



Digital Rights Management Demonstrator

Requirements, Analysis, and Design

Authors: Andre Osterhues,
Marko Wolf

Institute: Ruhr-University Bochum

Date: March 2, 2007

Abstract: This document describes a Digital Rights Management (DRM) demonstrator focussing on fair use aspects as well as overall security and unforgeability of the system. The DRM demonstrator allows the content providers to make sure that the previously defined license conditions are met by the user platform (policy enforcement). On the other hand, users are able to backup their media or transfer it to another device or user (provided that the accompanying license allows this).

Contents

I	Requirements Specification	3
1	Motivation and Problem Description	3
2	Security Environment	3
2.1	Roles and Interfaces	3
2.2	Assumptions	4
	/A 10/ Strong isolation	4
	/A 20/ Secure communication	4
	/A 30/ Fresh storage	4
2.3	Threats	4
	/T 10/ TCB Integrity Violation	4
	/T 20/ Malicious Device Drivers	4
	/T 30/ System Virtualization	4
	/T 40/ Trojan Horse	5
	/T 50/ Unauthorized content access	5
	/T 60/ License manipulation	5
	/T 70/ Client software manipulation	5
3	Functional Requirements (Use Case Model)	5
3.1	Target Groups	5
3.2	Roles and Actors	5
3.3	Overview	5
3.4	Use Cases	5
4	Security Objectives & Security Requirements	12
4.1	Security Objectives	12
	/SO 10/ Separability	12
	/SO 20/ Privacy	12
	/SO 30/ License integrity	12
	/SO 40/ Licenses unforgeability	12
	/SO 50/ License enforcement	12
	/SO 60/ License availability	12
	/SO 70/ Trust verification of TCB	12
4.2	Security Requirements	12
	/SR 10/ Integrity of the TCB	12
5	Supplementary Requirements	13
5.1	Preconditions	13
	/PR 100/ L4-based resource management layer	13
5.2	Required Criteria	13
	/RC 10/ L4 Support	13
5.3	Desired Criteria	13
	/DC 10/ Multi-User Support	13
5.4	Distinguishing Criteria	13

5.5	Execution Environment	13
5.5.1	Software	13
5.5.2	Hardware	13
5.6	Development Environment	13
5.6.1	Software	13
5.6.2	Hardware	14
II	Design	15
6	Architecture Description	15
7	Design Model	15
7.1	Design System User Platform	15
7.2	Design Subsystem Usermode Linux (Legacy OS)	15
7.2.1	Use-Case Realization / UC 10 / (PERSEUS Initialization)	16
7.2.2	Use-Case Realization / UC 20 / (Certificate Creation)	17
7.2.3	Use-Case Realization / UC 50 / (Content Usage)	17
7.2.4	Use-Case Realization / UC 60 / (Content Transfer)	17
7.3	Design Subsystem Security Services	17
7.3.1	Design Trust Manager	17
7.3.2	Design Policy Manager	17
7.3.3	Policy Manager Initialization	18
7.3.4	Use-Case Realization / UC 50 / (Content Usage)	18
7.3.5	Use-Case Realization / UC 60 / (Content Transfer)	18
7.3.6	Design SoundServer	19
7.4	Design System Provider	19
7.4.1	Design Licensor and Composer	19
7.4.2	Use-Case Realization / UC 30 / (Content Licensing and Wrapping)	20
7.4.3	Use-Case Realization / UC 40 / (Content Aquisition)	20

Part I

Requirements Specification

1 Motivation and Problem Description

Distributed e-commerce applications increasingly demand for sophisticated functional and security requirements: The involved parties have different (security) policies regarding access and usage of digital goods. However, the existing solutions, most prominent digital rights management systems, suffer from various weaknesses: They mainly focus on the requirements of content providers and not on those of consumers. Moreover, they are vulnerable to various attacks, particularly on open computing platforms where consumers/users have the total control and can run exploits as well as reconfigure the underlying operating system.

The *Digital Rights Management Demonstrator* provides open security architecture that copes with these problems: It allows the transfer of rights and the enforcement of stateful licences, i.e., licences whose state changes depending on how often or how long they are used. Further, it provides security properties such as strong isolation of applications that are used to enforce the user's privacy policy, and a legacy operating system (currently Linux) in parallel.

2 Security Environment

The security environment consists of two platforms:

- Trusted Content Server TS
- Trusted Consumer Platform TC

The Trusted Content Server TS is used to store the media content and send it to the user. The Consumer Platform TC is the platform where the media content is consumed by the user.

2.1 Roles and Interfaces

The main parties involved are *Providers* and *Users*. A *Provider* represents both the content provider and the licensor. A *Provider* distributes content c together with an appropriate license lic_c . License lic_c defines the usage-rights applicable to c . A *User* consumes contents c provided by *Provider*. Therefore, *User* requests a usage-right on c and securely receives rendered content according to the corresponding license lic_c .

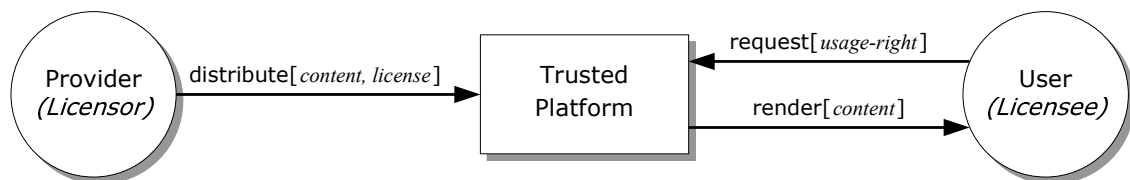


Figure 1: Roles and interfaces.

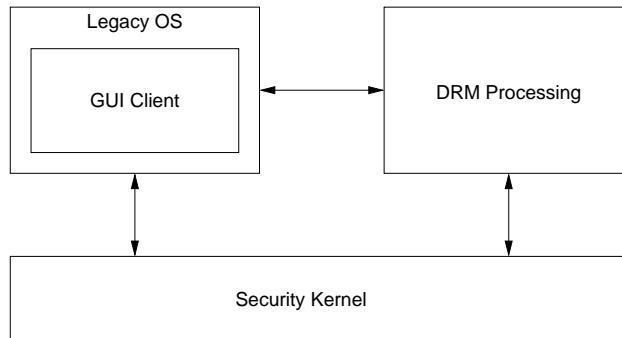


Figure 2: Architecture overview: The GUI Client inside of the Legacy OS on the left and the DRM Processing on the right run as applications on top of the Security Kernel.

2.2 Assumptions

/A 10/ Strong isolation

The underlying security kernel provides mechanisms to support strong isolation between applications running on top of it. An application can be, for example, a security critical service or a user-mode operating system.

/A 20/ Secure communication

The communication between applications is performed in a secure, controlled way. This can be achieved by using access control on inter-process communication.

/A 30/ Fresh storage

The security kernel provides fresh storage, i.e., a means to store data in a way immune to replay attacks. This could be obtained by using TPM 1.2 monotonic counters.

2.3 Threats

/T 10/ TCB Integrity Violation

An adversary may try to violate security policies by maliciously manipulating or replacing software components of the TCB. Examples are manipulations of the bootloader, to replace certain security-critical components, or to replace the whole security kernel.

/T 20/ Malicious Device Drivers

An adversary may try to violate security policies by manipulating/replacing device drivers such that hardware functions (e.g., direct memory access) can be used to violate security policies.

/T 30/ System Virtualization

An adversary may try to access sensitive data of the security kernel by executing it on top of a virtual machine monitor (VMM) that is under his control.

/T 40/ Trojan Horse

An adversary may try to get access to sensitive information by deceiving *Administrators* or *Users* (cf. Section 3.2) such that a compartment under control of the adversary claims to be a trusted compartment.

/T 50/ Unauthorized content access

An adversary may try to access the content unauthorized (e.g. by using another user's license).

/T 60/ License manipulation

An adversary may try to manipulate the license to circumvent.

/T 70/ Client software manipulation

An adversary may try to violate the client's security by maliciously manipulating the client software, i.e. the underlying operating system, device drivers and/or client application. For example, the file system might be changed to read-only such that a stateful license cannot be updated, or a device driver might be replaced by one that writes the decrypted content to the harddisk instead of displaying it.

3 Functional Requirements (Use Case Model)

3.1 Target Groups

- Home user (Single-user platform at home)

3.2 Roles and Actors

- *Provider*: The *Provider* controls the content server and defines license lic_c .
- *User*: The *User* controls the client platform and consumes content c
- *RemoteUser*: The *RemoteUser* is a user located on another platform who wants to receive c from the *User*.

3.3 Overview

Figure 3 illustrates the use cases detailed in Section 3.4 and their dependencies.

3.4 Use Cases

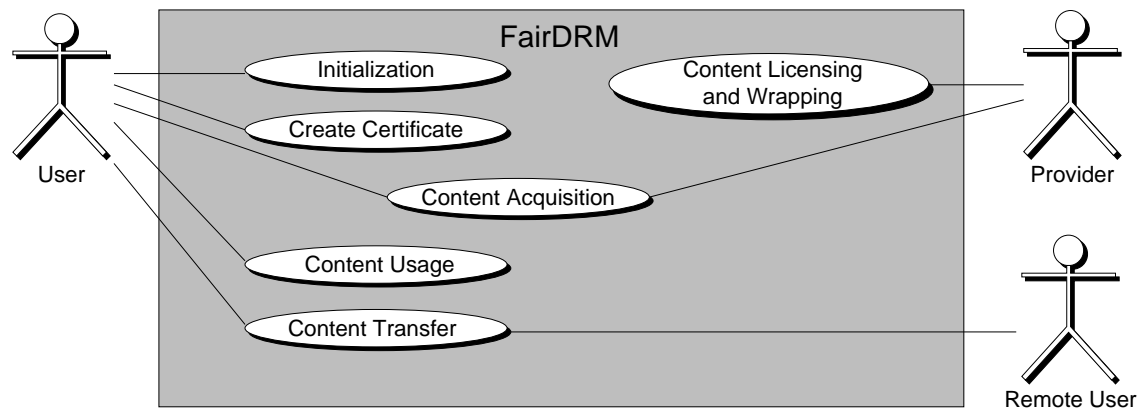


Figure 3: Use case view of the DRM demonstrator

USE CASE UNIQUE ID	/ UC 10 /
TITLE	PERSEUS Initialization
DESCRIPTION	A user boots the PERSEUS system.
ACTORS	<i>User</i>
RATIONALE	Booting the PERSEUS system.
PRECONDITIONS	The PERSEUS system is installed.
POSTCONDITIONS	The PERSEUS system can be used.
NORMAL FLOW	<ol style="list-style-type: none"> 1. <i>User</i> activates the PERSEUS system. 2. The PERSEUS system boots.

USE CASE UNIQUE ID	/ UC 20 /
TITLE	Certificate Creation
DESCRIPTION	<i>User</i> creates certificate <i>cert</i> from its client TC, so that <i>Provider</i> can verify the platform configuration of TC.
ACTORS	<i>User</i>
RATIONALE	Retrieve certificate <i>cert</i> of <i>User</i> 's platform TC.
PRECONDITIONS	TC runs the PERSEUS system (/ UC 10 /).
NORMAL FLOW	<ol style="list-style-type: none"> 1. <i>User</i> clicks on <i>Create Certificate</i> icon. 2. <i>User</i> uses file dialog to set certificate storage position. 3. <i>User</i> confirms certificate creation. 4. <i>Program</i> shows certificate creation progress. 5. <i>Program</i> shows generation success or error.

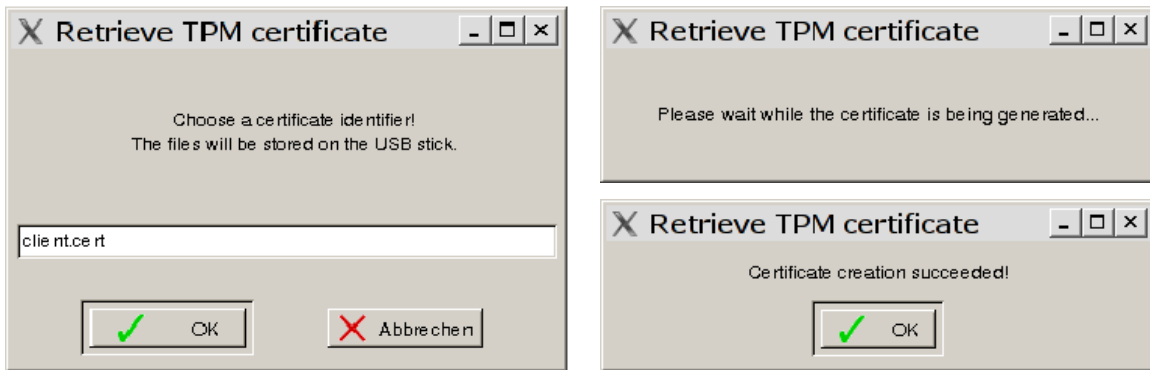


Figure 4: Create certificate dialogs

USE CASE UNIQUE ID	/ UC 30 /
TITLE	Content Licensing and Wrapping
DESCRIPTION	<i>Provider</i> uses TS to create a license lic_c for content c . Finally, lic_c and c are bound to the client certificate $cert$ from TC and wrapped into a secured file container $mediafile$.
ACTORS	<i>Provider</i>
RATIONALE	Create lic_c and wrap tuple $(c, lic_c, cert)$ into a secure file container $mediafile$ for target platform TC.
PRECONDITIONS	<i>Provider</i> has a website to communicate with <i>User</i> . <i>Provider</i> needs the certificate from TC (/ UC 20 /).
NORMAL FLOW	<ol style="list-style-type: none"> 1. <i>Provider</i> receives license request from <i>User</i> via <i>Provider</i>'s web site. 2. <i>Provider</i> checks for valid parameters (certificate, content, usage rights...). 3. <i>Provider</i> loads requested content c. 4. <i>Provider</i> creates license lic_c according given parameters. 5. <i>Provider</i> creates encrypted file container $mediafile$ with $(c, lic_c, cert) \rightarrow mediafile$. 6. <i>Provider</i> shows error or provides $mediafile$ to <i>User</i> via <i>Provider</i>'s web site.

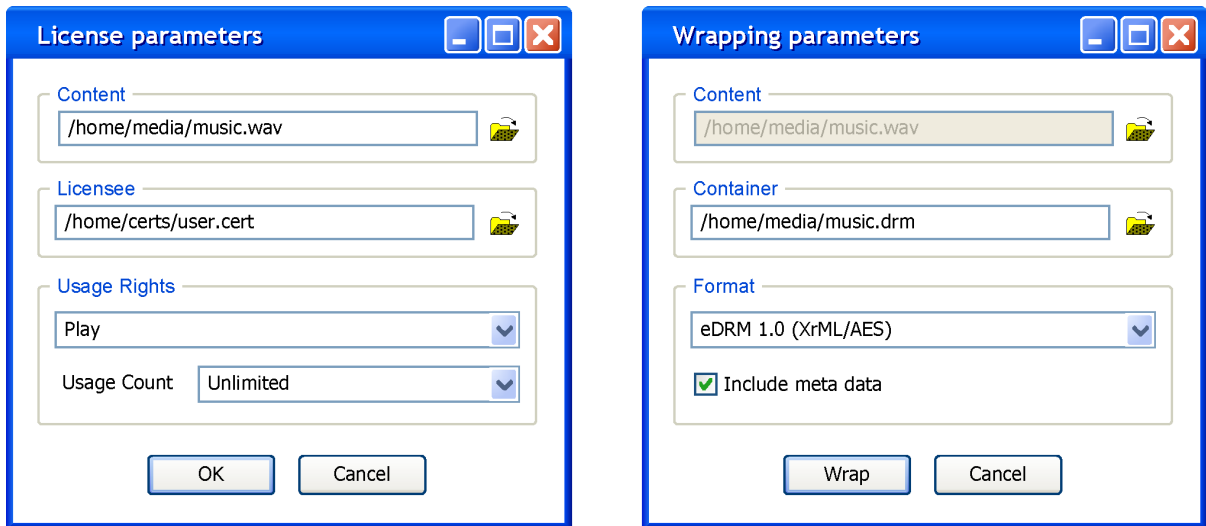


Figure 5: License and wrap content dialogs

USE CASE UNIQUE ID	/ UC 40 /
TITLE	Content Acquisition
DESCRIPTION	<i>User</i> wants to acquire some media content c according to a specific license lic_c .
ACTORS	<i>User</i> , <i>Provider</i>
RATIONALE	Acquire c from <i>Provider</i> according to lic_c .
PRECONDITIONS	TC runs the PERSEUS system (/ UC 10 /). TS is able to provide / UC 30 /.
NORMAL FLOW	<ol style="list-style-type: none"> 1. <i>User</i> selects some content c from a list on <i>Provider</i>'s web site. 2. <i>User</i> selects media and license parameters (media format, license count or license period. . .). 3. <i>Provider</i>'s web site displays pricing information for the selected content and license. 4. <i>User</i> agrees to pricing information and clicks on "Purchase" button. 5. <i>Provider</i> generates a license lic_c according to <i>User</i>'s selection. 6. <i>Provider</i> wraps license lic_c and content c according to / UC 30 /. 7. <i>User</i> downloads encrypted <i>mediafile</i> from <i>Provider</i>'s web site.

USE CASE UNIQUE ID	/ UC 50 /
TITLE	Content Usage
DESCRIPTION	<i>User</i> opens a secure file container <i>mediafile</i> to enclosed content <i>c</i> according to enclosed license <i>lic_c</i> .
ACTORS	<i>User</i>
RATIONALE	Access content <i>c</i> according to <i>lic_c</i> .
PRECONDITIONS	TC runs the PERSEUS system (/ UC 10 /). <i>User</i> has made an acquisition according to / UC 40 /.
NORMAL FLOW	<ol style="list-style-type: none"> 1. <i>User</i> clicks on <i>MediaPlayer</i> icon. 2. <i>User</i> uses file dialog to load <i>mediafile</i> (from file system). 3. <i>User</i> clicks on button “Access Media”. <ol style="list-style-type: none"> (a) <i>MediaPlayer</i> confirms access authorization. <i>MediaPlayer</i> activates rendering process $c \rightarrow c'$. or (b) <i>MediaPlayer</i> denies access authorization. 4. <i>MediaPlayer</i> dialog closes.
EXTENSIONS	Upon a successful rendering activation, <i>MediaPlayer</i> provides a “Stop Rendering” button to abort current rendering process.

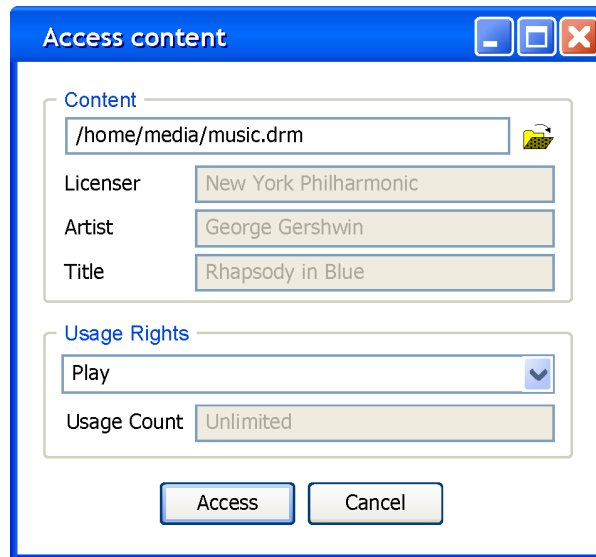


Figure 6: Access content dialog

USE CASE UNIQUE ID	/ UC 60 /
TITLE	Content Transfer
DESCRIPTION	<i>User</i> wants to transfer the content c according to the license lic_c from its platform TC to another platform TC_{dest} .
ACTORS	<i>User</i>
RATIONALE	Transfer content c from TC to TC_{dest} according to lic_c .
PRECONDITIONS	TC runs the PERSEUS system (/ UC 10 /). TC has a certificate $cert_{dest}$ from TC_{dest} (/ UC 20 /).
NORMAL FLOW	<ol style="list-style-type: none"> 1. <i>User</i> clicks on <i>Transfer</i> icon. 2. <i>User</i> uses file dialog to load resp. <i>mediafile</i> (from file system). 3. <i>User</i> uses file dialog to load $cert_{dest}$ (from file system). 4. <i>User</i> uses file dialog to set the storage position of <i>mediafile</i>'. 5. <i>User</i> clicks on button "Transfer Media". <ol style="list-style-type: none"> (a) <i>Transfer</i> program confirms transfer authorization. <i>Transfer</i> creates destination file <i>mediafile</i>'. or (b) <i>Transfer</i> denies transfer authorization. 6. <i>Transfer</i> dialog closes.
EXTENSIONS	Upon a successful creation of <i>mediafile</i> ', <i>User</i> moves <i>mediafile</i> ' to TC_{dest} , where it can be accessed according to / UC 50 /.

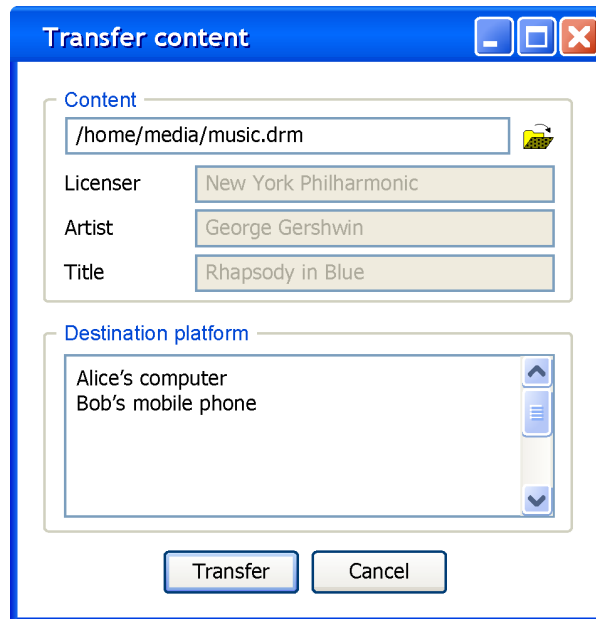


Figure 7: Transfer content dialog

4 Security Objectives & Security Requirements

4.1 Security Objectives

/SO 10/ Separability

The use of different security-critical applications based on the PERSEUS security architecture has to be at least as secure as the execution of the same applications on physically separated computing platforms connected via network.

/SO 20/ Privacy

The user's privacy policy must not be violated by the usage of licenses. This includes that the overall system enforces least privilege such that components not under full control of the user cannot access and leak information beyond the enforcement of licenses.

/SO 30/ License integrity

Both user and provider require that unauthorized manipulations of licenses must not be possible, as a license constitutes a valid contract between both parties and neither party wants the license to be changed to his disadvantage.

/SO 40/ Licenses unforgeability

The provider requires that an unauthorized issuing of a new license must not be possible, i.e., only authorized parties (e.g., providers) should be able to issue new licenses.

/SO 50/ License enforcement

The provider requires that the license must be enforced upon acceptance by the user. This implies that the user should be allowed to access and use the content only according to user-rights provided by the license.

/SO 60/ License availability

Users require that legally obtained licenses can be used at any time. This especially requires that denial of service attacks should not happen.

/SO 70/ Trust verification of TCB

The components of the TCB can be evaluated to allow a verification of trust.

4.2 Security Requirements

/SR 10/ Integrity of the TCB

The TCB should be protected from manipulations to guarantee the enforcement of security policies. No modification of the TCB must be allowed, except for changes that have been authorized by the platform owner.

5 Supplementary Requirements

5.1 Preconditions

/PR 100/ L4-based resource management layer

L4 must offer a method of achieving access control for L4 IPC calls between different L4 tasks (which constitute compartments).

5.2 Required Criteria

/RC 10/ L4 Support

The realization of the use cases should be deployable on an L4-based architecture.

5.3 Desired Criteria

/DC 10/ Multi-User Support

The PERSEUS security architecture should be able to handle multiple users.

5.4 Distinguishing Criteria

5.5 Execution Environment

5.5.1 Software

- Microkernel-based Architecture
 - Fiasco L4V2 μ -kernel
 - L4Env V0.2
 - L4-Linux
- Hypervisor-based Architecture
 - Xen 3.0
 - Xen-Linux 3.0

5.5.2 Hardware

- Intel LaGrande Platform
- AMD Pacifica Platform
- TPM 1.2 Platform

5.6 Development Environment

5.6.1 Software

- Linux 2.6.x
- gcc 3.4.x
- eclipse-3.1
- Borland Together 6.2
- AMD Pacifica Simulator

5.6.2 Hardware

- Intel LaGrande Development Platform
- AMD Pacifica Development Platform
- TPM 1.2 Development Board
- VESA 2.0 Graphics adapter

Part II

Design

6 Architecture Description

The overall architecture of the DRM demonstrator consists of two main components, the Trusted Content Server TS and the Trusted Consumer Platform TC (see Figure 8). After receiving a client certificate from TC, TS generates a license using the Licensor and composes the encrypted media file using the client certificate, the license and the content. The media file is then sent to TC, where it can be played back using the media player. The media player communicates with the policy manager, which checks whether the conditions in the license (play rights, platform) are met. If they are met, the policy manager sends the output to the sound server.

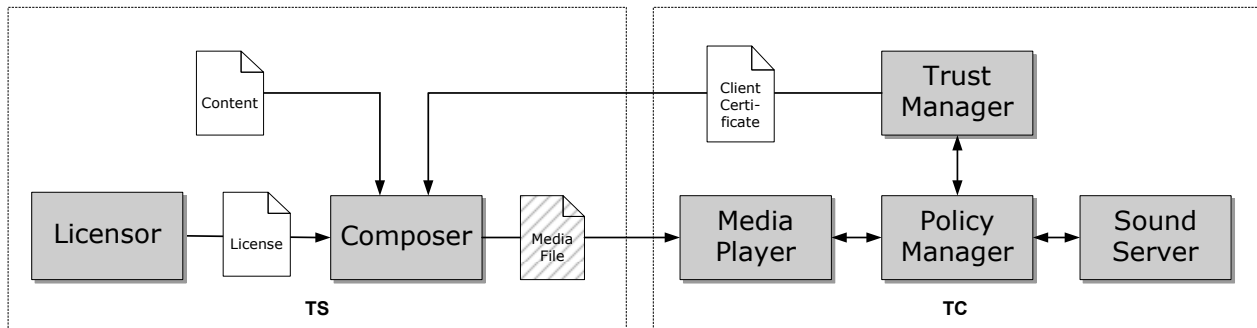


Figure 8: Architecture overview.

7 Design Model

The Design Model contains designs for the User platform (and the Usermode Linux subsystem), the Security Services, and the Provider platform.

7.1 Design System User Platform

Figure 9 depicts the consumer platform TC in more detail. Specifically, it shows where each of the before-mentioned services reside: the Media Player is a normal application running within the Legacy OS (currently Linux), where it also can locate the media. The Policy Manager—as a central component of the system—controls the communication between the Media Player and the security services.

Figure 10 shows the functional dependencies between the components running on the *User* platform TC.

7.2 Design Subsystem Usermode Linux (Legacy OS)

This subsystem consists of all realizations on *User*'s untrusted domain, i.e. all programs running inside the Legacy OS. This includes the certificate generation program *gencert*, the web browser, and the Media Player *mplayer*.

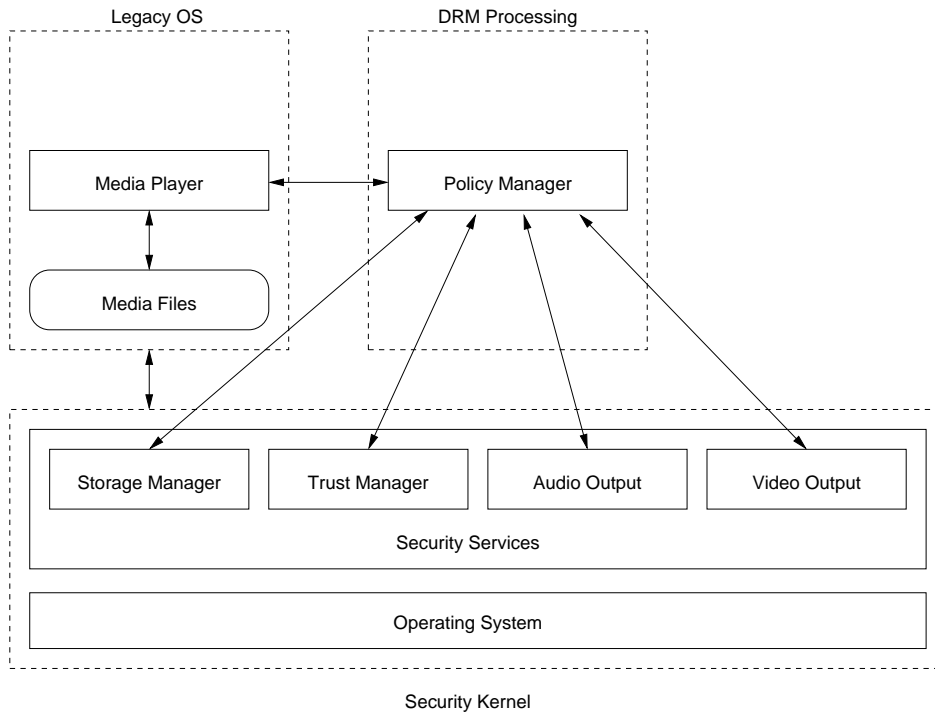


Figure 9: Detailed consumer platform (TC) architecture.

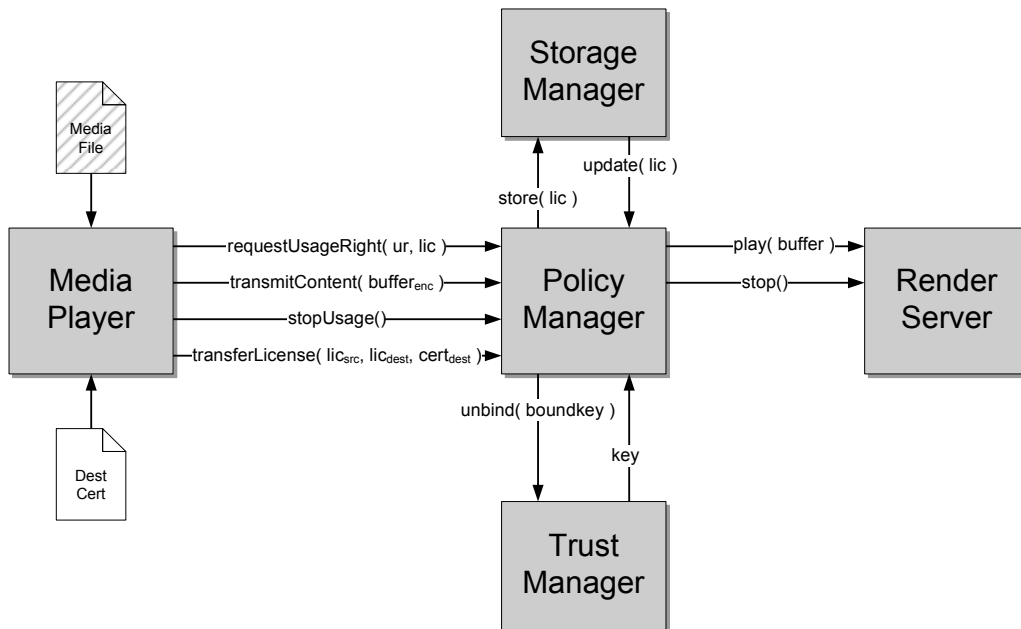


Figure 10: Basic picture client side.

7.2.1 Use-Case Realization / UC 10 / (PERSEUS Initialization)

1. PERSEUS init implementation. The *User* boots the PERSEUS system.

7.2.2 Use-Case Realization / UC 20 / (Certificate Creation)

1. The *gencert* program, which runs within the Legacy OS, accesses the Trust Manager to create a Certificate of the platform.

7.2.3 Use-Case Realization / UC 50 / (Content Usage)

1. Read and interpret composed media file.
2. Send BLOB buffer (bound AES key), SRK-encrypted binding key, license buffer, destination certificate and usage right UR_PLAY to the Policy Manager.
3. Display error message or send encrypted content consecutively to rendering service.

7.2.4 Use-Case Realization / UC 60 / (Content Transfer)

1. Read and interpret composed media file.
2. Send BLOB buffer (bound AES key), SRK-encrypted binding key, license buffer, destination certificate and usage right UR_TRANSFER to the Policy Manager.
3. On success, recompose received BLOB, SRK-encrypted binding key and license buffer to destination media file.

7.3 Design Subsystem Security Services

All realizations on *User's* trusted software layer (TSL). This includes the Trust Manager, the Policy Manager, and the Sound Server.

7.3.1 Design Trust Manager

The Trust Manager is a trusted service that provides an interface to the platform's TPM. It provides functionalities to:

1. generate a platform certificate
2. bind content
3. unbind content

7.3.2 Design Policy Manager

The Policy Manager is a trusted service that:

1. receives a media file and an according license
2. checks whether the media file is allowed to be rendered on the *User's* platform
3. verifies the rights (usage time period, usage count) defined in the license
4. updates the usage count of the license (if necessary)
5. renders the output

It connects to the Trust Manager to unbind the content. To render the output, the Policy Manager communicates with the Sound Server over secure IPC.

7.3.3 Policy Manager Initialization

1. Register at Names Service.
2. Provide public IPC interface.

7.3.4 Use-Case Realization / UC 50 / (Content Usage)

1. Receive BLOB buffer (bound AES key), SRK-encrypted binding key, license buffer and usage right.
2. Interpret license (i.e., create object from buffer).
3. Verify license signature (via RSA).
4. Read license freshness values via LicenseStore.
5. Verify usage right (via verified license).
6. Find TrustManager.
7. Request TrustManager for unbind of BLOB.
8. Store decryption key.
9. Find RenderServer for corresponding usage right (i.e., SoundServer).
10. Apply usage right (i.e., decrease counters if available).
11. Write new license freshness values via LicenseStore.
12. Decrypt contents and send them to RenderServer.

7.3.5 Use-Case Realization / UC 60 / (Content Transfer)

1. Receive BLOB buffer (bound AES key), SRK-encrypted binding key, license buffer, destination certificate and usage right UR_TRANSFER.
2. Interpret license (i.e., create object from buffer).
3. Verify license signature (via RSA).
4. Read license freshness values via LicenseStore.
5. Verify right for transfer (incl. verification of destination certificate)
6. Find TrustManager.
7. Request TrustManager for unbind of BLOB.
8. Apply usage right UR_TRANSFER (i.e., decrease counters if available).
9. Write new license freshness values via LicenseStore.
10. Bind decryption key to destination certificate.
11. Return BLOB, SRK-encrypted binding key and license buffer.

7.3.6 Design SoundServer

The Sound Server is a service which provides a secure way to output audio data to the audio hardware. The audio data can be provided in RIFF-WAVE or MP3 format. The interface consists of two basic functionalities (Play and Stop). For Play, the following procedure is executed:

1. Interpretation of header (RIFF-WAVE or MP3).
2. Setting of audio hardware (sampling rate, channels, bits per channel, volume).
3. Playback of audio data.

For Stop, the audio hardware is muted and reset.

7.4 Design System Provider

All realizations on the *Provider's* site. This includes license creation using the *licensor* program and content encryption using the *composer* program, as depicted in Figure 11.

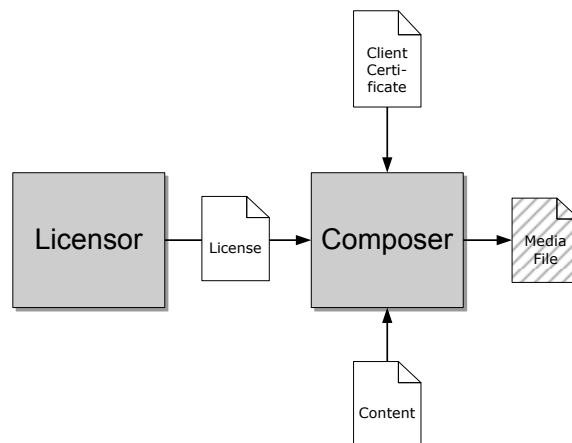


Figure 11: Content encryption & licence creation.

7.4.1 Design Licensor and Composer

Licensor and *composer* are the two main programs on the *Provider's* side.

Licensor The *licensor* program transforms a list of conditions such as playback count, playback time period and transfer count into a license.

Composer The composer reads the license and signs it with an RSA signature. Then it uses the binding key, which—among other information about the *User's* platform TC—is contained in the *User's* certificate, to bind a (random) symmetric media key to the *User's* platform. The media key is used to encrypt the content. Finally, the signed license, the bound media key and the encrypted content are wrapped into a single file.

7.4.2 Use-Case Realization / UC 30 / (Content Licensing and Wrapping)

1. Checking of the *User's* certificate. The *Provider* can choose whether to trust the *User's* platform TC or not.
2. Creation of a license. A license is generated according to the conditions that both *User* and *Provider* agree upon.
3. Wrapping of license and content. The content is encrypted using the certificate's binding key. The signed license and the encrypted content are wrapped into a single encrypted media file.

7.4.3 Use-Case Realization / UC 40 / (Content Aquisition)

1. The *Provider* offers his service using a web server displaying a list of media files and license conditions to choose from.
2. The *User* sends his platform certificate to the web server.
3. The *User* selects a media file and agrees upon some license conditions.
4. The web server processes the selection and produces an encrypted media file containing license and media content.