Flexible and Secure Enterprise Rights Management based on Trusted Virtual Domains

Yacine Gasmi, Ahmad-Reza Sadeghi, Patrick Stewin, Martin Unger, Marcel Winandy
Chair for System Security
Ruhr-University Bochum, Germany
{yacine.gasmi, ahmad.sadeghi, patrick.stewin, martin.unger, marcel.winandy}@trust.rub.de

Rani Husseiki, Christian Stüble
Sirrix AG security technologies, Germany
{r.husseiki, c.stueble}@sirrix.com

ABSTRACT

The requirements for secure document workflows in enterprises become increasingly sophisticated, with employees performing different tasks under different roles using the same proprietary platform. Particularly, fine-grained access control to document information is necessary in certain scenarios where the integrity and confidentiality of parts of documents is of high priority.

In this paper, we present a secure and flexible Enterprise Rights Management (ERM) system based on a refined version of the Trusted Virtual Domains (TVDs) security model that allows to establish isolated execution environments spanning over virtual entities across separate physical resources. Our security concept achieves a two-layered policy enforcement on documents: a *TVD Policy* ensuring isolation of the workflow from other tasks on the user platforms, and a role-based document-policy ensuring both confidentiality and integrity of document parts. Moreover, in contrast to existing solutions, our architecture offers advanced features for secure document workflows such as offline access to documents and transparent encryption of documents exchanged via USB, external storage or VPN communication between peer platforms. We also shed the light on key management, document structure and document policy enforcement mechanisms to support the ERM infrastructure. Finally, we prove our concept based on an implementation.

General Terms

Management, Design, Reliability, Security

Keywords

Trusted Virtual Domain (TVD), Enterprise Rights Management (ERM), Security Policy, Trusted Computing

1. INTRODUCTION

The consequences of unauthorized access to digital document content can be severe when a financial statement is modified, a confidential business document is disclosed to the public, or a design specification (e.g., in automotive industry) is leaked to a competitor. Therefore, financial institutions, governmental agencies and manufacturing companies all face the problem of controlling sensitive documents information when processed and exchanged among several parties within and beyond their boundaries. Consequently, security demands for digital document workflows have become increasingly sophisticated.

Different notions, sometimes interchangeably, have been already used to describe the general problem of secure document workflow in literature, namely secure information sharing [29], dissemination control [33, 23, 30], and persistent information control [21], in slightly different meanings, but always in the same context. Those were always differentiated from Digital Rights Management (DRM) for different technical and economical aspects, although overlapping exists in the core technology [29, 21]. The main reason is that DRM solutions focus on protecting a single file at a time, without considering the need for collaboration of different users with different rights over the document, which might entail a flow of the document across many platforms [14].

General security models addressing the problem of secure document management have also been discussed in [21] and [23]. From a technological perspective, one of the main problems is the establishment of trust in client platforms. Those are under control of the users who, due to various motivation factors, can attempt to break the control rules and gain unauthorized access to documents [21]. What paves the way for a user to attack a dissemination control system are the architectural security flaws in current operating systems which leave software-based controllers or monitors subject to alteration and circumvention attacks [25, 36, 30].

Many platform architectures have already been considered in order to strengthen the security of client platforms to ensure policy enforcement. *Trusted Computing* (TC) and *Virtualization* techniques have presented a big step forward in this direction [25, 26]. While virtualization can provide isolation of security critical components (e.g., reference monitors, ERM controllers, etc.) from potentially malicious software by executing them in separate virtual machines, TC provides a means for verifying the integrity of virtual machines and software components [29]. In this context, notions like "trusted viewer" [29] and "trusted reference monitor" [30] have been introduced. Moreover, the *Trusted Computing Group* (TCG)[1] has published several specifications

[1]The TCG is a consortium of a large number of IT enterprises, which proposes a new generation of computing platforms that employs both, supplemental hardware and soft-

on various concepts of trusted infrastructures (see, e.g., [35, 34]). In particular, it defines attestation protocols that allow a platform to provide evidence of its integrity, and therefore its trustworthiness with respect to well defined policies, to remote parties [24]. Such schemes have been used to report integrity measurements of virtual machines [16].

On the other hand, recent advances in IT and business security modelling has yielded to the concept of Trusted Virtual Domain (TVD) [8, 18] which leverages the combination of Trusted Computing and virtualization techniques in order to provide confinement boundaries for an isolated execution environment – a domain – hosted by several physical platforms. The TVD concept is a potential basis for a new approach for secure information sharing, since it achieves isolation between domains even when hosted on the same physical platform, and establishes trust between virtualized entities belonging to one domain even if hosted by independent physical platforms. Furthermore, it allows enforcement of a domain-wide policy mandating trustworthy security configurations. The idea of applying the TVD concept for secure information sharing has been addressed in [18].

In this paper, we address the particular problem of organization-scale secure information sharing, which is widely known under *Enterprise Rights Management* (ERM) [36, 2, 32], based on the TVD concept. ERM is mainly concerned with fine-grained access and usage of document content (read, print, copy, modify, transfer) by users with different rights. A ERM controller is responsible for such a document policy enforcement. It also allows protection of documents even when exchanged directly between peers. However, although many ERM solutions exist in the market [3, 1, 22], we believe that they are unable to ensure enforcement of document policies when deployed on commodity OS since the trustworthiness of platforms can not be guaranteed in this case. The ERM controller, being a separate client program or an integrated plug-in in native applications, is subject to manipulation or simply circumvention, let alone the attacks on decrypted usage policies, keys and document content in memory. The advantage of a TVD-based infrastructure is that it can help to achieve a distributed and isolated domain with trustworthy ERM characteristics. This allows information flow control of ERM documents and keys on hardware and OS level, as well as fine-grained control of access to document content on application level.

1.1 Contribution

In this paper, we extend the TVD approach for secure information sharing to account for fine-grained information flow control on document level, based on the ERM concept. To achieve this, we present a new ERM security concept that we realize by establishing an *ERM-TVD* that integrates ERM functionalities in a TVD infrastructure. Within this TVD, users can securely exchange documents using their software in a peer-to-peer scheme, with the assurance that document content will be protected from leakage, unauthorized modification or misuse. The ERM-based granular information flow control restricts the access to information within the ERM-TVD according to the user's role in the corresponding document workflow. Hence, users with different roles will have *unequal* access to shared information within the ERM-TVD. Moreover, this assurance is maintained even if platforms are not online, i.e., connected to a central server, therefore allowing offline and distributed access to documents which enhance the fluidity of the document workflow. In addition, mutual trust is inherently established between platforms that join the ERM-TVD without any need for further peer-to-peer protocols. Further on, the proposed ERM architecture evinces additional advantages:

- *Security*: The underlying TVD concept isolates workflows and enforces TVD-related security policies.
- *Compatibility/Interoperability*: Our concept allows to reuse existing application software (e.g., OpenOffice) that runs in a virtual machine together with an existing ERM controller, without relying on the controller's security, since the virtual machine is unable to violate the *TVD Policy* enforced transparently. For security reasons, we propose to use an ERM controller isolated from the application software.
- *Modularity*: Since a separate component, that cannot violate the ERM-TVD policy, enforces the document-level security policy, our architecture can easily be adapted to new document and policy formats.

1.2 Motivation Scenario

We consider the following typical scenario, and derive the requirements we need to fulfill in our architecture:

Within an enterprise, employees perform different tasks under different roles, for example accessing the internet, using intranet services, editing unclassified documents, and editing classified documents, such as patents. In the following we will focus on patents as our running example. Each of these tasks has different security requirements. In security-critical environments such as government and military institutes, classified documents are isolated by using physically separated computing platforms. However, in typical enterprise environments users perform these tasks using one computing platform providing a questionable isolation between them. In such an environment, users could intentionally or unintentionally bypass security policies by sending documents unencrypted over an insecure network connection, or by exchanging files using an USB stick.

Especially, the workflow of creating patents requires ERM features since patent documents are accessed by different users with different fine-grained access rights over the documents. The patent department needs to enhance the patent content security by increasing restrictions on document access and usage by different roles in the workflow. While the inventor should be able to read all parts of the document, an illustrator should only be able to insert figures, a literature reviewer should only be able to read and not modify, and an attorney should be able to modify only parts of the document. Moreover, the patent contributors would like to exchange the documents in an ad hoc manner, always keeping them protected from disclosure to other users, and from unauthorized usage by other contributors. For example, they would like to use USB sticks to exchange the patent document without worrying about the USB stick falling into the wrong hands, or they would like to send the document using a protected network connection. It should also be possible for them to access those documents offline, as they might need to securely work on them in a train or at home.

Therefore, enterprise workflow documents entail the fol-

ware. The claimed goal of this architecture is to improve the security and the trustworthiness of computing platforms (see, e.g., [35, 34]).

lowing high-level requirements: Firstly, documents accessed on computing platforms need strict isolation and protection from other unauthorized processes or users of the platforms. Secondly, contributors to the workflow may have different access rights on different parts of a document according to a document policy in order to protect both the confidentiality and integrity of documents. Thirdly, the fluidity of the workflow should be ensured by allowing exchange of documents by regular means, in addition to distributed and offline access, without violation of the security requirements.

1.3 Background and Definitions

Our approach is based on the TVD concepts defined in [6, 8, 9, 15]. Compared to this literature, our ERM-TVD is a refinement of the proposed TVD concepts. Furthermore, we also introduce related terms:

- We consider the *Trusted Computing Base* (TCB) as the set of all security crucial hard- and software components of a local computing platform responsible for preserving the local platform's trustworthiness. The security kernel[2] contains all software components of the TCB.
- A *compartment* is a local subject, e.g., a virtual machine (VM) or a native process identified by a local compartment ID. Compartments are isolated by the underlying security kernel. *Compartment configurations* uniquely identify the type of compartments.[3] The configuration is measured by the security kernel.
- A *domain* is a set of compartments identified by a domain ID. Domains are local types used by the security kernel to enforce a security policy, but they are, in contrast to TVDs, not visible to remote entities. Within a domain, compartments can communicate freely, but compartments running in different domains are strictly isolated from each other. Therefore, compartments can only communicate if they are members of at least one identical domain.
- A *Trusted Virtual Domain* (TVD) is defined as a coalition of compartments that can trust each other based on a security policy that is uniformly enforced independently of the boundaries of physical computing resources. It achieves an isolated execution environment where mandatory security policies are enforced. The trust between compartments is established based on platform integrity measurements with secure communication channels bridging those entities. Admission control to the TVD (i.e., the decision whether an entity running on top of a specific physical platform is allowed to join the TVD) is enforced based on a *TVD Policy*. Therefore, a special node in the TVD called *TVD Master*, e.g., implemented as a central server, controls the access to the TVD by following the admission control rules specified in the *TVD policy*.
- *Trusted Computing* (TC) as proposed by the *Trusted Computing Group* (TCG) is based on hardware and software extensions to computing platforms providing security and cryptographic functionalities. The core component is the *Trusted Platform Module* (TPM, cf. [35]) that also provides a set of registers for protected storage called *Platform Configuration Registers* (PCR) that store integrity measurements[4]. TPMs can be used to prove that a platform has booted a valid operation system with a trustworthy configuration[5].
- *Trusted Channels* are encrypted and authenticated connections established between peers where integrity information (measurement) is cryptographically linked to those peers (i.e., secure channel endpoints) to attest them, by enabling secure provisioning of configuration evidence of the peer platforms.

[2]E.g., [13] or [11] presented a security framework based on a security kernel.

[3]In general the configuration should express the input/output behavior of the underlying code. However, we take the approach proposed by the TCG.

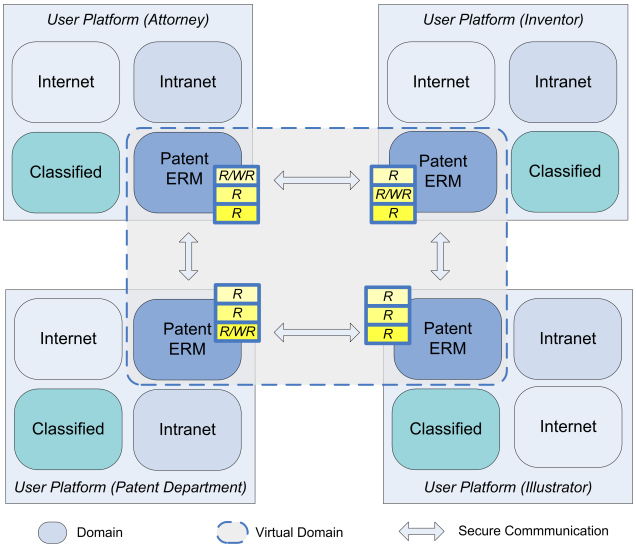

Figure 1: Different Tasks typically performed in Enterprise Environments on one Computing Platform

2. GENERAL IDEA

In this paper, we provide fine-grained dissemination of information within a single workflow domain by integrating ERM features into the IT infrastructure. For that, we propose a two-layered security policy enforcement in order to ensure document protection. The TVD concept is used to isolate workflows as well as tasks with different security requirements, e.g., internet access, classified data access, and patent creation (cf. Figure 1). Within a TVD, a trusted compartment on each of the contributing platforms called *ERM controller* enforces the document level security policy which provides fine-grained dissemination of information.

Our model allows establishment of a TVD in a desktop environment. A security kernel supporting virtualization ensures that users can use one or more computing platform to work within different TVDs simultaneously, since compartments of each TVD are strongly isolated from compartments of other TVDs. Moreover, the communication of compartments of the same TVD is encrypted using cryptographic keys that are only available to security kernels that fulfill the requirements of the *TVD Policy*. Therefore, both the computing platforms and the network infrastructure of an

[4]Cryptographic SHA-1 based hash values of binaries

[5]The term *configuration* follows the terminology of the TCG and means the integrity state of a platform or software component, e.g., taken during an integrity measurement and being represented as hash value of a program binary.

enterprise are virtually separated into different TVDs that may come with different security policies. On each platform, compartments belonging to the same TVD are executed within the same *virtual domain* allowing a free communication between them. However, an information flow between compartments of different domains is only allowed if it does not violate the *TVD Policy* of both domains.

While the isolation aspect of TVDs achieves the confidentiality requirement of classified data, patent documents need a specific security concept supporting ERM features, such as encrypted storage and document-level policy enforcement.

TVDs that are dedicated for document workflows and enforcement of document-level security policies are called ERM-TVDs. ERM-TVDs include compartments running a document policy enforcer (*ERM controller*). Moreover, our security model benefits from the isolation capability of the underlying security kernel to restrict the access of the ERM application to network and storage resources through the trusted *ERM Controller*. The same applies to the *ERM Controller* itself, which can only access virtualized resources through trusted encryption modules which act as interfaces to encryption services.

One important advantage of this concept is that it allows to use existing operating systems and applications as ERM compartments, e.g., OpenOffice or MS Word, without relying on their security: If defined by the ERM-TVD policy, the underlying security kernel encrypts all persistent storage (hard disk, USB) and network traffic (VPN) using a TVD-specific cryptographic key. This approach also allows offline access to documents since the ERM-TVD policy is enforced based on the configuration of the security kernel which guarantees trustworthiness of the executed compartments.

Since the *ERM Controller* is part of the ERM-TVD, a violation of the document policy, e.g., due to a bug in the *ERM Controller*, cannot violate the *TVD Policy*. Therefore, the *ERM Controller* can either be realized as a dedicated compartment running separately from the ERM application used to edit the document, or it can be an existing application running in the same compartment or a plugin to the document rendering engine. Therefore, existing *ERM Controllers* can also be used depending on their compatibility with the required document security policy. In our model, we consider fine-grained confidentiality and integrity requirements on parts of the document, which entails a dedicated *ERM Controller* with a specific document policy structure.

3. ERM ARCHITECTURE

3.1 Security Model

This section describes the security model of our approach and defines related terms. The security model can be divided into two parts, the *network security model* describing entities and their security properties in a distributed environment, and the *platform security model* implemented by a security kernel and used to realize the network security model.

3.1.1 Platform Security Model

The platform security model which is similar to sHype [27], describes the security model enforced on each physical platform by an abstract security kernel.

A compartment is started by the `startCompartment()` function. Compartments can query the configuration of other compartments using `getConfiguration()`. To prove the configuration to external entities, compartments can use the function `getCertificate()` to receive a certificate including the configuration of the security kernel and the compartment itself. Compartments can be member of different domains, as depicted in Figure 2. The domain membership is defined when a compartment is started using the `startCompartment()` function. However, a compartment can only start another compartment within a domain if the starting compartment itself is already member of that domain. To create new domains, the function `createDomain()` is provided, while the creating compartment of a new domain automatically becomes its first member.

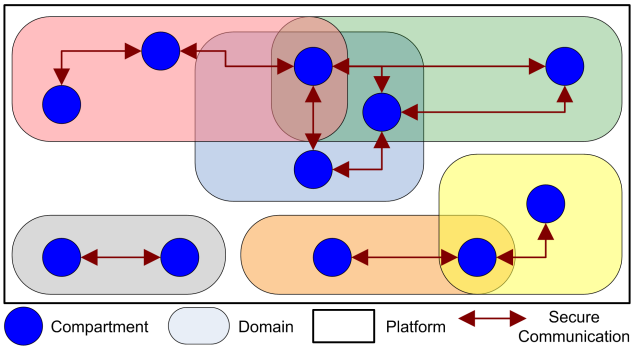

Figure 2: Domain Model enforced by Abstract Security Kernel

Figure 2 depicts how a security kernel enforces a policy based on domains: Only compartments that are members of the same domain can communicate with each other.

A *user* identified by a user ID is an external entity that can interact with compartments using a *session*. After successfull user authentication, the user ID is assigned to the corresponding session which starts the first user compartment using `startCompartment()`. A started compartment inherits this user ID that can be queried using `getUser()`. Moreover, users only have the right to communicate with certain domains defined by an assigned domain list.

Compartment security attributes to be managed by the security kernel are the compartment configuration, the list of domains, a unique compartment name, and a user ID. Domain security attributes are a unique name and the domain's admission control policy. User security attributes are the user name and a list of domains.

3.1.2 Network Security Model

The network security model is an instantiation and refinement of the TVD concept (cf. Section 1.3). In this model, the TVD is a protection boundary around a set of active nodes (compartments or physical machines), a trusted TVD management compartment called *TVD Master*, and a *TVD Policy* describing the security properties of that TVD. In our model, a TVD includes a set of domains of different platforms including their compartments (cf. Figure 3).

Compartments of the same TVD can communicate freely even if they are running on top of different physical machines. However, the *TVD Policy* defines security requirements regarding communication between compartments and persistent storage of compartments, to be enforced by the

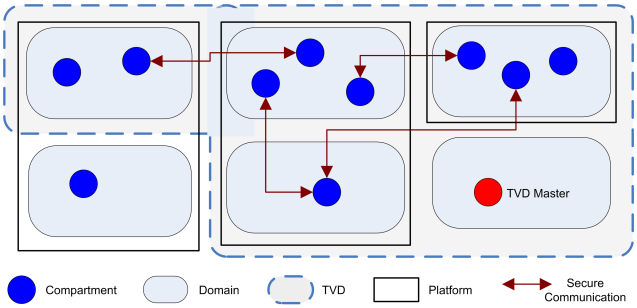

Figure 3: Network Security Model based on Trusted Virtual Domain Concept

systems hosting the domains. Moreover, a TVD supports admission control (i.e., the decision whether a compartment running on a specific physical platform is allowed to join the TVD) based on the compartment's configuration. The admission control policy is part of the *TVD Policy* and enforced by the *TVD Master*. The `joinTVD()` protocol performed with the *TVD Master* allows a node to become a new member of a TVD, while the `leaveTVD()` protocol removes a node from the TVD. In Section 3.3, we decribe in detail how a platform joins a TVD.

3.2 ERM Security Concept

In this section, we define an ERM security concept based on the security model discussed in the previous section, a trust model encompassing the components of the system, and security goals regarding document protection, namely the confidentiality and integrity of the document parts.

Since confidentiality of documents is a priority goal, our security concept leverages the strict domain isolation capability of the security kernel to achieve this goal based on the security model discussed in Section 3.1. As far as confidentiality is concerned, we assume that the requirement for document protection is that of enforcing encryption of the document content when stored in memory or communicated via network to other contributors to the workflow. For that, we present the security concept in Figure 4.

As shown in Figure 4, the *ERM Application Domain* includes ERM-related compartments, namely *Windows* and *Linux based ERM applications* in addition to an *ERM Controller* (a document policy enforcer discussed in Section 3.4). The compartments can freely communicate with each other. Another domain, the *Controller Domain* includes the *TVD Proxy* (cf. Section 1.3), a *Network Encryption* module acting as an interface to the network, a *Disk Encryption* module acting as an interface to hardware disks, a *File Encryption* module acting as an interface to external storage such as USB disks, and the *ERM Controller*. A third domain, the *Security Service Domain*, includes the *TVD Proxy*, the *Network Encryption*, the *Disk Encryption*, and the *File Encryption* modules.

An ERM-TVD spans over all the mentioned local compartments except for the ERM applications, and includes, in addition, the *TVD Master* which is running remotely.

This domain-based distribution of components achieves a topology that can be summarized as follows:

1. ERM applications can only communicate with the *ERM Controller* to access ERM functionalities.

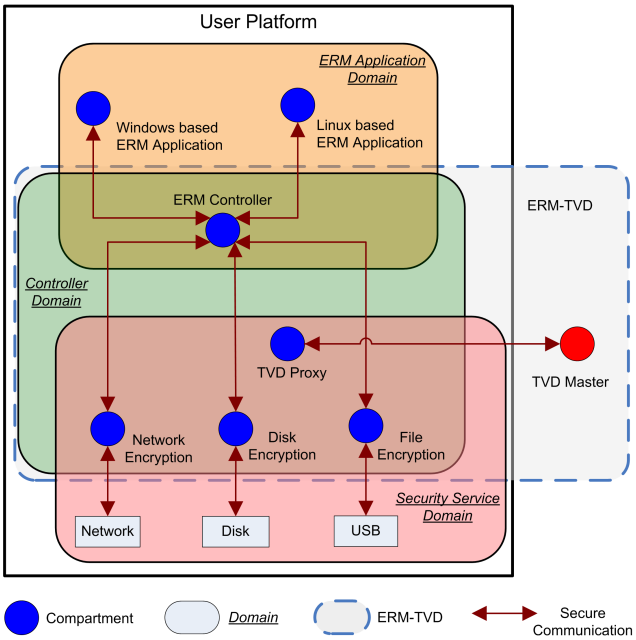

Figure 4: ERM Security Concept

2. *ERM Controller* can only communicate with the *Network Encryption*, *Disk Encryption*, and *File Encryption* modules to access network and storage resources.
3. *Network Encryption*, *Disk Encryption*, *File Encryption* and *TVD Proxy* are the only components that can access hardware resources.

Since the *ERM Controller* is trusted to join the ERM-TVD, the document policy is reliably enforced regardless of the trustworthiness of the ERM applications. The same applies to the encryption modules that reliably enforce network and storage encryption. The *TVD Proxy* is responsible for verifying the configuration of the mentioned components according to the ERM-TVD policy.

3.3 ERM-TVD Establishment

A sequence of steps is needed to establish the domains and the ERM-TVD on a platform for the user to access documents. The *TVD Proxy*, a local instance of *TVD Master*, is assumed to be already started in the *Security Service Domain*. It is meant to enforce the *TVD Policy*. The *TVD Policy* is retrieved by the security kernel of the platform via a Trusted Channel to the *TVD Master* (cf. Section 1.3). The channel is bound[6] to the TCB configuration, which guarantees its integrity and confidentiality. The *TVD Proxy* is usually started once the *TVD Policy* is retrieved from the *TVD Master*.

The establishment is devided into two phases (cf. Figure 5). In the following, we list the steps needed by a platform to establish the necessary domains and to join the ERM-TVD:

1. Join ERM-TVD: By using the function `joinTVD()`, the user triggers to join the platform to the ERM-TVD. In

[6]Binding is a TPM function that cryptographically relates/binds data to the platform's configurations reflected by a subset of PCR values of the TPM. Unbinding (i.e., decryption of the bound data) is only possible if the current PCR values match those stored with the bound data.

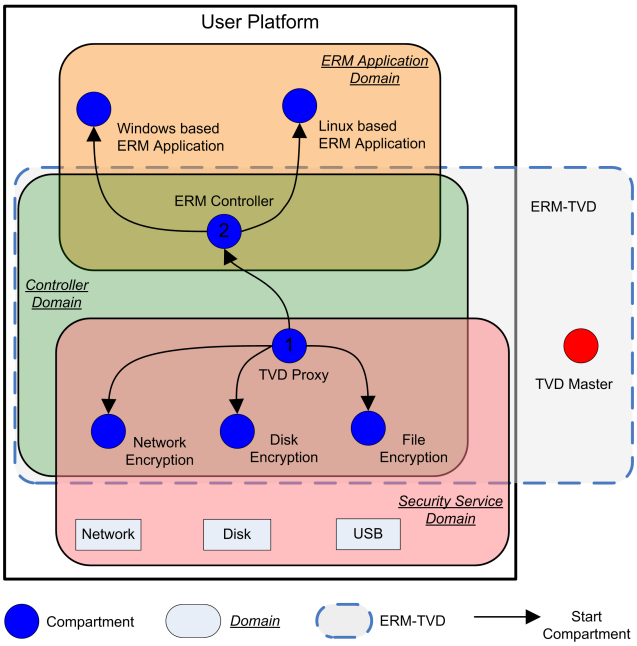

Figure 5: ERM-TVD Establishment Steps

this phase, the *TVD Proxy* starts all necessary compartments of the ERM-TVD and configures the necessary services. Specifically, it starts the *Network Encryption*, *Disk Encryption* and *File Encryption* modules in the *Security Service Domain*, and the *ERM Controller* compartment in a new domain, the *Controller Domain*. The *TVD Proxy* has to start the encryption modules. This ensures that the *TVD Proxy* can verify their trustworthiness[7] using measurement capabilities offered by the security kernel. If the encryption modules are compliant to the *TVD Policy*, this ensures that encryption is enforced whenever a document is stored or sent outside of the *Controller Domain*. In addition, the *TVD Proxy* configures the network and storage encryption components with the *TVD Key* (included in the *TVD Policy*). At this point, the *ERM Controller* can access network and storage resources through the encryption modules.

2. Start ERM application: Using the function `startERM()`, the user can start the ERM application. In this phase, the *ERM Controller* starts a new domain – the *ERM Application Domain* – with the compartment running the ERM application as domain member. This compartment does not have to be trusted and depends on the default application for ERM-protected documents that the user has already chosen. At this stage, the user can access documents using the ERM application.

3.4 Document Policy Enforcement

In order to attain a higher level of integrity and confidentiality of document content, a second layer of policy enforcement is necessary on document level. As stated in Section 1.2, contributors to a workflow have different access rights on different parts of the document depending on a document policy. Hence, we introduce a document structure and an associated document policy that account for this requirement. Then, we define a protocol to enforce the document policy.

[7]The *TVD Proxy* is responsible for *TVD Policy* enforcement.

3.4.1 Document and Policy Structures

In order to provide confidentiality and integrity of document content based on rights of workflow contributors on different parts of the document, we define a structure for documents that is based on document types (dt) (e.g., patent, article, ermpaper, ...) and document parts (dp), with each document part having its own type (dpt) (e.g., metadata, text, figures, ...) and actual content (dpcontent). A document $\mathcal{D}$ is defined as follows:

$\mathcal{D}$:= {ID, dt, DocParts},
dt ∈ DocTypes := $\{patent, article, ermpaper, \ldots\}$,
DocParts := $\{dp_1, \ldots, dp_n\}$,
dp := {dpt, dpcontent},
dpt ∈ DocPartTypes := $\{metadata, text, figures, \ldots\}$
dpcontent := $\{\ bd \mid bd\ is\ binary\ data\ \}$

The document policy (DocPolicy) associates a certain document to a set of rules (DocRules). Those rules define access rights (DocPartAccessRights) for a certain role (role) on a certain part of the document according to the document type (dt). For simplicity, we account only for *no access*, *read*, and *write* rights (cf. AccessRights). The document and its policy are linked by an identifier (ID). The document policy can be defined as follows:

DocPolicy := {ID, DocRules},
DocRules := {dt, DocPartAccessRights},
DocPartAccessRights := {dpt, role, dpar},
role ∈ Roles := $\{inventor, illustrator, attorney, \ldots\}$
dpar ⊂ AccessRights := $\{read, write, noaccess\}$

3.4.2 Enforcement Protocol

The *ERM Controller* has the role of the document policy enforcer. In the following, we describe how the document policy is enforced after the ERM-TVD has been established, and the keys have been handed to the corresponding components. For simplicity, we assume that the document policy is attached to the document. Figure 6 shows the interaction between the *ERM Controller* (ERM Controller), the ERM application compartment (ERM Application), the *File Encryption* module (File Enryption) and a *File Provider* (File Provider) (local memory, USB memory, Fileserver, other platform, ...) in order to enforce the document policy. The document received by ERM Controller is decrypted by the *Security Services Domain*. Therefore, document content, metadata and policy are accessible by ERM Controller. We also assume that the user has been already authenticated, and its role is available to ERM Controller. For simplicity, we consider an asymmetric key for the ERM-TVD, where all members share the same public key (PK_{ERMKey}) and secret key (SK_{ERMKey}).

ERM Controller is triggered when a user attempts to load a document from File Provider. Afterwards, the following steps take place:

1. ERM Controller asks File Provider for the document that returns it in decrypted form.
2. File Enryption requests the document from File Provider and decrypts it using SK_{ERMKey}.
3. ERM Controller receives the decrypted document from File Enryption and renders the filtered document content

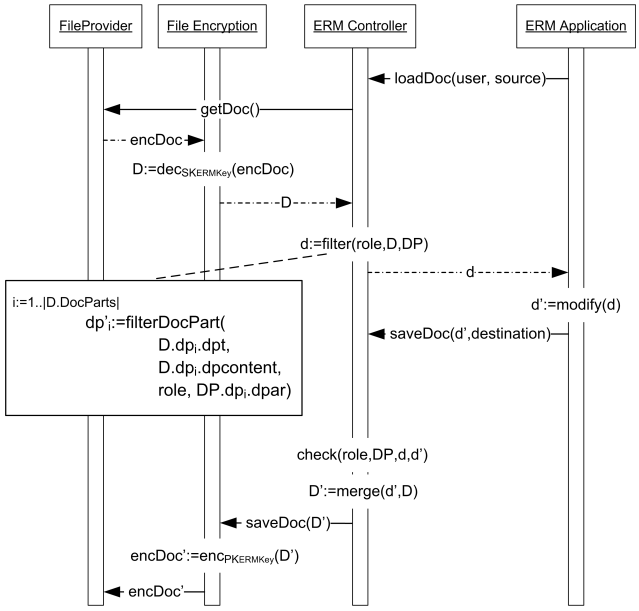

Figure 6: DocPolicy **Enforcement, FileProvider (e.g., USB memory, HD, Fileserver, etc.)**

(by parts) for ERM Application according to the rights of the user stated in the document policy.

4. If the user modifies to the document and attempts to save it, the command and the modified content are intercepted by ERM Controller which verifies if the active role is granted the corresponding right according to the document policy. If so, ERM Controller merges the modifications with the original document content and invokes File Enryption to store the document (e.g., on same File Provider).
5. File Enryption encrypts the document with PK_{ERMKey} and stores it on the chosen destination.

3.5 Key Management Notes

To enable ERM-TVD members to encrypt and decrypt documents, encryption services have to be configured with adequate keys. In our proof of concept, we use a symmetric shared key, i.e., shared by all TVD members (cf. Section 4). However, the following can be noted regarding key distribution and usage:

1. Since access to the keys depends on the *TVD Policy* enforcement on a user platform, the keys represent a cornerstone of ERM-TVD membership. This means, that once a platform has the keys configured for the corresponding security service, it would be able to access documents within the ERM-TVD as long as the keys are valid. Consequently, a mandatory requirement for revocation of an ERM-TVD membership is the revocation of the corresponding keys. While broadcast encryption schemes can be useful to encrypt documents distributed by a central server, they can not account for peer-to-peer exchange of documents due to their unidirectional aspect: users need to decrypt a document, edit it, and encrypt it to be broadcasted again, which needs a new round of broadcast encryption every time a document is sent out by a user. Public key broadcast encryption schemes (see, e.g., [10]) allow users to encrypt using one public key (for the whole ERM-TVD), while decryption is done using his own set of keys. However, revocation in such schemes after a user has been granted access need further investigation, since revoked user platforms should be unable to access already encrypted documents.
2. Using shared session keys for document encryption is not a practical approach, since encrypted documents are not only valid for a short time. Hence, changing the session key would require re-encryption of each encrypted document stored on local or external memory. Session keys can however be useful for peer-to-peer transfer of documents over the network.
3. An adequate keying scheme for an ERM-TVD infrastructure would allow encryption of documents with shared keys, but keeps the possibility for a central authority to revoke particlular members. Group oriented cryptography provide such functionlitities. This would allow the *TVD Master* to revoke the membership of ERM-TVD members based on a regular verification of their enforcement of the *TVD Policy*. Encrypted documents would then be inaccessible by the revoked members.
4. The "Lazy revocation" scheme proposed by Backes et al. [4] is advantageous for Key Management in the ERM-TVD since only newly modified documents need to be re-encrypted. This means that encrypted documents spread around all TVD members do not need to be re-encrypted every time that one member is revoked. However, this scheme assumes a central repository for documents. Therefore, applying it on a distributed architecture entails additional requirements, e.g., enforcing re-encryption of modified documents with a fresh key - a task performed by the central server in the original scheme.

4. IMPLEMENTATION

We have implemented a prototype of our design on an existing security kernel (Turaya) [11]. The Turaya security kernel comprises two layers: a hypervisor layer based on an L4 microkernel [19] and resource management services (memory management, I/O drivers), and the trusted software layer providing the security services to achieve our security model.

The L4 microkernel provides isolation of processes and controls inter-process communication (IPC). Compartments can be native L4 tasks or para-virtualized Linux instances (L4Linux)[8]. Communication between compartments can be allowed or denied by defining access rights to their IPC interfaces. The microkernel enforces this IPC access control.

The main services of the trusted software layer used for our model are the following:

- *CompartmentManager* (CM) is the service being responsible to start or terminate compartments. It is a native L4 task and defines the IPC access rights for started compartments. Upon system start up, CM is equipped with a configuration file (implementing the domain policies) that defines hash values of compartment images (e.g., binary program files or VM images) and maps

[8]In principle, implementations based on Windows compartments would also be possible, but is currently not supported natively by L4.

them to compartment and domain names. When a compartment is started, CM measures the compartment image, checks the policy, and assigns the compartment to the requested domain if allowed and sets up the IPC access rights. Hence, we can realize the platform security model by defining the IPC access rights to the compartments accordingly.

- *TrustManager* (TM) is a minimally configured L4Linux compartment which contains a TPM driver and an application providing an interface to the TPM, e.g., used to generate a certificate for the Trusted Channel during the TVD deployment phase. This service is needed by other services to make use of the TC functionality provided by the TPM.
- *StorageManager* (SM) is a native L4 process which encrypts data received from other compartments and binds the data to the integrity measurements of the TCB, using the binding function of the TPM via TM.
- *VPN Module* (VPNM) is a minimally configured L4Linux compartment that handles the network connections and in particular VPN tunneling between domains on different platforms. It provides virtual switch functionality (similar to the *vSwitch* in [9]) to isolate the network traffic of different domains on the same platform. Currently, our prototype implementation supports only two network domains, one with and one without a VPN connection. The latter is used to realize our ERM scenario where the *ERM Controller* domain can connect to another (physical) node in the ERM-TVD (a file server in our case). The virtual switch is configured with a shared ERM-Key to encrypt network traffic. As mentioned in Section 3.5, the use of asymmetric keys should also be possible depending on the chosen keying scheme for the ERM-TVD.

The combination of CM and the L4 microkernel on the one hand, and VPNM, SM, and TM on the other hand, provide effective means to realize our platform security model of domains of compartments: Compartments can be grouped to domains by CM, and the communication between compartments is either prohibited or allowed (via IPC access control and storage/network control) according to the policy. Currently, our prototype implementation includes only hard-coded domains. Future versions will include dynamic configuration and setup of arbitrary domains.

5. SECURITY CONSIDERATIONS

In this section, we shortly describe how our ERM architecture fulfills the fundamental security requirements of an ERM system according to an "Attack Model". For that, we consider attacks on the three layers of the architecture: the ERM Application Domain, the Controller Domain, and the Security Services Domain.

1. Interception of document content on application layer (e.g. installation of malicious programs to intercept decrypted output by screen shots): The trusted *ERM controller* starts only the ERM applications in their domain. If a malicious program is installed on the platform, the CM would not recognize it as member of the ERM application domain. Therefore, communication between the malicious program and the ERM applications would fail.
2. Alteration of *Controller Domain* components (e.g. modifying the filtering funtionality of the *ERM controller* to obtain full access rights on document content, modifying the *Network/Storage encryption* modules): The TVD admission control mandates enforcement of the *TVD policy*. The components which are members of the ERM-TVD should comply to pre-defined configurations stated in the *TVD policy*. Hence, if a component is modified, the CM will obtain unmatching hash values for it and would not set its membership in the ERM-TVD. Consequently, JoinTVD() would fail (cf. 3.3).
3. Alteration of Security Services Domain components (e.g. modifying TSL components, etc...): In the initialization phase preceeding the ERM-TVD establishment, the *TVD proxy* obtains the *TVD policy* from the *TVD Master* based on a certificate generated by the TM and containing hash values derived during boot-up and TPM-secured (cf. 3.3). Hence, if for example any of the TSL components (i.e. CM, SM, TM and VPNM) is modified, its configuration would not conform with the values stored in the TPM, which would make unbinding the *TVD policy* impossible. Therefore, the ERM-TVD establishment on the system would fail.

6. RELATED WORK

In [29], Sandhu et al. discussed the use of TC functionalities to enable a "trusted viewer" for secure information sharing. For that, they proposed a framework that divides their approach into three layers: policy model (top), enforcement model (middle) and implementation model (lowest). This work is orthogonal to ours since policy enforcement is based on the assumption that the trusted viewer has a "suitable degree of assurance", whereas in our work, one of the goals is to ensure protection of the *ERM Controller*. In [23], several application-level security architectures were proposed to track and control digital information dissemination. This work is complementary to ours, since the proposed security architectures can be integrated on top of our proposed infrastructure. In [33], dissemination control was decomposed into three levels of enforcement strength depending on the digital content type and value of the information. In [26], TC support for DRM licenses enforcement has been presented based on the PERSEUS architecture, where the integrity of the DRM controller is verified during the boot up process. The aim of this work is the prominent DRM commercial distribution model which does not account for the need for P2P distribution of the files. Matson et al. provide in [21] a solution for non-commercial distribution for digital files to achieve a so called "persistent information security", e.g., for health records. Local information of document usage policy is based on a "Rights Client" and a "Trust Plugin" which trustworthiness is assumed without TC support.

Our security model is in some aspects similar to the *sHype* security model. *sHype* [27] is a security architecture which describes how to add isolation enforcement and policy-defined resource control to a hypervisor[9], resulting in a hypervisor reference monitor that can enforce mandatory access control (MAC) policies on inter-VM communication. Instead of completely isolated entities, VMs can form a *coalition*. The hypervisor enforces isolation between coalitions, but allows sharing of resources, e.g., virtual network connec-

[9]Software layer that mediates the access of virtual machines (VMs) to resources

tions, within a coalition as defined by a MAC policy.

The Chinese Wall Policy [7] defines data sets and conflict of interest sets for objects. It allows a subject the freedom to access anything at first. However, when a choice is made, restrictions derived for that choice are enforced, i.e., a *Chinese Wall* is created around the data set, preventing any conflict of interest. In our model a compartment can only communicate with other compartments with which it is member in the same domain domain. This is similar to the data sets.

Similar to our domain model, the *zones* of the Solaris OS [12] partition a single system into labeled zones. Subjects and objects are always associated with a zone, and information flow rules determine the communication between zones according to their labels, i.e., mounting a filesystem of another zone. Files are automatically labeled when mounted. This is similar to our domain membership when a compartment is started, but our model is more general and not restricted to files alone.

P2P trust establishment based on TC has been discussed in [20, 31, 30]. In all mentioned models, attestation protocols are required between peers in order to establish mutual trust, as opposed to our architecture which achieves inherent trust establishment between peers even when not connected to a network. Instead, we achieved propagation of trust from a ERM central server to client platforms based on TC binding functionality.

Some of the available ERM solutions in the market allow P2P exchange of encrypted documents [3, 1, 22]. However, those have some deficiencies when they are executed on commodity platforms due to the inherent security problems of mainstream operating systems.

The proposed ERM architecture is based on the TVD concept defined by Griffin et al. in [15], Bussani et al. in [8] and extended by Cabuk et al. in [9] to establish a secure virtual network infrastructure for secure communication among TVD components. In the same scope, further work based on the TVD concept is done by Berger et al. [6] to implement a Trusted Virtual Data center (TVDc) that is intended to be used for hosting Virtual Machines (VMs) belonging to different customers on the same hardware platform. The implementation uses TVDs supported by TC technologies represented by an Integrity Measurement Architecture (IMA) [17, 28], a virtual Trusted Platform Module (vTPM) [5], and a security Hypervisor (sHype) [27]. Our architecture also benefits from the work done by Gasmi et al. [13] to establish Trusted Channels for transmitting security critical data among trusted components of the TVD architecture using Transport Layer Security (TLS) and TC technologies.

7. CONCLUSION AND FUTURE WORK

We presented a new security concept for solving fundamental ERM security problems based on an extension of the Trusted Virtual Domain concept. We provided a design for an ERM infrastructure that enables secure, distributed and collaboration-friendly work on digital documents by enforcing a two-layered policy enforcement: a TVD-level policy ensuring confidentiality of documents based on virtualization and isolation techniques, and a document-level policy providing confidentiality and integrity of document parts according to the role of a contributor within the document workflow. Our architecture also leverages TC technology to ensure trustworthiness of components and binding policies as well as keys to platform configurations. Hence, the infrastructure allows exchange of documents in an ad hoc manner only between trustworthy platforms: encryption of documents is enforced when exchanged via network or storage which allows flexibility in the document workflow. We also proved our concept based on an implementation providing basic functionalities for access control to documents on a TVD-level. We shed the light on possible key management schemes for accessing documents within the TVD, in addition to potential document policy structures. However, those aspects need further investigation.

8. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their helpful suggestions and their very valuable comments.

9. REFERENCES

[1] Adobe Systems Inc. Adobe LiveCycle Policy Server: Document-level persistent protection and dynamic control for multiformat enterprise rights management. `http://www.adobe.com/de/products/server/policy/pdfs/ps\_datasheet.pdf`, 2006.

[2] A. Arnab and A. Hutchison. Requirement analysis of enterprise DRM systems. In Information Security South Africa, 2005.

[3] Authentica Inc. Page Recall: The Key to Document Protection. `http://www.adobe.com/de/products/server/policy/pdfs/ps\_datasheet.pdf`, 2002.

[4] M. Backes, C. Cachin, and A. Oprea. Lazy Revocation in Cryptographic File Systems. In *SISW '05: Proceedings of the Third IEEE International Security in Storage Workshop*, pages 1–11, Washington, DC, USA, 2005. IEEE Computer Society.

[5] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: virtualizing the trusted platform module. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2006. USENIX Association.

[6] S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan. TVDc: Managing Security in the Trusted Virtual Datacenter, 2008. ACM SIGOPS Operating Systems Review, Vol 42, Issue 1.

[7] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 206–214. IEEE, 1989.

[8] A. Bussani, J. L. Griffin, B. Jasen, K. Julisch, G. Karjoth, H. Maruyama, M. Nakamura, R. Perez, M. Schunter, A. Tanner, L. V. Doorn, E. V. Herreweghen, M. Waidner, and S. Yoshihama. Trusted Virtual Domains: Secure Foundations for Business and IT Services. Technical Report Research Report RC23792, November 2005.

[9] S. Cabuk, C. Dalton, H. Ramasamy, and M. Schunter. Towards Automated Provisioning of Secure Virtualized Networks, 2007. ACM CCS 2007.

[10] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Proceedings of the Digital Rights Management Workshop 2002, volume 2696 of LNCS*, pages 61–80, 2002.

[11] European Multilaterally Secure Computing Base (EMSCB) Project. Towards Trustworthy Systems with Open Standards and Trusted Computing, 2008. http://www.emscb.de.

[12] G. Faden. Multilevel Filesystems in Solaris Trusted Extensions. In *Proceedings of the 12th ACM Symposium in Access Control Models and Technologies*, pages 121–126. ACM, 2007.

[13] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N. Asokan. Beyond Secure Channels, 2007. ACM Workshop on Scalable Trusted Computing 2007.

[14] E. Gaudet. DRM vs. ERM: battle to control data. http://www.networkworld.com/news/tech/2006/121806techupdate.html, December 2006.

[15] J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, , and R. Cáceres. Trusted Virtual Domains: Toward Secure Distributed Services. In *1st IEEE Workshop on Hot Topics in System Dependability*, June 2005.

[16] V. Haldar, D. Chandra, and M. Franz. Semantic remote attestation: a virtual machine directed approach to trusted computing. In *VM'04: Proceedings of the 3rd conference on Virtual Machine Research And Technology Symposium*, pages 3–3, Berkeley, CA, USA, 2004. USENIX Association.

[17] T. Jaeger, R. Sailer, and U. Shankar. PRIMA: policy-reduced integrity measurement architecture. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 19–28, New York, NY, USA, 2006. ACM.

[18] Y. Katsuno, M. Kudo, P. Perez, and R. Sailer. Towards Multi-Layer Trusted Virtual Domains, 2006. The 2nd Workshop on Advances in Trusted Computing.

[19] J. Liedtke. On Microkernel Construction. In *15th ACM Symposium on Operating System Principles*, 1995.

[20] V. Likitalo. Remote Attestation and Peer-to-Peer Net. http://www.tml.tkk.fi/Publications/C/18/likitalo.pdf, 2005.

[21] M. Matson and M. Ulieru. Persistent Information Security – Beyond the e-Commerce Threat Model. In *The Eighth International Conference on Electronic Commerce*, August 2006.

[22] Microsoft. Microsoft Windows Rights Management Services for Windows Server 2003 - Helping Organizations Safeguard Digital Information from Unauthorized Use. Whitepaper, 2003.

[23] J. Park, R. Sandhu, and J. Schifalacqua. Security architectures for controlled digital information dissemination. In *ACSAC '00: Proceedings of the 16th Annual Computer Security Applications Conference*, page 224, Washington, DC, USA, 2000. IEEE Computer Society.

[24] S. Pearson. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[25] J. Reid and W. Caelli. DRM, Trusted Computing and Operating System Architecture. 2005.

[26] A. Sadeghi, M. Wolf, C. Stüble, N. Asokan, and J. Ekberg. Enabling Fairer Digital Rights Management with Trusted Computing, October 2007.

[27] R. Sailer, T. Jaeger, E. Valdez, R. Cáceres, R. Perez, S. Berger, J. L.Griffin, and L. van Doorn. Building a MAC-based Security Architecture for the Xen Opensource Hypervisor. In *21st Annual Computer Security Applications Conference (ACSAC)*, Dec 2005.

[28] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association.

[29] R. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by Trusted Computing and PEI models. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 2–12, New York, NY, USA, 2006. ACM.

[30] R. Sandhu and X. Zhang. Peer-to-peer access control architecture using trusted computing technology. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 147–158, New York, NY, USA, 2005. ACM.

[31] S. Schechter, R. Greenstadt, and M. Smith. Trusted Computing, Peer-To-Peer Distribution, and the Economics of Pirated Entertainment. In *The Second Annual Workshop on Economics and Information Security*, May 2003.

[32] E. Sebes and M. Stamp. Solvable Problems in Enterprise Digital Rights Management. http://home.earthlink.net/~mstamp1/papers/DRMsebes.pdf, 2004.

[33] R. Thomas and R. Sandhu. Towards a Multi-dimensional Characterization of Dissemination Control. In *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, page 197, Washington, DC, USA, 2004. IEEE Computer Society.

[34] Trusted Computing Group. TCG Specification Architecture Overview. Trusted Computing Group: https://www.trustedcomputinggroup.org/groups/TCG_1_3_Architecture_Overview.pdf, Mar. 2003. Specification Revision 1.3 28th March 2007.

[35] Trusted Computing Group. TPM Main Specification v1.2. https://www.trustedcomputinggroup.org, November 2003.

[36] Y. Yu and T. Chiueh. Display-Only File Server: A Solution against Information Theft Due to Insider Attack. October 2004.