Policy Driven Remote Attestation

Anandha Gopalan, Vaibhav Gowadia, Enrico Scalavino, and Emil Lupu

Department of Computing
Imperial College London
180, Queen's Gate
London, SW7 2RH, U.K.
{a.gopalan,v.gowadia,e.scalavino,e.c.lupu}@imperial.ac.uk

Abstract. Increasingly organisations need to exchange and share data amongst their employees as well as with other organisations. This data is often sensitive and/or confidential, and access to it needs to be protected. Architectures to protect disseminated data have been proposed earlier, but absence of a trusted enforcement point on the end-user machine undermines the system security. The reason being, that an adversary can modify critical software components. In this paper, we present a policy-driven approach that allows us to prove the integrity of a system and which decouples authorisation logic from remote attestation.

Keywords: Remote Attestation, Trusted Platform Module, Policy based attestation.

1 Introduction

Governments, businesses and social organisations require the exchange of data between employees as well as with other organisations. This data is often sensitive and/or confidential but its exchange is vital for the successful functioning of these organisations. In particular, this becomes more important with more and more organisations and employees using mobile devices which increases the probability of the data falling into the wrong hands due the thefts of these devices.

Privacy and business confidentiality requirements demand that only authorised people should be granted access to sensitive data, and the usage of sensitive data needs to be controlled even after the data has been disseminated to data consumers. Data may reside at many locations such as server-side data stores, end-user machines, and portable disks. Controlling usage of sensitive data irrespective of its location requires that the sensitive data is always encrypted when stored or transmitted to force all access through a trusted Policy Enforcement Point (PEP). Only the PEP should be able to obtain decryption keys when permitted by a trusted Policy Decision Point (PDP).

Sandhu et al. [14,18] and Gowadia et al. [7] have described various security architectures to protect disseminated or shared data. A common factor among these architectures is the need for a trusted policy enforcement point. A potential threat is that an adversary can modify critical software components such

R. Prasad et al. (Eds.): MOBISEC 2011, LNICST 94, pp. 148–159, 2012.

as enforcement and policy evaluation components, or the underlying operating system. Such attacks can be mitigated in corporate environments by restricting the rights of users, so that they are unable to modify critical files on a system where sensitive data may be used. However, this assumption does not hold when dealing with a malicious insider, when the equipment is stolen or otherwise falls into the wrong hands. This is especially true in the case of using mobile devices due the high risk of theft. It is then necessary to increase the level of assurance provided. To provide a high assurance of system integrity we must rely on a trusted hardware component (e.g., the Trusted Platform Module (TPM), whose specifications are defined by the Trusted Computing Group (TCG) [23]). The TCG is working on the Mobile Trusted Module (v2.0) specification to improve mobile phone security.

The process of verifying integrity of remote systems using TPMs is called *remote attestation*. Remote attestation requires that a trusted verifier is able to verify integrity based on a digitally signed list of checksums provided by the data consumer. The process of verifying this evidence requires the verifier to maintain a large up to date database of acceptable software components that may exist on a data consumer's system. It is often desired that such a task be outsourced to a specialist. Managing trusted reference data for computing software measurements is a cumbersome task and by outsourcing the task to a specialist, we can reuse work done by the specialist verifier, thus reducing the cost of maintaining updates. However, out-of-the-box implementations of verifier software (e.g. [17]) only provide a boolean decision regarding a system's integrity. A decision about the system's integrity is often not sufficient to ensure that a program's behaviour will be as expected [8]. For example, acceptable program behaviour may depend on specific compositions of system components and their versions.

Considering the above requirements, an authorisation policy for accessing sensitive data should be able to specify whether remote attestation is required or not. If required, it should also specify who the trusted verifier is and which conditions should be satisfied by system components in addition to the conditions over subject and data attributes.

Different organisations may want to specify different constraints over system components. Therefore, a boolean result from a common verification authority is not sufficient. Instead, a verifier should provide functionality to allow a data provider to identify attributes of components on the data consumer's machine in addition to verifying their integrity. These attributes can then be utilised to evaluate authorisation policies within an organisation as needed.

In this paper, we present a policy-driven approach that allows us to specify the remote attestation requirements as part of authorisation policies. We also describe our architecture (and its implementation) that is used to prove the integrity of the system. A Trusted Platform Service (TPS) was designed and developed to allow easy integration of secure applications with the remote attestation module. The decoupling of remote-attestation and authorisation logic allows for greater flexibility for an organisation to integrate data protection

frameworks (such as Consequence [5]) with third-party attestation authorities. Our Trusted Platform service can be used by applications to handle remote attestation requests.

The major contribution of this paper is to illustrate the specification of remote attestation requirements as part of authorisation policies and to describe a modular implementation for easy integration of Trusted Computing technology with existing usage control systems. We believe that our approach and implementation on this platform will spur further investigation in this field.

The rest of the paper is organised as follows. Section 2 presents the overall architecture of our system. Section 3 presents the implementation details along with the various components that are used by our system. Section 4 presents research related to this paper, while Section 5 concludes the paper and provides ideas for future work.

2 Architecture

Evaluation of access rights on protected data requires identification of applicable policies and characteristics of the data. Therefore, metadata and policies are typically attached with the protected data during dissemination. In addition, the *content key* used for encrypting the data is encrypted with the public-key of a policy enforcement authority and also attached to the protected data. When a user requests access to the protected data, user and contextual attributes are also needed to evaluate the access request. These attribute values (as part of a credential / security token) must be provided by an authority trusted by the data provider or data owner.

Requirements for remote attestation of a system's integrity and state can be considered as contextual attributes and expressed as part of the usage control policies. An example policy using trusted credentials is shown below. Users can obtain the privileges for the "commander" role only if they present the required credentials. Access conditions can be further specified as part of authorisation policies (e.g. users must have an acceptable version of the glibc library).

```
authority A = "MyOrgCA"; authority V = "verifierCA";

credtype authn(uid, group);
credtype attestation(integrityCheck);
credtype version( glibcMajor, glibcMinor );

credential authnCred = authn signedby A;
credential attCred = attestation signedby V;
credential verCred = version signedby V;

role commander requires attCred.integrityCheck =="true"
and verCred and authnCred.group="officer" ;

authorization confidentiality0auth0 = allow read()
    target( dataCategory == "personalSensitive" )
    to commander
    when ( (verCred.glibcMajor == 2 and verCred.glibcMinor>5)
        or verCred.glibcMajor>2 );
```

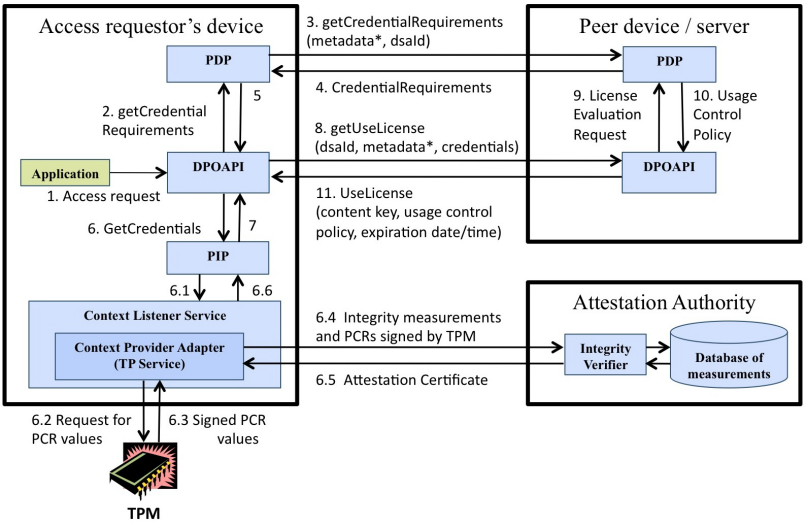

Fig. 1. Overall Architecture of the Remote Attestation Service

In Fig. 1, we show the interactions between components of our data protection framework. In this section, we describe how the framework can be integrated with a Trusted Platform Service (TPS). However, similar ideas may also be applied for integration with other dissemination control systems. The framework can be used to enforce *usage control* over data when data is shared within an organisation, and also if the data is shared across administrative domains using a *Data Sharing Agreement (DSA)*.

The Policy Enforcement Point (PEP) comprises a Data Protection Object (DPO) API and the application used to access the data. The DPO API is a generic enforcement component that can be used in different applications to enforce usage control. The framework contains both local and remote policy evaluation and enforcement components. The remote components play an important role in verifying credentials and evaluating policies for granting initial access to data. The local components are used to reevaluate policies and enforce a continuous control over the usage of data.

When a request to access protected data is made, the DPO API asks the Policy Decision Point (PDP) to determine the credentials needed to evaluate the access request. This is an optimisation step, as otherwise time may be spent on obtaining unnecessary credentials. The PDP then searches the policies associated with the protected data to determine the applicable ones on the basis of the metadata and requested action and determines the corresponding credential requirements. The credential requirements are expressed as pairs (credential type, issuer), where an issuer is the authority trusted to specify values for the credential type. For example, (authn, "MyOrgCA") is a requirement for a credential of the type *authn* that is signed by authority "MyOrgCA". The credential type is defined by a list of attribute names that must be present in a credential of that

type. The issuer or authority value, e.g. "MyOrgCA" is a unique name for the authority used by the Policy Information Point (PIP) to identify it.

The PIP dispatches requests for the credentials to security token services. The architecture allows configuration of multiple *context adapters* into the PIP. Each context adapter can interact with a particular type of credential provider and obtain the requested credentials. The integration of the remote attestation process with the data protection framework has been realised by creating a specific context adapter.

If remote attestation is required, the DPO API requests the Trusted Platform Service (TPS) to obtain credentials from the specified verifier. The TPS collects the checksums values of all applications and modules loaded on the data consumer's machine (using the Integrity Measurement Architecture (IMA) [17]), along with a cumulative checksum maintained by the trusted hardware component called the Trusted Platform Module (TPM). The checksums are sent to the remote attestation authority specified in the given credential requirement. The verifier validates the checksums from the data consumer's machine against a database of known checksums for the trusted applications and system software. The attestation authority verifies that the measurements are from acceptable applications and then it calculates an expected value for the aggregate checksum based on the measurements presented. If the calculated aggregate checksum matches with the value signed by the data consumer's TPM, then the attestation authority knows that the data consumer's system has loaded only trusted components. In such a case, the attestation authority can issue a certificate for the successful integrity check of the data consumer's system. In addition, the verifier can attest specific information (such as version number) about components that were relevant for the policy evaluation.

In the previous example, the policy requires a minimum version (but not the latest version) for the glibc library. Such a policy may be useful as the applications accessing the data may have a vulnerability when using an older version of the library. The intention of such a policy is to capture organisation-specific security requirements that need not be enforced by the verifier. This separation of responsibilities between the verifier and the organisation specific authorisation policies is referred to as *decoupling of remote attestation logic from authorisation logic.*

After obtaining the required credentials, the DPO API requests the decryption key from the policy enforcement service, which verifies the credentials and asks a policy-service to check whether the access is authorised. As shown in Fig. 1, a successful evaluation leads to the issue of a use-license (which includes the decryption key).

Our approach has the advantage of mitigating the possibility of granting access to systems with known vulnerabilities in a way that does not require the attestation authority to know the organisation policy. After the use-license has been released, the enforcement module at the data consumer's machine protects decryption keys using the secure storage feature of the TPM. To ensure the correct functioning of the system, it is critical to ensure the integrity of the PDP,

DPO API and PIP components. These must be verified during a secure boot procedure as mentioned in Schmidt et al. [20].

To cope with a system's state change after attestation, we require re-attestation of the system to obtain a use-license for each data item and to renew any existing use-licenses, which expire at a time governed by the usage control policy. In our work, we do not specify an integrity metric, since one may need to analyse a huge number of software components and identify which ones are critical to system security. This only furthers our case for outsourcing the attestation task to a specialist.

3 Implementation

In this section, we present the background and implementation details of the architecture presented in Section 2.

3.1 Trusted Platform Module

The Trusted Platform Module is a micro-controller chip located on the motherboard and contains cryptographic engines and memory (both persistent and volatile). The components of a TPM are shown in Fig. 2. The TPM specification is an industry specification released by the Trusted Computing Group [23]. A TPM provides sealed storage and remote attestation capabilities. It performs cryptographic computations internally, i.e. hardware and software components outside the TPM do not have access to the execution of crypto functions within the TPM hardware.

Cryptographic operations are performed in a TPM by using a cryptographic accelerator, an engine for SHA-1, a HMAC engine, a Key Generator and a Random Number Generator. This allows RSA encryption and decryption and can also be used to sign data. The Platform Configuration Registers (PCRs) are each 20-bytes long and are used to store measurements (SHA-1 hash values) of the hardware and software configurations of the platform. A PCR r (with value r_t at time t) is updated with a new measurement m by padding m to the existing value in r and then taking the hash (using SHA-1) of the resultant value. In particular, $r_{t+1} = SHA-1(r_t \mid\mid m)$.

To use the TPM functionalities such as TPM Quote (which produces a signed composite hash of the selected PCRs and external data, such as a nonce), the TPM needs to be set up with an Attestation Identity Key (AIK) and it's appropriate credentials. This key will be used to sign PCR values (such as from TPM Quote) and the associated credential will guarantee that the quote is coming from a genuine TPM. Using the platform credentials of the TPM, an AIK was created apriori by using the services provided by PrivacyCA [15].

3.2 Integrity Measurement Architecture

The runtime system of the device used by the requester must be attested to ensure its integrity. Attestation requires measurement of all components from the

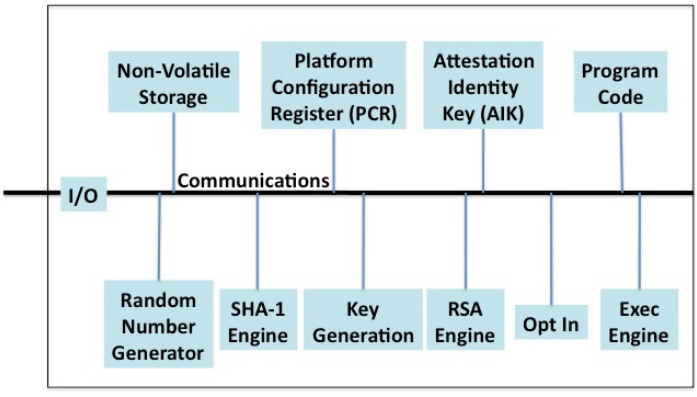

Fig. 2. Trusted Platform Module (TPM)

boot process up to the application layer. This is achieved by using the Integrity Measurement Architecture (IMA) [17]. To aid in taking measurements, hooks are placed in the Linux kernel. All executable content, as well as application-related file content (configuration files, libraries, etc.) are measured (a SHA-1 checksum of the file is taken) before they are loaded. The files measured include: kernel modules, executables, configuration input files. Each time a file is measured, a given PCR (*PCR-10*) is extended with the new value, and this value is also added to the measurement list that is stored in the kernel. This measurement list is also accessible using the filesystem. To attest a platform, the value of *PCR-10* along with its measurement list is sent to the verifier (Attestation Authority) who can check the state of the software stack. We use the Integrity Measurement Architecture that is available as part of the Linux Kernel (version 2.6.32).

3.3 Trusted Boot

A basic principle followed in trusted platform technologies is to verify the integrity or trust of every critical component before it is executed or loaded. Therefore, in addition to checking the runtime integrity of a system, we must also ensure that the system was booted correctly and is running the appropriate operating system. This is achieved by building a chain of trust starting with the Core Root of Trust for Measurement (CRTM), which is a trusted code in the BIOS boot block. It reliably measures integrity values of other entities, and stays unchanged during the lifetime of the platform. CRTM is an extension of normal BIOS which is run first to measure other parts of the BIOS block before passing control. The BIOS then measures hardware and the bootloader and passes control to the bootloader. The bootloader measures the OS kernel image and passes control to the OS. Each step of the boot process extends the appropriate PCR value in the TPM with the measurements taken in that step. These measurements attest the integrity of the system. For the purpose of our implementation, we used Trusted Grub v1.1.5 [24] as our boot loader and Linux kernel 2.6.32. This process is shown in Fig. 3.

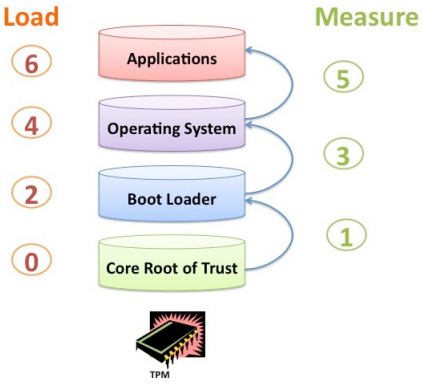

Fig. 3. Trusted Boot Process

3.4 Trusted Platform Service

The Trusted Platform Service (TPS) provides the enforcement architecture with two functions to: (i) verify the integrity of the system and (ii) protect encryption keys. The Trusted Platform Service is implemented in C and modelled as a "listener" service that waits for requests on a particular port. It provides the functionality of interacting with system-level libraries that measure system state and is also responsible for interactions with the verifier. This service uses the TrouSerS (Trusted Computing Software Stack) API [22] for accessing the various functionalities of the TPM. The verification of the software stack is done using the Trusted Platform Module, along with kernel and boot-loader improvements, and the Integrity Measurement Architecture (IMA) [17].

When the Trusted Platform Service receives a request from the PIP, it initiates the verification process. The request from the PIP contains information about the verifier (including the name and port number) and using this the TPS contacts the verifier. Upon receipt of a nonce (to ensure freshness) from the verifier, the TPS packages the required data and sends it to the verifier. The packaged data consists of: (i) TPM Quote (signed by the TPM), (ii) TPM credentials, (iii) values of selected TPM PCRs (chosen depending on what we wish to check - this includes *PCR-10*), and (iv) IMA measurement list.

3.5 Attestation Authority

The Attestation Authority is implemented in C and modelled as a "listener" service that waits for requests on a particular port. The Attestation Authority is provided with a database of expected checksum values of various programs. It is against this list that incoming requests are checked. Upon receiving a request from the Trusted Platform Service, the Attestation Authority replies back with a nonce. The resultant response from the Trusted Platform Service is the data blob containing the results of the TPM Quote function along with the TPM credentials, as well as the IMA measurement list. The Attestation Authority

initially verifies the TPM Quote (using the credentials provided for the TPM), after which the PCR values are checked. To verify the integrity of the software stack, the IMA measurement is used to re-compute the value of *PCR-10* and it is compared to the received value. Finally, the hash values of the applications (and their versions) are checked against the available database.

Depending on the outcome of the verification process, the Attestation Authority will send back an Attestation certificate (signed by it to ensure authenticity) certifying the required credentials (as specified by the authorisation policy). We use X.509 certificates for this purpose. This method of attestation allows for decoupling between the evaluation of authorisation policies (at the data consumer's side) and remote-attestation at the Attestation Authority.

4 Related Work

Remote attestation is an integral part of Trusted Computing and the ability to verify the software and/or hardware running on a machine is of paramount importance. There has been however, some criticism about whether this is a viable or impractical solution, since it requires the attesting authority (or verifier) to know apriori the needed software configurations as well as their checksums. Lyle et al. show that this is indeed a viable solution for web services [11]. Our remote attestation scheme is generic enough to work with web services as well as other technologies.

The Integrity measurement architecture (IMA) proposed by Sailer et al. [17] uses binary attestation to measure all the programs and code before they are loaded into the system to run. This architecture was extended by Jaeger et al. in [9], where the proposed architecture only measures the programs and code that are needed by the verifier. This was due to the fact that the verifier does not necessarily need to know the measurements of all the programs that are running on the system. In our work, we use IMA because we do not distinguish between what is required by the verifier and that which is required by the enforcement layer. This is due to the fact that these two are decoupled and different third-party verifiers could be used for our purpose.

In the event that the software platform cannot guarantee that the software it is running is reliable, it is advisable to move the required program code away from the untrusted platform. The technique used for this is called code-splicing, which involves splitting the program code into "critical" and "non-critical" sections, so that only non-critical code is run on the untrusted platform. This in turn guarantees that code that is critical has not been tampered with. These ideas have been presented by Ceccato et al. [3], Dvir et al. [6] and Zhang et al. [27].

Kennell et al. proposed a system called "Genuinity", which verifies if the hardware and the software running on a system are "genuine" [10]. This is achieved without the use of any special hardware and by using a timed execution of a checksum function that provides a fingerprint of the running applications. The time taken to execute the checksum is verified by the verifier. As shown by Shanker et al. [21], this is not a viable solution due to the assumptions that are to be imposed and it is also prone to substitution attacks.

Using the idea of calculating a checksum, Schellekens et al. [19] proposed a system where the time stamping functionality of the TPM is used to calculate the execution time of the checksum locally. This is then used in conjunction with a timing based remote attestation mechanism to prove the integrity of the system.

Alawneh et al. propose an architecture to protect data within an organisation [1]. Their threat model is that of a rogue employee who can potentially disseminate data to the outside world. The proposed system binds the sensitive content within the organisation to specific devices, thereby restricting the content from being leaked to other devices. In the case when devices need to share information, these devices are allocated to the same dynamic domain.

Although the aforementioned systems all provide varying degrees of remote attestation and verification, none of them decouples the attestation process from the enforcement process. This allows for more flexibility with respect to the type of remote attestation chosen, as well as more fine grained access control at the enforcement layer.

Sadeghi et al. [16] provide reasons as to why binary attestation may not be the most useful form of attestation. They argue that rather than using the checksums of the software programs on a machine, it is better to asses the state of a platform based on some pre-determined security properties. This type of attestation that uses the security property rather than binary attestation is referred to as property-based attestation. Property-based attestation uses binary attestation to prove some "property" of the system (for example, if certain hash values match, then we can state that the system is in a particular state). There have been several techniques that have been proposed for property-based attestation [4,13]. In [12], the authors have proposed the idea of a Property Manifest, which can be used to define security policies for policy-based attestation.

Haldar et al. proposed the concept of semantic remote attestation in [8], wherein the security of the system is guaranteed through program analysis. To verify whether a program's execution will satisfy the security properties, the authors analyse the program by using a Trusted Virtual Machine.

Though both property-based attestation and semantic remote attestation provide the techniques to attest whether a remote machine is trusted or not, neither of these decouple the enforcement layer from the attestation authority. Also, this decoupling allows our system to use any underlying attestation mechanism and this adds to the flexibility of the presented system.

In [26], Yu et al. propose a system for guaranteeing the freshness of the integrity measurement that is used in the attestation. This proposed solution, called RTRA, uses the run-time state of the attesting party. RTRA can be directly integrated into our architecture by using it as the underlying attestation mechanism.

Often privacy is also a concern when sharing system measurements with the verifier. Our scheme may be used together with privacy preserving protocols such as Brickell et al.'s attestation scheme [2]. The high-level description of their scheme implies changes in the cryptographic protocol between IMA, TPM, and

the verifier. However, our architecture resides at a higher-level and is independent of the cryptography specifics used (e.g. signature schemes).

The goals of Trusted Network Connect [25] seem very similar to our work. In their architecture the integrity measurement verifiers act as the PDP, which is different from our architecture as we separate the PDP and verifier. They also have the problem of policies being evaluated by the verifier itself.

5 Conclusion and Future Work

In this paper, we have presented a policy-driven approach that allows us to prove the integrity of a system while decoupling authorisation logic from remote attestation. This decoupling of remote attestation and authorisation logic allows for greater flexibility for an organisation to fine tune their authorisation policies by using different data protection frameworks with third-party attestation authorities. This system relies on a trusted hardware component, the Trusted Platform Module and uses the Integrity Measurement Architecture. In the future, we would like to evaluate our system with different policy-based data protection frameworks as well as with different attestation mechanisms.

References

1. Alawneh, M., Abbadi, I.M.: Sharing but protecting content against internal leakage for organisations. In: DBSec, pp. 238–253 (2008)
2. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, pp. 132–145. ACM, New York (2004)
3. Ceccato, M., Preda, M., Nagra, J., Collberg, C., Tonella, P.: Barrier slicing for remote software trusting. In: Seventh IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2007, September 30-October 1, pp. 27–36 (2007)
4. Chen, L., Landfermann, R., Löhr, H., Rohe, M., Sadeghi, A.-R., Stüble, C.: A protocol for property-based attestation. In: Proceedings of the First ACM Workshop on Scalable Trusted Computing, STC 2006, New York, NY, USA, pp. 7–16 (2006)
5. Consequence Project, `http://www.consequence-project.eu/`
6. Dvir, O., Herlihy, M., Shavit, N.: Virtual leashing: Internet-based software piracy protection. In: Proceedings of 25th IEEE International Conference on Distributed Computing Systems, ICDCS 2005, pp. 283–292 (June 2005)
7. Gowadia, V., Scalavino, E., Lupu, E.C., Starostin, D., Orlov, A.: Secure cross-domain data sharing architecture for crisis management. In: Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management, DRM 2010, New York, NY, USA, pp. 43–46 (2010)
8. Haldar, V., Chandra, D., Franz, M.: Semantic remote attestation - a virtual machine directed approach to trusted computing. In: USENIX Virtual Machine Research and Technology Symposium, pp. 29–41 (2004)
9. Jaeger, T., Sailer, R., Shankar, U.: Prima: Policy-reduced integrity measurement architecture. In: Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies, SACMAT 2006, New York, NY, USA, pp. 19–28 (2006)

10. Kennell, R., Jamieson, L.H.: Establishing the genuinity of remote computer systems. In: Proceedings of the 12th Conference on USENIX Security Symposium, vol. 12, pages 21. USENIX Association, Berkeley (2003)
11. Lyle, J., Martin, A.: On the feasibility of remote attestation for web services. In: International Conference on Computational Science and Engineering, CSE 2009, vol. 3, pp. 283–288 (August 2009)
12. Nagarajan, A., Varadharajan, V., Hitchens, M., Arora, S.: On the applicability of trusted computing in distributed authorization using web services. In: DBSec, pp. 222–237 (2008)
13. Nagarajan, A., Varadharajan, V., Hitchens, M., Gallery, E.: Property based attestation and trusted computing: Analysis and challenges. In: NSS, pp. 278–285 (2009)
14. Park, J., Sandhu, R.S., Schifalacqua, J.: Security architectures for controlled digital information dissemination. In: Proc. of ACSAC, p. 224 (2000)
15. PrivacyCA, http://www.privacyca.com/
16. Sadeghi, A.-R., Stüble, C.: Property-based attestation for computing platforms: caring about properties, not mechanisms. In: Proceedings of the 2004 Workshop on New Security Paradigms, NSPW 2004, New York, NY, USA, pp. 67–77 (2004)
17. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a tcg-based integrity measurement architecture. In: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM 2004, vol. 13, pages 16. USENIX Association, Berkeley (2004)
18. Sandhu, R.S., Ranganathan, K., Zhang, X.: Secure information sharing enabled by Trusted Computing and PEI models. In: ASIA CCS, pp. 2–12 (2006)
19. Schellekens, D., Wyseur, B., Preneel, B.: Remote attestation on legacy operating systems with trusted platform modules. Sci. Comput. Program 74, 13–22 (2008)
20. Schmidt, A.U., Leicher, A., Cha, I., Shah, Y.: Trusted platform validation and management. International Journal of Dependable and Trustworthy Information Systems (IJDTIS) 1(2), 1–31 (2010)
21. Shankar, U., Chew, M., Tygar, J.D.: Side effects are not sufficient to authenticate software. In: Proceedings of the 13th USENIX Security Symposium, pp. 89–101 (2004)
22. TrouSerS - The open-source TCG Software Stack, http://trousers.sourceforge.net/
23. Trusted Computing Group, http://www.trustedcomputinggroup.org/
24. Trusted Grub, http://sourceforge.net/projects/trustedgrub/
25. Trusted Network Connect, http://www.trustedcomputinggroup.org/files/resource_files/51F9691E-1D09-3519-AD1C1E27D285F03B/TNC_Architecture_v1_4_r4.pdf
26. Yu, A., Feng, D.: Real-Time Remote Attestation with Privacy Protection. In: Katsikas, S., Lopez, J., Soriano, M. (eds.) TrustBus 2010. LNCS, vol. 6264, pp. 81–92. Springer, Heidelberg (2010)
27. Zhang, X., Gupta, R.: Hiding program slices for software security. In: Proceedings of the International Symposium on Code Generation and Optimization: Feedback-Directed and Runtime Optimization, CGO 2003, pp. 325–336. IEEE Computer Society, Washington, DC (2003)