Digital Rights Management (DRM) and Trusted Computing

Thierry Sans
What I will try to address in this talk

- What is DRM? (technologically speaking)
- Why DRM technologies seem not work in practice?
- Is Trusted Computing the silver bullet?
- Why the usage of DRM is so controversial?
- What is the future for DRM technologies?
The boundaries of a Information System

How can I be sure that my data are safe outside of my trusted domain?
Here is where the new problem comes in.

How can I be sure that Bob will not tamper his system to bypass control mechanisms and have a full access to my data?
The big picture

▪ Situation
  • Alice has sensitive information to share with Bob

▪ Problem
  • Alice does not trust Bob in ensuring the confidentiality and the integrity of her information

▪ Solution
  • Alice must have a way to ensure that Bob will use this information according to her security policy
  
Digital rights Management (DRM)
Anatomy of a (certain) DRM framework

- License Server
- Content Server
- Rendering Application

Information → Content Server → Content

Metadata

Policy → License Server → License

Rendering Application
Security Focus: The content

- The content carries *sensitive* information
  - Protecting *unauthorized disclosure* (confidentiality) of the payload
  - Preventing *unauthorized modification* (integrity) of the payload and of the meta-data

The content must be protected during transport, *storage* and usage

??
Protection scheme

- Naïve (and usual) approach
  - Encrypt the content with a symmetric key

- Problem
  - The rendering application needs the key to decrypt the content and performed authorized actions
  - This key must not be uncovered by the end-user

- Hypothesis
  - The rendering application is trustworthy
Different solutions in practice

- Solution 1
  - Retrieve the key during execution (on-line)
  - A user may need to use the content off-line

- Solution 2
  - Every content is encrypted with the same symmetric key and this key is hidden in the rendering application
  - If the key is compromised, all the contents are compromised (like CSS for DVDs)
Other solution

- Solution 3

  - The symmetric key is transferred
    - with the content itself (like *Apple Fairplay*)
    - with the license (like *Microsoft Windows DRM*)

  - But then the key itself needs to be protected

- Solution 3 extended

  - The key is protected using a private key (asymmetric)

  - But the public needs to be protected as well

  - isn't it the same problem though?!
DRM framework with content protection

Content Server

License Server

Rendering Application
Security Focus: The license

- The license contains information that needs to be protected
  - The usage control policy rules
  - (possibly) the key to decrypt the content
  - Information linking the content and the license
    - content hash
    - or a reference to a digital signature
    - or a reference to a watermark
Writing the usage control policy

- Need a way to identify the right consumer
  - May reference a certain user identity or a certain application instance

- Need a language to describe what are the authorized usage of the content
  - Rights Expression Language (REL)

- But what are the actions that the rendering application enforces
  - Rights Data Dictionary (RDD)
DRM framework with content protection

Diagram showing the interaction between Content Server, License Server, and Rendering Application.
Retrieving a license (and its key)

Step 0: Implantation of the certificate

Step 1: License Request

Step 2: Application authentication

Step 3: License Response

Authority

License Server

Rendering Application
Security Focus: The Rendering Application

- The rendering application must be trustworthy
  - The functionalities (content rendering) must not be tampered
  - The license verification must not be bypassed
  - The (user or application) identity and the authentication mechanism must not be tampered
  - The key must not be uncovered (hidden in the application code and in the memory during execution)

The rendering application is a critical application since it is executed on a non-trusted platform
Protecting the rendering application

- How to guarantee that the application has not been tampered?

  - The black box model (security through obscurity)
  - The source code and/or the compiling method must not be revealed (code obfuscating)
  - The Trusted Computing approach
  - Having a proof that the code has not been tampered and cannot been tampered during execution
Is it possible to have open-source DRM systems?

- The program can be “partially” open-source (without cryptographic key information) but the binary cannot be recompiled and used by the end-user

Problem with the GPL3 that extends the concept of open design (affecting the code, the hardware and cryptographic key information)
What I will try to address in this talk

- What is DRM? (technologically speaking)
- Why DRM technologies seem not work in practice?
- Is Trusted Computing the silver bullet?
- Why the use of DRM is so controversial?
- What is the future for DRM technologies?
Existing (commercial) DRM applications

• Rudimentary DRM systems (fixed license)
  • Audio CD
    • XCP (Extended Copy Protection)
      also know as the “Sony Rootkit”
  • DVD
    • CSS (Content Scramble System)
  • HD-DVD (RIP) and Blue-Ray
    • AACS (Advanced Access Content System)
  • Apple products
    • FairPlay (implemented in iTunes/iPod)
Existing (commercial) DRM applications

• More advanced DRM systems (user-defined license)
  • Adobe products
    • Adobe DRM (for the eBooks)
  • Microsoft PlaysForSure products
    • (Janus) Windows Media DRM
      (an implementation interpretation of MPEG-21)
  • Mobile Phones
    • OMA-DRM (Open Mobile alliance pushed by Nokia)
• ... far to be exhaustive
Standards and open frameworks

- **Standards**
  - ISO/IEC MPEG-21
  - OMA-DRM

- **Open DRM Framework**
  - Open-IPMP
  - Open Media Commons
The “smart cow” problem

"It only takes one smart cow to open the latch of the gate ....

...and then all the other cows follow"

Programs breaking DRM protection

• DeCSS breaking CSS (DVDs)
• FairUse4WM breaking Windows Media DRM
• QTFairUse6 breaking Apple FairPlay
The Kerckhoffs' principle (1883)

- “The enemy knows the system”
  - The security of a communication should not rely on the fact that the algorithms are secrets

Most of existing DRM applications have a wrong security design since they violate this principle.
Achieving security is a complex task

- Is it possible to design a trusted rendering application on a non trusted platform?
  - The application is dependable of the OS
  - The OS is dependable of the hardware

It seems hard ... might even being impossible

Running a trustworthy application may necessarily imply having a trusted platform as well
What I will try to address in this talk

- What is DRM? (technologically speaking)
- Why DRM technologies seem not work in practice?
- Is Trusted Computing the silver bullet?
- Why the use of DRM is so controversial?
- What is the future for DRM technologies?
A (magic?) solution: Trusted Computing

- The application must be trustworthy
  - Trusted Third Party (TTP) is needed to attest that the application will behave in an expected way

- Trusted Computing Platform (TCPA)
  - Embed a TTP in a chip (TPM)
What “trust” means? A naïve definition

- What “trusting somebody or something” means?
  - I know or I believe that he/she/it will behave in an expected way to perform a given task

- How can I trust someone or something?
  - Depends on my beliefs and on my knowledge
  - Depends on the sensitivity of the task

From “wait and see” toward “prove me first”
The Trusted Computing definition of trust

- **With Trusted Computing, what trust “means”?**
  - I am sure that the platform will behave as expected to achieve a given task

- **How can I be sure about it?**
  - The platform can show me assertions about:
    - What are the current applications running
    - What is the operating system (and its configuration)
    - What is the hardware
  - ... and I believe that they are all trustworthy for the intended purpose
How Trusted Computing Platforms work?

- How can I trust the assertions made by a platform? Because I trust
  
  • The way they have been created by performing some measures on the platforms
  
  • The way they have been communicated to me with a proof that the measurement has been done by a trusted third party

Trusted Computing provides the trusted third party also called the root of trust
Overview

Core Root of Trust for Measurement (CRTM) + Trusted Platform Module (TPM)

Trusted Computing Platform (TCPA)
Trusted Platform Module (TPM)

- **Composed of**
  - A series of registers to store the measures
  - Cryptographic processor
  - A 2048 bits RSA public key and private key pair
    - Create randomly on the chip at manufacture time
    - Cannot be changed
    - The private key never leaves the chip
Building the Chain of Trust (Transitive Trust)

Software Component

Measures

CRTM

Sends Value

TPM

Execution Order

Building the Chain of Trust
The proof that a platform is a Trusted Platform

Step 0: Implants the Certificate

Step 1: Builds the Chain of Trust

Step 2: Send the proof (certified measures)

Step 3: Check the authenticity

Can I trust you?

Yes, I'm a Trusted Platform and there is the proof
Misconceptions about Trusted Computing

- A certain definition of trust
  - “If you want me to trust you, prove me that you always behave in the expected manner for the intended purpose”

- “Trusted Computing Platform does not mean a platform that is trustworthy” [Bruce Schneier]
  - Need criteria to evaluate in which case the platform can be trusted or not (and so a trustworthy platform)
But a solution hard to use in practice

- What is a “measure” in practice?
  - The hash of the binary code loaded
  - This hash will be different according to the configuration of the program or the OS
  - Not sustainable in practice (specially for OS)

- A potential solution: Virtualization
  - The base OS is a virtual machine player
  - For a specific purpose, the protected content will be used within a dedicated virtual machine
What I will try to address in this talk

- What is DRM? (technologically speaking)
- Why DRM technologies seem not work in practice?
- Is Trusted Computing the silver bullet?
- Why the use of DRM is so controversial?
- What is the future for DRM technologies?
The controversial usage of DRM

- DRM technologies aims at
  - *In theory*, protecting the confidentiality and integrity of sensitive information
  - *In practice* protecting copyrighted contents

- copyrights == sensitive information
- Using DRM implies inner restrictions that a legitimate end-user does not want (and should not want)
- But a good business model may lower these restrictions

> Example of Apple iTunes/iPod
TC and DRM: a solution for ...

- Protecting multimedia content or software
  - Some people say:
    “Annoying honest people and failing to prevent piracy at the end. That's what DRM is all about!”

- Locking and imposing a proprietary technology
  - Some people say:
    “My computer, my audio player, my mobile phone are locked and no longer under my control because of all of these companies seeking for monopoly and because Big Brother is watching me”
But it is also a solution for ....

- Preventing cheating on online videogames
- Ensuring fair routing and fair exchange in P2P based ad-hoc networks
- Ensuring the integrity of calculus in massive distributed Grid Computing
- Protecting from malicious code when I'm using banking application (in a certain limit)
The future of DRM systems

- Protecting user privacy
  - I would want to say
    “I want to control how all these web-sites are using my personal data. Why should I trust them?”

- Cooperation between federated information systems (E-DRM: Enterprise DRM)
  - I would want to say
    “I need to protect sensitive information that I'm sharing with business partners but without building a common dedicated infrastructure”
Conclusion

- I hope I (almost) convinced you that

  • DRM aims at protecting sensitive information which are out of the scope of a trusted domain
  • DRM systems can be (and must be) open and can be implemented using open-source design
  • DRM systems can be (and must be) interoperable
  • DRM can be user-centric and user-controlled
  • Existing DRM technologies are not mature
DRM Usage Control

- DRM have a so bad reputation that it should not be named DRM any more

  - I proposed “Technology-Which-Must-Not-Be-Named”

  - Some people in the research community call it “usage control”
Bibliography on DRM

- [Book] Digital Rights Management Technological, Economic, Legal and Political Aspects
  - E. Becker, W. Buhse, D. Günnewig, N. Rump

- [Book] Digital Rights Management, Business and Technologies
  - Bill Rosenblatt, Bill Trippe and Stephen Mooney

- [Website] DRM Watch : The Leading Resource For Digital Rights Management
  - Edited by Bill Rossenblatt
    www.drmwatch.com
Bibliography on Trusted Computing

  - Siani Pearson (HP Labs)

- [Website] The trusted Computing Group
  - www.trustedcomputinggroup.org

- [Article] `Trusted Computing' Frequently Asked Questions
  - Ross Anderson (University of Cambridge)
    www.cl.cam.ac.uk/~rja14/tcpa-faq.html

- [Article] Clarifying Misinformation on TCPA
  - David Safford (IBM Research)