Distributed Usage Control Enforcement through Trusted Platform Modules and SGX Enclaves

Paul Georg Wagner
Karlsruhe Institute for Technology,
Karlsruhe, Germany
paul.wagner@student.kit.edu

Pascal Birnstill
Fraunhofer IOSB, Karlsruhe, Germany
pascal.birnstill@iosb.fraunhofer.de

Jürgen Beyerer
Fraunhofer IOSB, Karlsruhe, Germany
juergen.beyerer@iosb.fraunhofer.de

ABSTRACT

In the light of mobile and ubiquitous computing, sharing sensitive information across different computer systems has become an increasingly prominent practice. This development entails a demand of access control measures that can protect data even after it has been transferred to a remote computer system. In order to address this problem, sophisticated usage control models have been developed. These models include a client side reference monitor (CRM) that continuously enforces protection policies on foreign data. However, it is still unclear how such a CRM can be properly protected in a hostile environment. The user of the data on the client system can influence the client's state and has physical access to the system. Hence technical measures are required to protect the CRM on a system, which is legitimately used by potential attackers. Existing solutions utilize Trusted Platform Modules (TPMs) to solve this problem by establishing an attestable trust anchor on the client. However, the resulting protocols have several drawbacks that make them infeasible for practical use. This work proposes a reference monitor implementation that establishes trust by using TPMs along with Intel SGX enclaves. First we show how SGX enclaves can realize a subset of the existing usage control requirements. Then we add a TPM to establish and protect a powerful enforcement component on the client. Ultimately this allows us to technically enforce usage control policies on an untrusted remote system.

CCS CONCEPTS

• **Security and privacy → Access control**; *Privacy-preserving protocols*; *Digital rights management*; *Information flow control*;

KEYWORDS

Usage Control, Access Control, Trusted Reference Monitor, Trusted Platform Module, SGX, Secure Remote Computation

ACM Reference Format:
Paul Georg Wagner, Pascal Birnstill, and Jürgen Beyerer. 2018. Distributed Usage Control Enforcement through Trusted Platform Modules and SGX Enclaves. In *SACMAT '18: The 23rd ACM Symposium on Access Control Models & Technologies (SACMAT), June 13–15, 2018, Indianapolis, IN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3205977.3205990

1 INTRODUCTION

Most modern computer systems rely on mechanisms that can restrict the access to certain system resources like files and services. Especially in the context of mobile and ubiquitous computing, digitally managing the access to sensitive information clearly plays an essential role in designing data processing systems that are both secure and privacy friendly. Traditional access control models are generally implemented through a reference monitor that is invoked whenever a subject requests access to a particular object. The reference monitor then evaluates available access control policies and enforces the resulting access control decision on the subject. However, for some use cases it is not sufficient to merely control the access to information once at the time of data request. Sometimes a generalized model is necessary that can continuously monitor and control the actual *usage* of information over an extended period of time. Such a model includes solutions to many questions of digital rights management (DRM) and it may also allow for the implementation of privacy enhancement technologies that monitor and control the usage of personal data after it has been released. To adequately reflect these important requirements, Park and Sandhu [8] developed the notion of usage control (UC). Furthermore, distributed usage control models have been proposed [9] that tackle the problem of transferring information across different usage control domains. In these scenarios, the data access should be continuously controlled on a remote computer system even after the data has left the domain of the data provider. In order to achieve this, the data provider deploys usage control policies to a client-side reference monitor (CRM) before the data access is granted on the server. Afterwards the CRM is responsible for enforcing the deployed policies on the data. The policies that should be enforced on the client side can be of different nature.

- Restricting access: The user of the client system should only be allowed to access the files in certain situations or at certain times.
- Usage control: The user of the client system may be allowed to access the data, but their use is restricted. For example, the user should not be able to disseminate the data further.
- Secure computation: The user must never access the transmitted data directly. Instead, a trusted module on the client side performs a computation on the data, after which the user can get access.

While most usage control models focus on policies that mainly reflect the former two cases, the latter is of great use when implementing privacy protecting mechanisms. For example, a provider of personally identifiable information could issue usage control policies that enforce data anonymization on the client before the critical information is used otherwise. With powerful usage control models already established, it is still an open question how to implement and protect a client-side reference monitor in a possibly hostile environment. Previous suggestions utilize Trusted Platform Modules (TPMs) to establish a trusted computing base, but this solution has drawbacks and is not sufficient to appropriately protect the transmitted data. This work proposes a CRM implementation that establishes trust by using TPMs along with Intel SGX enclaves running on an untrusted system. Ultimately this allows us to technically enforce usage control policies on a remote system, whose user we do not trust. In section 2 we will present existing propositions of a CRM design, which are based on TPMs, and discuss their drawbacks. In section 3 we present a simple CRM design that uses the well researched secure remote computation features of Intel SGX. However, this design cannot enforce arbitrary usage control policies. Afterwards in section 4 we generalize the design by adding another TPM, which yields a solution that can enforce all mentioned usage control policies. We finally conclude in section 5.

2 RELATED WORK

Implementing a reference monitor in a possibly hostile environment requires a Trusted Computing Base on the target system. This can be achieved by a Trusted Platform Module (TPM). A TPM is a dedicated hardware chip that extends a computer with basic security related features [6]. It uses volatile platform configuration registers (PCRs) to *measure* the current hardware and software configuration as an unforgeable hash. This allows the system to *seal* confidential data to a certain TPM state. Furthermore, remote parties can verify that the target system is in a certain state by *attesting* to certain PCR values. Hence a TPM can be used to protect system components in an untrusted environment.

Sandhu and Zhang [11] introduced the notion of a trusted reference monitor (TRM) inside the client-side operating system. The TRM is a reference monitor that operates in an untrusted environment, but is protected from external modification by a TPM. Implemented as a kernel module, the TRM is part of the measurement chain during the boot process. Before transmitting any data or policies, the data provider remotely attests to the PCR values of the client system. Only if the remote system is in a trustworthy state (i.e. the TRM is unmodified and running), information is transmitted. Since we focus on implementing a trusted reference monitor on client systems, in the following sections we will use TRM and CRM synonymously. Based on the work of Sandhu and Zhang, Sevinç et al. [12] developed a protocol that relies on a TPM to remotely verify the integrity of the client software stack. In this protocol, secrets are only transmitted to the client if the attestation is successful and the remote system can show the correct PCR values. Furthermore, the server binds the secret data to a key that is sealed to the required PCRs. That way the transmitted data can only be unsealed and used as long as the client system is in a trustworthy state. However, by relying only on TPMs, Sevinç's protocol has several drawbacks that have not yet been addressed. For example, the TPM cannot distinguish between trusted and untrusted processes. Even in trusted system states (i.e. the TRM is running and has not been tampered with) there will be untrusted user processes active in the system. In that case only trusted processes, such as the TRM itself, should be able to unseal the data. If the sealed data is intercepted during transmission, or is in any way available later on, any user process can request the TPM to unseal the data if only the PCRs still have the correct values. This bypasses TRM control on a software level. In order to distinguish trusted from untrusted processes, the TRM design may include operating system based protection mechanisms, such as access rights on files and directories. There are techniques available to include executable content and security extended file attributes into the TPM measurement chain, such as the integrity measurement system and the extended verification module for Linux [10]. However, since in this case the user of the client system is an attacker, TPM based mechanisms are not sufficient to protect the sealed data that way. The user can mount the hard drive and access the sealed data in a secondary operating system. Even though the sealed data cannot be unsealed in this untrusted system state, the user can still make a copy of the encrypted data without changing the original file meta data. When booting the unmodified operating system, the PCRs are filled with the correct values and the user can unseal the copied data.

To conclude, the proposed solutions for a secure TRM implementation rely on establishing trust using a TPM, but are not sufficient to properly protect the transmitted data. This is mainly a result of the attacker model, which includes valid users of the client system itself, who can use the TPM, launch untrusted system processes and have physical access to the hardware.

3 IMPLEMENTATION WITH INTEL SGX

Intel's Software Guard Extensions (SGX) consist of a set of processor instructions extending the x86 architecture, along with special hardware that is included in newer Intel CPUs. SGX can provide integrity and confidentiality, even if privileged software such as the operating system is malicious. This is achieved by executing user code in a protected container called *enclave*, which cannot be accessed by other user processes or even by the operating system itself. The enclave is executed by trusted hardware and is isolated from the rest of the system (reverse sandboxing). It uses encrypted memory to protect the confidentiality of data and can verify the integrity of the code by communicating with the Intel Attestation Service (IAS). Because the runtime state of the enclave cannot be influenced from the outside, SGX represents a trusted computing design. SGX allows to encapsulate critical software, for example cryptographic libraries or key management services, in protected shells that will behave in expected ways. Architectural details of SGX and a comprehensive analysis of its security is provided in [5].

3.1 Secure Remote Computation

SGX has been designed to ease the implementation of secure remote computation. Secure remote computation is the problem of using a remote computer, owned by an untrusted party, to perform some computation on certain confidential data. In our case a remote service provider needs to provision secrets to untrusted clients, who run trusted code inside an enclave. Before transmitting the data, the service provider has to convince himself that he is communicating with a certain enclave, which is running in a secure environment. With SGX-enabled processors, this can be achieved by initiating a remote attestation process. This process consists of four phases [1].

(1) *Enclave Launch*: The untrusted system launches code inside an enclave. During the launch, code and data are cryptographically hashed. This hash is called the enclave's *measurement*.

(2) *Attestation*: The enclave contacts the remote service provider and signals that it is ready for provisioning. The enclave produces a signed quote that includes the enclave measurement. This quote is sent to the service provider.
(3) *Provisioning*: The service provider verifies the quote by contacting the IAS. This ensures that the service provider is indeed communicating with the correct enclave. The service provider uses the attestation protocol to establish a secure channel to the enclave and transmits the sensitive data.
(4) *Sealing/Unsealing*: The enclave receives the sensitive data and seals it to its current state. Sealed data is encrypted and can be securely stored outside the enclave (e.g. in files). It can only be decrypted by the same enclave in the same state.

After the attestation process, the enclave received all necessary data and performs the desired computations while protecting both confidentiality and integrity. The necessary secure channel is established by a modified Sigma protocol that facilitates a Diffie-Hellman Key Exchange (DHKE) between the enclave and the service provider. During the remote attestation process the enclave sends a quote containing its enclave measurement to the service provider. The service provider forwards the quote to the IAS, where it can be verified. If the quote verifies correctly, the service provider is confident that he communicates with the right enclave (measurement is correct) and that only the enclave knows the established Diffie-Hellman key. Both parties can then derive a symmetric secret from the Diffie-Hellman key and use it to encrypt their communication for the provisioning phase. In [7] the remote attestation protocol is explained in greater detail. A similar protocol is also possible between two enclaves that reside on one SGX platform. This is called *local attestation*. It can be used to locally verify the integrity of another enclave and establish a secure channel between them.

3.2 TRM Design

Trusted Reference Monitors can be implemented on SGX-enabled processors by using the remote attestation functionality. The TRM is realized as a trusted enclave that is running on the client. A data provider (DP) can remotely attest to the identity and integrity of the enclave (via the IAS), and hence establish trust in the remote reference monitor. The sigma protocol instance authenticates the channel and gives both sides a shared secret, which the data provider uses to encrypt the data. The untrusted software on the client acts as a man in the middle, but can neither decrypt nor modify any messages between enclave and data provider. Once the secure channel has been established, the data provider deploys the signed access control policies at the remote TRM and transmits the encrypted data. On the client side, the enclave verifies the signature of the policies and seals the received data. Untrusted processes can request data access at the enclave, which evaluates the policies and enforces the resulting access control decisions. Figure 1 shows the resulting sequence of data request, transmission and access control enforcement at the client. Of course, the TRM can also enforce complex access control policies that require computation on the data before access can be granted. For example, a provider of personally identifiable information can instruct the TRM to remove personal information from the data for certain requests.

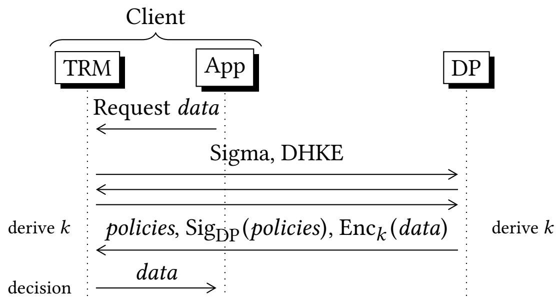
Client
TRM
App
DP
Request *data*
Sigma, DHKE
derive k
policies, $\text{Sig}_{\text{DP}}(\textit{policies})$, $\text{Enc}_k(\textit{data})$
derive k
data
decision

Figure 1: TRM implementation with SGX.

3.3 Security Analysis

In terms of the attacker model we have to distinguish between *internal* and *external* attackers. Internal attackers are the untrusted parts of the client system, outside the trusted enclave. This includes any untrusted user process running on the client, as well as privileged software like the operating system. The goal of internal attackers is to bypass the policy enforcement and directly access the protected data without access control restrictions. External attackers reside outside the client system and can intercept and modify messages between the data provider and the client. Their goal is to extract information about the protected data from the exchanged messages.

Based on this attacker model, the security of the TRM design immediately reduces to the design of the SGX remote attestation procedure as well as the security of the used cryptographic schemes. The integrity of the enclave code is ensured because the data provider verifies the quote containing the enclave measurement with the IAS. Only the TRM enclave is able to generate a quote that is correctly signed. An internal attacker, who tampers with the TRM code before launching the enclave, changes the enclave measurement in the process, which will fail the quote verification step on the data provider. The communication channel between the data provider and the trusted enclave is established by the sigma protocol. The resulting channel is authenticated by digital signatures provided by the enclave and the data provider. The data provider signs his messages with his private key. The enclave can verify the signatures by using the data provider's public key that is usually hard coded into the enclave software. Since the public key is part of the enclave code, its integrity is protected by the enclave measurement. The enclave is authenticated by the correct quote signature. The quote contains the public part of an asymmetric key pair that the enclave generated for communicating with the data provider. This key is used for authenticating further messages to the data provider. Furthermore, a shared Diffie-Hellman key is established between the enclave and the data provider. Since the channel is authenticated, no external or internal attacker is able to intercept the Diffie-Hellman key exchange and hence does not know the shared secret after the sigma protocol finishes. Under the assumption that the key derivation function and the used encryption scheme are secure, the protected data cannot be decrypted on the way to the enclave. On the client system, the SGX-enabled hardware ensures that no internal attacker can access any data residing in an enclave. The integrity of the included access control policies is assured by the digital signature of the data provider. Since the SGX specification includes hardware protection mechanisms, such as reserved memory areas for enclaves, physical attacks become less feasible for internal attackers. The security of the remote attestation and the underlying SGX design are analyzed in greater detail in [1, 5].

3.4 Problems

Since the security of the proposed design is based on the security of the SGX architecture, attacks against SGX hardware or protocols immediately impact the TRM implementation. As shown in [3, 14], SGX has certain weaknesses against side channel information leakage attacks. This could allow internal attackers to extract knowledge about enclave data. However, recent research has attempted to prevent such information leakage by detecting external intervening in enclave execution [4, 13]. Another problem is the remote policy deployment mechanism of the access control system. In order to remotely deploy new policies at the TRM, the data provider needs to connect to the client system and transmit the encrypted policies. An internal or external attacker could block respective messages from the data provider to the TRM, and thereby prevent policy updates. In that case the TRM does not know about the new policies and will continue to evaluate old ones for access control decisions. The data provider will notice the failed policy deployment, but has no way of notifying the enclave if the attacker severs all communication. This is a general problem of distributed access control systems and it also affects access right revocation. A possible solution is for the TRM to regularly query policy updates at the data provider and deny accesses if the data provider is not reachable. The downside of this solution is the increased communication overhead. It also requires a suitable challenge-response protocol in order to prevent replay attacks. Finally, implementing a TRM inside an enclave leaves only limited options with regard to policy enforcement. Most importantly, the TRM cannot enforce policies on data that has been released outside the enclave. Once an access is granted and data leaves the enclave into untrusted space, no further control on the copied data is possible. Therefore this TRM design is sufficient for remote access control, but not for cross-domain usage control. Of course, the TRM can also enforce computations on the protected data (e.g. remove sensitive personal information) before they are released without further restrictions. However, due to the isolation of the enclave against the rest of the untrusted system, the TRM implementation has only limited resources available. Enclaves can only use a modified version of the standard C library, along with a limited amount of protected memory. Hence it is unfeasible to perform complex calculations (e.g. anonymize privacy impacting images) inside an enclave. All in all, using SGX to protect a TRM results in a design that can remotely enforce access control policies and provide secure computation. However, this solution is not sufficient for enforcing distributed usage control in general.

4 IMPLEMENTATION WITH SGX AND TPM

As shown in the previous sections, neither TPMs nor SGX enclaves alone are sufficient to securely implement a powerful TRM on a remote client system. TPMs offer a single isolation container that covers all the software running on the computer, including the operating system. This makes it possible to protect the integrity of kernel modules that can enforce complex policies on a low system level. Such a powerful kernel level component is necessary to implement distributed usage control systems. However, TPMs cannot sufficiently protect the integrity of policies and the confidentiality of data in this use case (see section 2). SGX-enabled processors can be used to implement a TRM design based on enclaves. A TRM enclave can use the SGX remote attestation protocol to perform remote policy deployment and secure remote computation. On the other hand, the SGX enclave is technically isolated from the rest of the system, which makes it impossible to enforce usage control policies on data outside the enclave. Therefore SGX enclaves cannot implement a TRM for general usage control purposes. A possible solution is to combine SGX enclaves with a separate TPM that is protecting the rest of the system. An external enforcement component is realized as a standard kernel module and gets measured by a separate TPM. The TRM itself consists of SGX enclaves that communicate with the data provider and evaluate the deployed policies. The enclaves verify the integrity of the external enforcement component using the local attestation mechanism of the TPM.

4.1 TRM Design

We base our TRM design on the well researched XACML architecture [2]. The TRM components are shown in figure 2.

Policy Enforcement Point. The PEP is the enforcement component of the TRM. It intercepts data accesses in the system and enforces usage control decisions on it. The PEP cannot be isolated in a SGX enclave, because it needs to track data flows and enforce policies throughout the system. Hence the PEP is implemented as a dedicated kernel module. Its integrity is protected by a TPM measurement during the boot sequence.

Policy Decision Point. The PDP evaluates a set of deployed policies for each requested data access in order to reach an access decision for the request. It is implemented as a trusted SGX enclave. It communicates with the external PEP to announce access control decisions. The PEP notifies the PDP about an intercepted event, the PDP evaluates the respective policies and returns the resulting decision, which the PEP enforces on the requester.

Policy Retrieval Point. The PRP is the part of the TRM that securely receives policies and data from remote data providers. It is also the main component that establishes trust with the remote data provider. This includes verifying the integrity of the other parts of the TRM, especially the PEP and the PDP. The PRP is implemented as an enclave on the client system and can use the SGX remote attestation protocol to prove its integrity to the data provider.

Policy Information Point. The PIP is a user space application that provides information about the current system state for the policy evaluation. It encapsulates the PDP and PRP enclaves and acts as an interface between PEP and PDP. The PIP holds a data flow model, which maps a representation of the protected data existing on the computer to a set of containers, e.g. files or user processes that contain this information. The data flow model gives a comprehensive overview of the current system state with regard to the distribution of protected data. The PIP is notified by the

PEP about system events that can initiate a data flow, for example certain system calls, and updates the data flow model accordingly. The PDP can use this information in order to come to an access control decision for a particular access request.

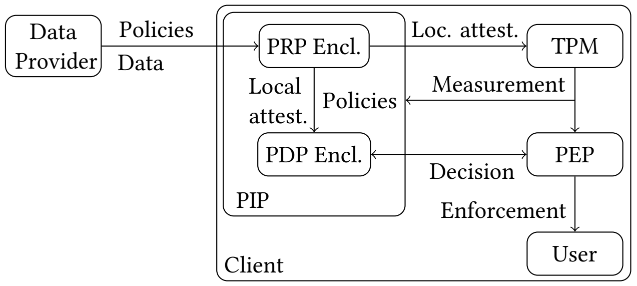

Figure 2: TRM design with TPM and SGX.

The TRM protocol consists of four phases (see figure 3).

(1) *Initialization:* During the boot sequence of the client system, the TPM measures the PEP kernel module as well as the PIP application in oder to detect tampering. The PIP application launches the PRP and the PDP enclaves.
(2) *Establishing trust:* In order to verify the integrity of the client system, a remote data provider initiates the remote attestation protocol with the PRP enclave. Three sigma messages S1-S3 are exchanged between the data provider and the PRP enclave (see section 3.1). If the sigma protocol is successful, the data provider trusts the remote PRP implementation and a shared symmetric key *sk* is established. The PRP enclave is responsible for verifying the integrity of the other trusted components of the TRM. First, the PRP checks the PCR values of the external TPM in order to establish trust in the PEP and the PIP implementations. If the PCR values match a known value, the components are unmodified and can be trusted. Then, the PRP uses the SGX local attestation protocol to verify the state of the PDP enclave. If this attestation is successful, the PDP enclave has been launched unmodified. In that case a transitive trust relationship is established, since the data provider trusts the PRP, and the PRP trusts the PDP and PEP. Finally, the trusted PRP notifies the data provider that the remote platform is trustworthy.
(3) *Policy deployment:* The data provider uses the established secure channel to transmit the protection policies to the PRP enclave. The policies are signed by the data provider and the PRP verifies the signature. Then the PRP deploys the policies to the PDP, using the locally established secure channel between the two enclaves. After the protection policies have been successfully deployed on the remote system, the data provider transmits the sensitive data to the PRP enclave. In order to prevent information leakage, the protected data is encrypted with the shared symmetric key. Finally, the PIP is notified about the existence of new data and updates its data flow model accordingly.
(4) *Policy enforcement:* The active PEP kernel module intercepts data requests of users in the system and notifies the PIP about respective events. The PIP can update the mappings of its data flow model according to the triggered event (e.g. a read() system call), and then forwards the event to the PDP enclave for decision. The PDP evaluates the deployed policies for the intercepted event and may query the data flow model in the process. The resulting access control decision is returned via the PIP to the PEP, which finally enforces it on the requester. Unlike in the XACML architecture, the PIP is also responsible for initially forwarding the protected data to the requester. A user process can request access to the data that the remote data provider has transmitted to the client system. If the PDP access control decision is positive, the PIP releases the protected data. Afterwards, the usage of the released data in the memory of untrusted user processes is supervised by the PEP. The PIP then merely updates the data flow model by changing the mapping of the data to containers based on the intercepted PEP events.

4.2 Security Analysis

The attacker model for this TRM design is the same as for the previous, SGX-based solution. We distinguish between internal attackers, most importantly the user of the client system, and external attackers that eavesdrop on the communication between data provider and client. However, unlike in the previous section, we now assume that an internal attacker does not have root access to the client system. The most important issue with this TRM design is how the trust relationship between the remote data provider and the TRM on the client system is established. This is achieved in a transitive fashion. The first trust relationship to consider is between the remote data provider and the PRP enclave. This relationship is established using the SGX remote attestation protocol. The sigma protocol identifies the PRP enclave and ensures that it has not been altered. It has already been analyzed in the previous chapter. After the sigma protocol finishes, the server trusts the PRP implementation, as well as the established key. Furthermore, the SGX hardware prevents modifications and information leaks while the enclave is running. As presented in the previous section, SGX-enabled processors provide some security guarantees against both internal and external attackers. The other trust relationships are established between the PRP enclave and the other parts of the TRM design. The PRP enclave has the responsibility to issue the policy deployment and to only release the data outside the protected enclave if it is protected by the usage control system. For this, the enclave verifies the integrity of the PIP and PEP external modules by comparing the PCRs of the external TPM to known "good" values. During the boot sequence, the TPM measures both the PEP and the PIP components. If an internal attacker modifies the PEP or PIP module in order to influence the TRM implementation, the measurements will change and the PCR verification fails. In that case, the PRP will not release any sensitive data outside the enclave. An internal attacker could try to launch the PRP enclave inside an untrusted system process instead of the PIP and retrieve the protected data after the policy deployment step. This is prevented by measuring each start of an enclave individually. The PIP launches each TRM enclave exactly once during startup, and launching any more instances will result in changed PCR values. Furthermore, the reference monitor of the operating system ensures that any non-root system user (i.e an internal attacker) cannot tamper with the modules after the system is booted and the measurement is performed. This presupposes that the operating system is properly configured and the client user

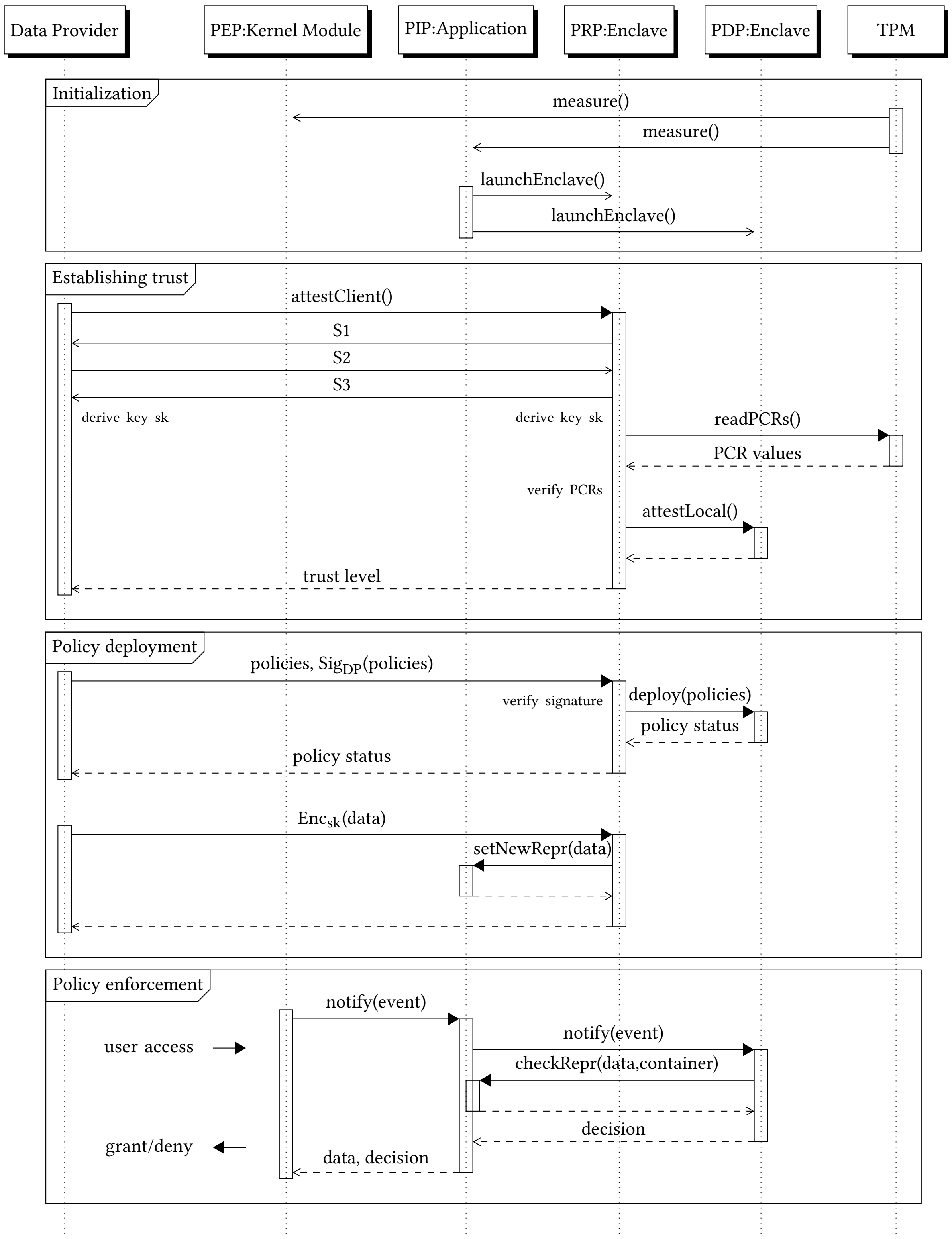

Figure 3: Sequence diagram of TRM protocol interaction.

does not have root rights at the system, which is a common demand for TPM protected systems (c.f. section 2). External attackers are not relevant for this trust relationship, since no messages between the TRM modules ever leave the client system. Finally, the trust relationship between the PRP and the PDP has to be established. Since the PDP is also an enclave, this is achieved by a SGX local attestation. The local attestation protocol verifies the integrity of the enclave code, which prevents internal attackers from tampering with the PDP implementation. After the enclave is launched, the SGX hardware isolates it from the rest of the system. Hence internal attackers cannot influence the PDP component during runtime. Again, external attackers are not relevant for this trust relationship, because no messages leave the client system.

4.3 Open Issues

The main problems of the proposed general TRM design arise from the fact that it includes a TPM, as opposed to the purely SGX-based solution. However, including a TPM is necessary in order to implement general distributed usage control. A powerful PEP, which can enforce usage control policies throughout the whole system, can only be implemented outside a SGX enclave. Hence the trust anchor of this design, unlike with many SGX-enabled applications, is not just the set of SGX enclaves alone, but also includes the TPM and the operating system. This ultimately leads to weaker security guarantees. Depending on the policies and the way of the data usage, hardware attacks may become a problem again. Only the SGX enclaves are protected against information leakage by hardware mechanisms, so data that exited the enclaves can be intercepted at the hardware bus. Furthermore, the TRM design requires additional assumptions that SGX-based designs do not necessarily demand. Unlike before, the operating system's reference monitor has to be trusted. If the operating system is vulnerable, an untrusted user could influence the user space TRM modules, for example the PEP or the PIP, during runtime. For the same reason, the client system user must not be root. Even though a root user cannot forge TPM measurements, he still can influence the non-enclave TRM modules after the measurement has been done. Those TPM-related assumptions are much stronger than those required with purely SGX-based designs, but they are common for usage control systems. Other unsolved questions concern the actual implementation of the proposed TRM design. It is not yet clear how the PCR values of an external TPM can be checked from inside an enclave. Intel does not provide an interface to access external TPMs from inside an enclave, because in the standard use cases, the SGX hardware replaces traditional TPMs. In our use case however, SGX cannot replace the external TPM, since we require to take measurements during the boot sequence. Of course it is possible to use an `OCALL` invocation in order to query the external PCR values from outside the enclave, but in that case the returned PCR values might be vulnerable to modification by an untrusted system process. Moreover, the TPM needs to be able to measure every SGX enclave launch separately, in such a way that each enclave launch changes the PCR values. Otherwise the user can easily impersonate a PIP, launch a second PRP enclave instance and retrieve the data himself. It is not yet clear how this mechanism can be realized in a SGX-enabled system that also features a TPM.

5 CONCLUSION

In this work we showed that it is possible to achieve secure computation as well as remote access control with SGX-enabled processors. However, enforcing distributed usage control on remote client systems requires a Policy Enforcement Point that can intercept events on an operating system level. This component cannot be implemented as an isolated SGX enclave. Therefore we proposed a TRM design with separate, but interdependent components that reside inside as well as outside of isolated enclaves. An additional TPM establishes the required trust outside of SGX enclaves. Before transmitting sensitive data to a client, the data provider uses the SGX remote attestation protocol to verify the integrity of the enclave software that is running on the client. The enclave then evaluates the measurements of the external TPM, thereby ensuring the integrity of the TRM parts that reside inside the operating system kernel. Ultimately, a transitive trust relationship is established that enforces deployed usage control policies on remote systems.

Necessary future work towards a full working implementation includes analyzing the size of various PRP and PDP code bases. Currently SGX enclaves are still quite limited in size, which could make an SGX implementation of advanced applications unfeasible. Furthermore, the ways of communication between the TPM and the SGX enclaves have to be researched further.

REFERENCES

[1] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, Vol. 13.

[2] Anne Anderson, Anthony Nadalin, B Parducci, D Engovatov, H Lockhart, M Kudo, P Humenn, S Godik, S Anderson, S Crocker, et al. 2003. extensible access control markup language (xacml) version 1.0. *OASIS* (2003).

[3] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. 2017. Software Grand Exposure: SGX Cache Attacks Are Practical. *arXiv preprint arXiv:1702.07521* (2017).

[4] Sanchuan Chen, Xiaokuan Zhang, Michael K Reiter, and Yinqian Zhang. 2017. Detecting privileged side-channel attacks in shielded execution with Déjá Vu. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 7–18.

[5] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016 (2016), 86.

[6] Trusted Computing Group. [n. d.]. TCG architecture overview. (TCG Specification). ([n. d.]).

[7] Intel. 2016. Intel®Software Guard Extensions Remote Attestation End-to-End Example. (2016). https://software.intel.com/en-us/articles/intel-software-guard-extensions-remote-attestation-end-to-end-example

[8] Jaehong Park and Ravi Sandhu. 2004. The UCON ABC usage control model. *ACM Transactions on Information and System Security (TISSEC)* 7, 1 (2004), 128–174.

[9] Alexander Pretschner, Manuel Hilty, and David Basin. 2006. Distributed usage control. *Commun. ACM* 49, 9 (2006), 39–44.

[10] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert Van Doorn. 2004. Design and Implementation of a TCG-based Integrity Measurement Architecture.. In *USENIX Security Symposium*, Vol. 13. 223–238.

[11] Ravi Sandhu and Xinwen Zhang. 2005. Peer-to-peer access control architecture using trusted computing technology. In *Proceedings of the tenth ACM symposium on Access control models and technologies*. ACM, 147–158.

[12] Paul Sevinç, Mario Strasser, and David Basin. 2007. Securing the distribution and storage of secrets with trusted platform modules. *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems* (2007), 53–66.

[13] Ming-Wei Shih, Sangho Lee, Taesoo Kim, and Marcus Peinado. 2017. T-SGX: Eradicating controlled-channel attacks against enclave programs. In *Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA*.

[14] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. 2015. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 640–656.