Abstract: This paper presents a distributed digital rights management platform (DRMp) based on a web-oriented watermarking protocol. The platform enables web service providers (SPs) to dynamically supply copyright protection services on behalf of web content providers (CPs) in a security context. Thus, CPs exploiting the platform can take advantage of copyright protection services supplied by SPs acting as trusted third parties (TTPs) without having to directly implement them. On the other hand, SPs can follow the proposed design approach to implement protection procedures that can be easily exploited by CPs without imposing a tight coupling among the involved web entities.

Key Words: Digital rights management, watermarking protocol, service framework

1 Introduction

Digital watermarking [3] has gained popularity as a main technology to set up reliable DRMs, which are software platforms purposely developed to implement the copyright protection of digital contents distributed on the Internet [1]. However, some documented and relevant problems, such as the “customer’s right problem” [11] and the “unbinding problem” [10], can arise if DRMs are directly managed by CPs that autonomously implement the watermarking procedures or insert watermarks that are not properly bound to the distributed contents or to the transactions by which the contents are purchased. In fact, such DRMs end up implementing protection processes that do not correctly take into account the buyers’ rights, since the watermark is autonomously inserted by CPs, i.e. the copyright owners, without any control.

A possible solution to the above problems consists in developing DRMs that can exploit TTPs able to implement watermarking and dispute resolution protocols by which CPs can both obtain an adequate protection of their contents distributed on the Internet and determine the identity of guilty buyers with undeniable evidence [10]. However, this means that DRMs have to be characterized by distributed software architectures that enable distinct web entities to dynamically cooperate in a secure context without imposing a tight coupling among them.

This paper presents a distributed DRMp based on a web-oriented watermarking protocol able to solve the problems described above, such as the “customer’s right problem” and the “unbinding problem” [5]. The platform enables SPs to dynamically supply copyright protection services based on watermarking technologies on behalf of CPs in a security context. Thus, CPs that exploit the proposed platform can take advantage of copyright protection systems supplied by SPs acting as TTPs without having to directly implement them. On the other hand, SPs can follow the proposed design approach, based on web services [2] and other specific programming techniques [6, 15], to implement copyright protection procedures, such as watermarking procedures, which can be easily integrated into the proposed DRMp and exploited by CPs [4].

The outline of the paper is as follows. Section 2 describes the watermarking protocol assumed as service model by the DRMp. Section 3 presents the architecture of the DRMp adopting the proposed protocol. Section 4 describes the main implementation details of the platform. In Section 5 a brief conclusion is available.

2 The Watermarking Protocol

The proposed DRMp implements a copyright protection process that exploits a watermarking protocol based on four web entities: the buyer (B), the content...
provider or seller (CP), the service provider (SP), and
the trusted watermark certification authority (WCA).
The protocol, which is widely discussed in [5],
• solves both the “customer’s right problem” and
the “unbinding problem”;  
• keeps B anonymous during the purchase web
transactions;
• enables users who are not provided with digital
certificates issued by Certification Authorities to
purchase protected digital contents;
• does not require B to be able to autonomously
perform security actions that are not automatically
implemented by common web browsers;
• does not require a double watermark insertion,
thus both avoiding to discredit the embedded
protection and a possible degradation of the final
quality of the distributed contents.

In particular, the protocol assumes that B is identified
by means of a code associated to his/her credit card,
and forces WCAs to be directly involved in the com-
mercial transactions that take place whenever B wants
to buy a digital content distributed by CP [5].

2.1 The Protection Protocol

Figure 1 shows the scheme of the interactions taking
place during a web transaction by which B purchases a
copy of the digital content X distributed by CP. In par-
cular, the notation N.n denotes the message dispatch
from the entity N, at the step n of the protocol. Mes-
sages are exchanged through SSL connections charac-
terized by different authentication schemes negoti-ated
at the connection start-up. Furthermore, the notation
E\_\textit{entity}(data) specifies a ciphered token whose data
are encrypted with the entity’s secret key, whereas
the notation Eph\_\textit{entity}(data) specifies a ciphered to-
ken whose data are encrypted by exploiting a cryp-
tosystem that is “privacy homomorphic” with respect
to the watermark insertion [10].

B visits the CP’s web site and, after having chosen X, negotiates with CP to set up a common agree-
ment AGR, which states the rights and obligations of
both parties as well as specifies the digital content of
interest. During this negotiation phase B may have
free access to the CP’s web site or use a registered
pseudonym, thus keeping his/her identity unexposed.

Then, B communicates his/her will of buy-
ing X to CP by sending the negotiated AGR
(B.1). Upon receiving AGR, CP generates the token
E\_\textit{CP}(TID, XD, AGR, T\_CP) that includes: (1) the
transaction identifier TID, which is a code used by

![Figure 1: The protection protocol.](image-url)
the B’s credit card, WCA generates the token $E_{WCA}(UD, CC, T ID, XD, AGR, CP, T_{WCA}, np)$, which includes, among the others, $T_{WCA}$ and $np$: the former is a timestamp that makes the token’s freshness assessable, whereas the latter is a flag specifying that the B’s credit card has not been charged yet. The token is sent to B as a “temporary” purchase certificate, whereas is sent to CP as a “temporary” sale certificate (WCA.4). In addition, WCA also returns some other information to B and CP in order to enable them to make a check on the current transaction. In fact, B and CP cannot access the information contained in the temporary, ciphered certificates, and this prevents them from maliciously modifying the certificates. However, they can verify the plaintext data exchanged in WCA.4, and so they can abort the transaction if data turn out to be invalid.

After verifying WCA.4, CP sends WCA the watermarking request OP5, which includes $X$. After receiving $X$, WCA encrypts it by exploiting a cryptosystem that is “privacy homomorphic” with respect to the subsequent watermark insertion [10]. WCA also generates the fingerprinting code $W U$, which will have to be embedded in $X$ to identify B. To this end, in order to always associate the same code to the B’s credit card, whereas $\tau$ varies under different digital contents and timestamps, $W U$ is the concatenation of $\mu$ and $\tau$. In addition, WCA exploits a further function $\epsilon$ to generate an extended version of $W U$, denoted as $W UL$, whose ciphered form will be used by CP to refer to the corresponding sale certificate. Thus, after the correct download of $X$, WCA can charge the B’s credit card and generate the token $E_{WCA}(UD, CC, T ID, XD, AGR, CP, SP, WU, WUL, T_{def}^{WCA})$, which represents the definitive version of the purchase and sale certificates to be sent to B and CP respectively (WCA.11 and WCA.12). In addition, WCA also sends CP $E_{WCA}(WUL)$, which will be used by CP to refer to the corresponding sale certificate, $SP$ and $T ID$ in its databases.

2.2 The Identification and Arbitration Protocol

This protocol, shown in Figure 2, can be conducted whenever a pirated copy $X'$ of a protected digital content $X$ owned or distributed by CP is found in the market. To this end, CP can ask WCA for starting the protocol by sending it $X'$ and the reference to the SP exploited to protect the original content $X$ (CP.1).

WCA sends SP the ciphered content $E_{ph_{WCA}}(X')$ (WCA.2), and this prevents SP from getting access to the final versions of the protected contents distributed by CP. Then, SP extracts the embedded watermark from $E_{ph_{WCA}}(X')$ and communicates the fingerprinting code $E_{WCA}(WUL)$ to WCA (SP.3). WCA takes charge of generating the

<table>
<thead>
<tr>
<th>Legend</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL with authentication of the receiver</td>
<td></td>
</tr>
<tr>
<td>SSL with authentication of the sender</td>
<td></td>
</tr>
<tr>
<td>SSL with authentication of both sender and receiver</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 2: The identification and arbitration protocol.](attachment:image.png)
extended version of $WU'$, denoted as $WUL'$. Then, WCA sends CP $E_{WCA}(WUL')$ (WCA.4).

CP accesses its databases and uses $E_{WCA}(WUL')$ to search them for a match. When a match is found, CP retrieves the sale certificate $E_{WCA}(UD, CC, TID, XD, AGR, CP, SP, WU', WUL', T_{def}^{WCA})$ as well as further information associated to $E_{WCA}(WUL')$, and requires the buyer identification by sending these data to WCA (CP.5).

WCA decrypts the sale certificate and verifies all data received from CP. If all data turn out to be correct, the identity of the buyer is revealed, and WCA can adjudicate him/her to be guilty, thus closing the case. Otherwise, the protocol ends without exposing any identity.

3 The DRM Platform

The proposed platform, whose architecture is shown in Figure 3, consists of three main parts. The first part groups the web servers of a CP, and represents the “front-end” tier of the platform seen by user clients. The second part is represented by the WCA, and is the “middle” tier of the platform. The third part represents the “back-end” tier of the platform, and is basically composed of the web services [2] implemented by SPs and made available to the platform by means of a purposely designed “service framework” [4].

The platform has been developed according both to the WS-Security (Web Services Security) specifications, which define a set of SOAP header extensions for end-to-end SOAP messaging security, and to a “federated model” for the identity management of the web services’ operators [14].

The front-end tier of the platform is represented by the CP’s web applications, whose logic is assumed to be adapted to the distributed architecture of the platform. As a consequence, a CP wanting to exploit the proposed DRMp has to modify its applications in order to integrate them in the new interaction scheme involving the WCA.
This section focuses on the main implementation details concerning with the back-end tier of the proposed DRMp, which has been designed as a service framework (see Figure 4). In particular, to support flexibility and to make the service integration easy and dynamical, the service framework has been developed in Java. It consists of two main sections: the former comprises the internal services of the framework, whereas the latter comprises the web services supplied by SPs and dynamically loaded in the framework.

The internal services have been implemented in several software components whose interfaces have been defined according to a “component framework” approach [15]. In particular, each component of the framework implements a set of behavior rules that define the execution environment and the skeleton of the framework, i.e. the set of minimal services needed to enable SPs to dynamically integrate their web services in the DRMp. Furthermore, to facilitate the integration among the framework components and the external web services, four specific design patterns [6] have been used: “inversion of control”, “separation of concerns”, “proxy” and “adapter”.

The first pattern enables the framework to control the execution of components and to coordinate all the events produced by transactions. The second pattern allows the internal services of the framework to be implemented as collections of well-defined, reusable, easily understandable and independent components [8]. The third pattern is exploited to create “surrogate” objects for the external web services grouped in the framework in order to control the access to them. Finally, the fourth pattern is used to convert the specific interfaces of the external web services into simplified interfaces that can be invoked by the internal components of the framework. In fact, the proxy and adapter patterns make it possible to dynamically load the external web services into the framework without imposing constraints on their public interfaces.

Figure 6 shows the scheme of the interactions, labelled by numbers, taking place when a service request is issued to the framework. In particular, the request is received by the unified access point, which is a web service that acts as a service dispatcher toward all the external web services dynamically loaded into the framework and made available to CPs. Therefore, when the unified access point receives a service request from outside (1), it contacts the service manager (2), which has the main task of managing the request.

The service manager accesses the database manager (3), which determines the characteristics, i.e. the “profile” (4), that a web service provided by the framework should have in order to match the requirements derived from the external request issued. To this end, the database manager manages a repository (see Figure 7) which specifies some main parameters able to characterize the web services’ implementations. Such parameters can specify, for exam-
able a dynamical loading of external web services into the framework without imposing constraints on the web services’ interfaces, the service engines exploit the adapter objects, which are dynamically generated and loaded whenever an external web service is discovered and loaded into the framework (11a,11b,11c) [7, 9].

If the required service is not available within the framework, the service directory has to start the discovery service supplied by specific framework components implemented by exploiting the JINI framework [13]. The discovery service is activated via the service controller (6,7). It can search UDDI registries to discover the required service (8) and can return the result to the service directory (9,10,11), which communicates it to the service manager (12). To this end, it is worth noting that the discovery service is designed to exploit an extended UDDI model by which it is possible to find for a web service with the requested protection and QoS characteristics [12]. Therefore, the discovery service can search UDDI registries to discover a web service with the characteristics imposed by the service profile previously determined. Once the web service is found, the service QoS certifier can verify the service characteristics claimed by its SP (9a,9b,9c): if the characteristics map with the requirements specified by the service profile, the web service is loaded into the framework and the profile is stored in the internal repository by the database manager.

When this phase ends, the new external web ser-
vice results in being loaded into the framework, and its adapter object is generated and linked to the corresponding service engine by the service controller (11a,11b,11c). Then, the service manager can generate the proxy object (13a,13b) that will receive the service requests issued to the framework and coming from outside (20).

5 Conclusions

This paper presents a distributed DRM platform based on a web-oriented watermarking protocol and implemented by using web services and specific programming technologies. The platform enables SPs to dynamically supply copyright protection services on behalf of CPs in a security context, whereas CPs exploiting the platform can take advantage of a copyright protection system without having to directly implement it. In fact, the proposed design approach, based on a service framework, makes the DRM platform modular and easily extensible, since the copyright protection services developed by SPs can be easily and dynamically exploited by CPs without requiring a tight coupling among the involved web entities.

References:


