Hans Löhr, Gianluca Ramunno, Ahmad-Reza Sadeghi, and Davide Vernizzi. Anonymous Authentication with TLS and DAA. In *TRUST*, pages 47–62, 2010.
9. Liqun Chen. A DAA Scheme Using Batch Proof and Verification. In *TRUST*, pages 166–180, 2010.
10. Liqun Chen, Dan Page, and Nigel P. Smart. On the Design and Implementation of an Efficient DAA Scheme. In *CARDIS*, pages 223–237, 2010.
11. W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory, Vol IT-22*, pages 644–654, 1976.
12. Federal Office for Information Security. Algorithms for qualified electronic signatures. Technical Report 19, Feb 2010.
13. Trusted Computing for the Java(tm) Platform. URL: http://trustedjava.sourceforge.net/.
14. Trusted Computing Group. *Trusted computing platform alliance (TCPA) main specification, version 1.1b.* Available at www.trustedcomputing.org, 2001.
15. Trusted Computing Group. *Trusted computing platform alliance (TCPA) main specification, version 1.2.* Available at: www.trustedcomputing.org, 2003.

16. Trusted Computing Group. *TCG EFI Platform Specification V1.20*. Available at: www.trustedcomputing.org, 2006.
17. Adrian Leung and Chris J. Mitchell. Ninja: Non identity based, privacy preserving authentication for ubiquitous environments. In *Ubicomp*, pages 73–90, 2007.
18. Jiangtao Li and Anand Rajan. An Anonymous Attestation Scheme with Optional Traceability. In *TRUST*, pages 196–210, 2010.
19. J. Marchesini, S. Smith, O. Wild, and R. MacDonald. *Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear*. TR2003-476, Dartmouth College, 2003.
20. Mohammad Nauman, Sohail Khan, Xinwen Zhang, and Jean-Pierre Seifert. Beyond Kernel-Level Integrity Measurement: Enabling Remote Attestation for the Android Platform. In *TRUST*, pages 1–15, 2010.
21. D.R. Piegdon and L. Pimenidis. hacking in physically addressable memory. In *Proc. 4th International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA'07)*.
22. A.R. Sadeghi, M. Selhorst, C. Stueble, C. Wachsmann, and M. Winandy. TCG Inside? A Note on TPM Specification Compliance. In *Proceedings of the first ACM workshop on Scalable trusted computing*, pages 47–56. ACM, 2006.
23. M Schroeder. Engineering a security kernel for multics. *In Fifth Symposium on Operating Systems Principles. November 1975*, pages 125–132, 1975.
24. A. Shubina, S. Bratus, W. Ingersoll, and S.W. Smith. The Diversity of TPMs and its Effects on Development. In *ACM STC '10*, 2010.
25. TPM Quote Tools. Available at: http://sourceforge.net/.
26. Jesse Walker and Jiangtao Li. Key Exchange with Anonymous Authentication using DAA-SIGMA Protocol. In *IACR eprint archive*, 2010.
27. Infineon Technologies Website. URL: http://www.infineon.com/cms/en/product/channel.html?channel=ff80808112ab681d0112ab692060011a.

A Mutual Attestation Protocol

0. *System I* generates a AIK key AIK_I and obtains a certificate $Cert_I$. *System II* generates a AIK key AIK_{II} and obtains a certificate $Cert_{II}$.
1. *System I* and *system II* exchange their certificates and public AIK keys:

$$I \xrightarrow{\left(AIK_I^{pub}, Cert_I\right)} II, \quad II \xrightarrow{\left(AIK_{II}^{pub}, Cert_{II}\right)} I$$

2. *System I* verifies *system II*'s certificate. *System II* verifies *system I*'s certificate. The protocol continues upon successful verifications of both certificates.
3. Let p be the agreed-upon prime of required security parameter and a group generator g that will be used to generate the Diffie-Hellman parameters. *System I* randomly selects an integer a in the interval $[2, p-1]$. *System II* randomly selects an integer b in the interval $[2, p-1]$. *System I* computes value A and *system II* computes value B:

$$A = g^a \bmod p, \quad B = g^b \bmod p.$$

4. System I sends A to system II. System II sends B to system I:

$$I \xrightarrow{A} II, \quad II \xrightarrow{B} I$$

5. *System I* computes the key $\mathbf{skey}_I$. *System II* computes the key $\mathbf{skey}_{II}$:

$$\mathbf{skey}_I = B^a, \quad \mathbf{skey}_{II} = A^b$$

6. The systems process mutual system integrity check. The steps are:

a *System I* signs the PCR values and hashed $\mathbf{skey}_I$ and forwards it to *system II*. *System II* signs the PCR values and hashed $\mathbf{skey}_{II}$ and forwards it to *system I*.

$$I \xrightarrow{Quote(PCR_I \,||\, H(\mathbf{skey}_I)} II, \quad II \xrightarrow{Quote(PCR_{II} \,||\, H(\mathbf{skey}_{II})} I$$

b *System I* verifies the signature. *System I* compares the received PCR_{II} values with its own PCR values, then it compares the received $H(\mathbf{skey}_{II})$ with its own hashed key $H(\mathbf{skey}_I)$. If all the values agree, *system I* grants database access to system II. The session key is the common keys $H(\mathbf{skey}_{II}) = H(\mathbf{skey}_I)$.
c *System II* verifies the signature. *System II* compares the received PCR_I values with its own PCR values, then it compares the received $H(\mathbf{skey}_I)$ with its own hashed key $H(\mathbf{skey}_{II})$. If all the values agree, *system II* grants database access to *system I*. The session key is the common keys $H(\mathbf{skey}_{II}) = H(\mathbf{skey}_I)$.

B Example PCR Values

```
multiple systems - test case 2
Lenovo T510 System 1                          IBM T60
PCR-00: 42 A2 AF 18 81 7C ...                 PCR-00: A2 7B 2C EF 5B 0B ...
PCR-01: 48 DF F4 FB F3 A3 ... <===>           PCR-01: 48 DF F4 FB F3 A3 ...
PCR-02: 24 5B 5C E4 FF F1 ...                 PCR-02: 53 DE 58 4D CE F0 ...
PCR-03: 3A 3F 78 0F 11 A4 ...                 PCR-03: 3A 3F 78 0F 11 A4 ...
PCR-04: 1E F2 2E 55 6D 02 ...                 PCR-04: C0 D0 F2 DF 3D F9 ...
PCR-05: 3F C8 89 02 05 59 ...                 PCR-05: 13 E3 62 E8 6D 4B ...
PCR-06: 58 5E 57 9E 48 99 ...                 PCR-06: 58 5E 57 9E 48 99 ...
PCR-07: 3A 3F 78 0F 11 A4 ...                 PCR-07: 3A 3F 78 0F 11 A4 ...
PCR-08: 03 B3 B2 AE 7E 2B ...                 PCR-08: 03 B3 B2 AE 7E 2B ...
PCR-09: F6 E5 F7 35 B0 2F ...                 PCR-09: F6 E5 F7 35 B0 2F ...
PCR-10: 00 00 00 00 00 00 ...                 PCR-10: 00 00 00 00 00 00 ...
PCR-11: 00 00 00 00 00 00 ...                 PCR-11: 00 00 00 00 00 00 ...
PCR-12: F0 22 54 28 39 D1 ...                 PCR-12: F0 22 54 28 39 D1 ...
PCR-13: 34 42 7B 49 32 23 ...                 PCR-13: 34 42 7B 49 32 23 ...
PCR-14: A8 28 2F BD A7 BC ...                 PCR-14: A8 28 2F BD A7 BC ...
```

Fig. 2. Boot-up PCR Values of IBM T510 and IBM T60 Without Extended Reporting: different hardware configurations with the same bootchain, but without extended reporting results the same PCR 1

```
single system - test case 2
Lenovo T510 System 2                          Lenovo T510 System 2
PCR-00: 42 A2 AF 18 81 7C ...                 PCR-00: 42 A2 AF 18 81 7C ...
PCR-01: 56 6E BA FB 53 FE ... <***>           PCR-01: CB B7 0F E9 7A D0 ...
PCR-02: 24 5B 5C E4 FF F1 ...                 PCR-02: 24 5B 5C E4 FF F1 ...
PCR-03: 3A 3F 78 0F 11 A4 ...                 PCR-03: 3A 3F 78 0F 11 A4 ...
PCR-04: 78 6E AD 00 83 A0 ... <***>           PCR-04: 1E F2 2E 55 6D 02 ...
PCR-05: 3F C8 89 02 05 59 ...                 PCR-05: 3F C8 89 02 05 59 ...
PCR-06: 58 5E 57 9E 48 99 ...                 PCR-06: 58 5E 57 9E 48 99 ...
PCR-07: 3A 3F 78 0F 11 A4 ...                 PCR-07: 3A 3F 78 0F 11 A4 ...
PCR-08: 03 B3 B2 AE 7E 2B ...                 PCR-08: 03 B3 B2 AE 7E 2B ...
PCR-09: F6 E5 F7 35 B0 2F ...                 PCR-09: F6 E5 F7 35 B0 2F ...
PCR-10: 00 00 00 00 00 00 ...                 PCR-10: 00 00 00 00 00 00 ...
PCR-11: 00 00 00 00 00 00 ...                 PCR-11: 00 00 00 00 00 00 ...
PCR-12: F0 22 54 28 39 D1 ...                 PCR-12: F0 22 54 28 39 D1 ...
PCR-13: 34 42 7B 49 32 23 ...                 PCR-13: 34 42 7B 49 32 23 ...
PCR-14: A8 28 2F BD A7 BC ...                 PCR-14: A8 28 2F BD A7 BC ...
```

Fig. 3. Boot-up PCR values of IBM T510 Before and After Boot Order is Changed: changing bootorder or dynamically selecting a different bootmedia results differences in PCR 1 and 4

```
single system - test case 3
Lenovo T510 System 1                          Lenovo T510 System 1
PCR-00: 42 A2 AF 18 81 7C ...                 PCR-00: 42 A2 AF 18 81 7C ...
PCR-01: 48 DF F4 FB F3 A3 ...                 PCR-01: 48 DF F4 FB F3 A3 ...
PCR-02: 24 5B 5C E4 FF F1 ...                 PCR-02: 24 5B 5C E4 FF F1 ...
PCR-03: 3A 3F 78 0F 11 A4 ...                 PCR-03: 3A 3F 78 0F 11 A4 ...
PCR-04: 1E F2 2E 55 6D 02 ... <***>           PCR-04: A3 CE B1 EF AC 90 ...
PCR-05: 3F C8 89 02 05 59 ... <***>           PCR-05: 99 21 E8 EA 42 08 ...
PCR-06: 58 5E 57 9E 48 99 ...                 PCR-06: 58 5E 57 9E 48 99 ...
PCR-07: 3A 3F 78 0F 11 A4 ...                 PCR-07: 3A 3F 78 0F 11 A4 ...
PCR-08: 03 B3 B2 AE 7E 2B ... <***>           PCR-08: 00 00 00 00 00 00 ...
PCR-09: F6 E5 F7 35 B0 2F ... <***>           PCR-09: 00 00 00 00 00 00 ...
PCR-10: 00 00 00 00 00 00 ...                 PCR-10: 00 00 00 00 00 00 ...
PCR-11: 00 00 00 00 00 00 ...                 PCR-11: 00 00 00 00 00 00 ...
PCR-12: F0 22 54 28 39 D1 ... <***>           PCR-12: 00 00 00 00 00 00 ...
PCR-13: 34 42 7B 49 32 23 ... <***>           PCR-13: 00 00 00 00 00 00 ...
PCR-14: A8 28 2F BD A7 BC ... <***>           PCR-14: 00 00 00 00 00 00 ...
```

Fig. 4. Boot-up PCR values of IBM T510 based on two different Operating Systems: identical hardware runninig on different Linux Operating Systems results differences in PCR 4,5 and 8 - 15