Digital Rights Management using a Mobile Phone

Imad Abbadi[*]
Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
I.Abbadi@rhul.ac.uk

Chris Mitchell
Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
C.Mitchell@rhul.ac.uk

ABSTRACT

This paper focuses on the problem of preventing illegal copying of digital assets without jeopardising the right of legitimate licence holders to transfer content between their own devices, which make up a domain. Our novel idea involves the use of a domain-specific mobile phone and the mobile phone network operator to authenticate the domain owner before devices can join a domain. This binds devices in a domain to a single owner, that, in turn, enables the binding of domain licences to the domain owner. In addition, the way in which we control domain membership, and the use of the domain-specific mobile phone that enables a domain owner to add devices wherever he/she is physically present, ensures that devices joining the domain are in physical proximity to the mobile phone, preventing illicit content proliferation.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection-copyright protection; D.4.6 [**Operating Systems**]: Security and Protection-trusted computing

General Terms

Design, Security

Keywords

DRM, Copyright Protection, Authorised Domain Management, 3GPP GAA, Access Control, Trusted Computing

1. INTRODUCTION

Unauthorised reproduction of digital assets is not a new issue. Concerns about protecting digital assets are raised every time reproduction tools such as tape recorders or photocopiers become available to consumers. In the past, however, content piracy was limited to distribution via physical media. The digitisation of information, the development of telecommunication technologies such as broadband and mobile networks, and the spread of the Internet have led to a huge rise in digital content piracy, as content can be shared and transferred instantly with no loss of quality.

1.1 Authorised Domains

Most current DRM systems recognise that consumers have more than one device, which they would like to use to access their content without requiring multiple licences. Many DRM system providers (see, for example, section 8) have incorporated the concept of an *authorised domain* into their content protection solutions. Such a domain is a collection of devices belonging to a single owner, within which digital assets can be freely moved.

Abbadi [7] analysed the authorised domain concept, and divided the devices in a domain into two categories: *roots* and *leaves*. The domain root (unique per domain) represents a licensed content holder. The leaves in a domain receive content (and means to access the content) from the domain root. The content piracy problem can then be divided into two sub-problems. The first is *Root Distribution*, where the root of the domain illegally distributes content and any associated passwords to an unlimited number of users. For example, the content holder could illegally distribute content and its associated password to devices outside the domain, i.e. devices which are not leaves in this domain. The second is *Leaf Distribution*, where an individual leaf in a domain illegally redistributes content to devices outside the domain, as if it is the licensed content holder. For example, after receiving content and any associated passwords/keys, a leaf device could illegally re-distribute the content and passwords/keys to user$_1$, which in turn re-distributes them to user$_2$, and so on (where user$_1$ and user$_2$ are not leaves in this domain).

A fundamental DRM requirement is to restrict Root Distribution to legitimate devices owned by the domain owner, and to completely prevent Leaf Distribution. In addition, a DRM system should satisfy other requirements as discussed in [7], such as: Flexible Rights Structure, Ease of Use, Content Mobility, Performance, Ease of Recovery, and Interoperability. As discussed below, the scheme proposed here satisfies all these requirements.

Other Authorised Domains management solutions for DRM (see, for example, section 8) typically attempt to address these problems by using a counter to control the number

[*]The author is a senior member at *Groups Technology Services, Associated Newspapers – UK*, which has partially funded this work.

of devices that can simultaneously access domain content. However such counter-based mechanisms have significant security and usability limitations. For example, in many schemes (see, for example, section 8) devices can abuse the system by joining and then leaving multiple domains to illegally use their content. Moreover, in many schemes, increasing the domain size limit requires re-initialising and reconfiguring the domain, and, in some schemes, domains cannot be expanded. In addition, there is no binding between the domain key (content protection key) and the domain owner; i.e. the Leaf Distribution problem arises. Also, all these solutions have additional problems in addressing other fundamental DRM requirements, such as content backup and recovery, ease of use, performance, etc. These issues are discussed in section 8.

1.2 Our novel solution

The system proposed here works by creating a domain that contains all devices belong to a single owner, within which digital assets can be freely moved. Each domain has a unique secret key shared by all domain devices that is used to encrypt all digital content encryption keys in the domain. The domain-specific key is generated automatically by a mobile phone owned by the domain owner, and is unavailable in the clear, even to the domain owner. These measures stop the domain owner from disseminating the domain-specific key. Consequently, distributing encrypted content to a device outside the domain will not enable access to the content unless this device joins the domain to receive the domain-specific key.

Before the mobile phone transfers the domain-specific key to devices joining the domain, it authenticates the domain owner using a shared secret, e.g. a PIN, and the mobile phone and the mobile network operator are then mutually authenticated based on the 3rd Generation Partnership Project (3GPP[1]) Authentication and Key Agreement protocol. This mutual authentication establishes session keys shared by the mobile phone and the mobile network operator. These session keys are later fetched by a Network Application Function, which maintains and manages consumer domains. The Network Application Function is provided by the mobile network operator using the General Authentication Architecture mechanism. Next, the mobile phone ensures a joining device is in close proximity to itself to prevent devices joining a domain via the Internet.

If the above procedure succeeds, the mobile phone releases the domain key to the joining device. The domain key is then securely stored by the joining device, is not available in the clear even to the domain owner, and cannot be copied from this device to other devices. Only the domain-specific mobile phone can release the domain key to other devices after authenticating the domain owner. This binds the domain key to the domain owner.

In addition, each domain has two associated limits, maintained and controlled by the mobile phone network operator, one to control the number of devices that can simultaneously access domain content, and the other to control the total number of devices that can join a domain. The latter limit is designed to stop domain owners abusing the system by allowing multiple devices to join and then leave their domains.

[1]http://www.3gpp.org

These measures address both the Root and the Leaf Distribution problems, as discussed in section 5. However, our proposed scheme does not stop legitimate controlled content sharing, and the physical proximity check does not stop legitimate downloading of digital content from a remote location, as outlined in section 4.4. In addition, the general approach of this scheme could also be useful in various other applications requiring strong authentication, because it strongly binds a domain to its owner. Our solution can be implemented irrespective of device type, and only requires the user platforms on which content is to be stored and used, to have functionality that can be trusted to perform the proposed DRM scheme correctly.

1.3 Why mobile phones?

The main reason for choosing mobile phones as domain controllers is that mobile phones are personalised, portable (enabling a domain owner to add devices wherever he/she is physically present) and ubiquitous. In addition, the existing mobile network infrastructure enables the authentication of subscribers, and provides a Network Application Function service as part of the General Authentication Architecture that is used to manage consumer domains. The participating network operators might want to charge for their service; however, network operator support is only required when creating a domain and when devices join the domain, and hence the cost impact of a modest charge should be manageable. The extra costs in implementing the solution could be covered from the expected reduction in piracy.

1.4 Organisation of the paper

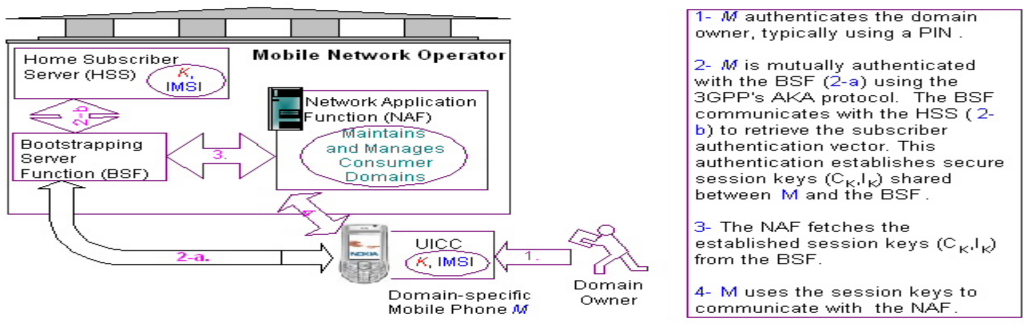
This paper is organised as follows. Section 2 describes the 3GPP General Authentication Architecture. Sections 3 and 4 describe the proposed solution and the process workflow. Section 5 discusses controlling domain membership. Section 6 analyses the system security requirements, threats, and services. Section 7 describes how the platform trust requirements underlying the proposed solution can be met using functionality in the trusted computing group (TCG[2]) specifications. Section 8 describes related work, and section 9 provides conclusions.

2. GENERAL AUTHENTICATION ARCHITECTURE

The scheme we propose relies on the mobile network operator supporting the 3GPP General Authentication Architecture (GAA) [5, 6]. Specifically we use the General Bootstrapping Architecture mechanism that builds upon the secret key K shared by the UMTS IC Card (UICC) in the mobile phone and the mobile network operator's Home Subscriber Server (HSS). The General Bootstrapping Architecture makes use of two network elements, known as the Bootstrapping Server Function (BSF) and the Network Application Function (NAF). Figure 1 summarises the GAA workflow.

The Bootstrapping Server Function has an interface with the Home Subscriber Server and with the Network Application Function, and is part of the mobile network operator. The Bootstrapping Server Function acts as an intermediary between the mobile phone, the Home Subscriber Server and the Network Application Function. The mobile phone and

[2]http://www.trustedcomputinggroup.org


Mobile Network Operator
Home Subscriber Server (HSS)
K, IMSI
2-b
Bootstrapping Server Function (BSF)
3.
Network Application Function (NAF)
Maintains and Manages Consumer Domains
4.
2-a.
UICC
K, IMSI
1.
Domain-specific Mobile Phone M
Domain Owner

1- M authenticates the domain owner, typically using a PIN .

2- M is mutually authenticated with the BSF (2-a) using the 3GPP's AKA protocol. The BSF communicates with the HSS (2-b) to retrieve the subscriber authentication vector. This authentication establishes secure session keys (C_K,I_K) shared between M and the BSF .

3- The NAF fetches the established session keys (C_K,I_K) from the BSF.

4- M uses the session keys to communicate with the NAF .

Figure 1: The General Authentication Architecture

the Bootstrapping Server Function are mutually authenticated using the 3GPP Authentication and Key Agreement (AKA) protocol [5, 20]. This mutual authentication establishes secure session keys that are later fetched by the Network Application Function from the Bootstrapping Server Function, and then used to secure communications between the Network Application Function and the mobile phone. Communications between the Network Application Function and the Bootstrapping Server Function, and between the Bootstrapping Server Function and the Home Subscriber Server, are beyond the scope of this paper; details can be found in [5].

The 3GPP Authentication and Key Agreement protocol works as follows. The mobile phone sends an authentication request to the Bootstrapping Server Function, which then contacts the Home Subscriber Server and retrieves the subscriber security settings and an authentication vector RAND||AUTN||XRES||C_K||I_K. This vector is calculated using the mobile-specific secret key K, a random challenge RAND, and a set of functions shared by Home Subscriber Server and the mobile phone. XRES is the "expected response", used later to authenticate the mobile phone, C_K is the session Cipher Key, I_K is the session Integrity Key, and AUTN is equal to SQN$\oplus$AK||AMF||MAC, where SQN is a sequence number, AK is the Anonymity Key, AMF is the Authentication Management Field, and MAC is a Message Authentication Code for the string SQN||RAND||AMF, computed using the key K.

Next, the Bootstrapping Server Function forwards AUTN and the challenge (RAND) to the mobile phone, which uses its UICC to calculate I_K, C_K, RES, and AK, and recover AMF and MAC from AUTN. The UICC then derives SQN from the calculated AK and the received SQN$\oplus$AK, which is used to verify the freshness of the received message. The UICC calculates the MAC, and then compares it with the MAC recovered from AUTN. If they are equal, then the mobile phone has successfully checked the validity of the network. Subsequently, the mobile phone sends an Authentication and Key Agreement Digest, calculated using RES, to the Bootstrapping Server Function. The Bootstrapping Server Function recomputes the Digest, using XRES, and deems the mobile phone authenticated if the recomputed value equals the received value.

The Bootstrapping Server Function and the mobile phone use the session keys C_K and I_K to establish a secure channel that protects messages exchanged between them, and between the mobile phone and the Network Application Function. The secure channel is implemented using a stream cipher and a MAC function. For further details of these services see, for example, [3].

3. SYSTEM MODEL

The model for our scheme involves five main types of entity: Domain Devices, Mobile Phones, Mobile Network Operators, Rights Issuers, and Content Distributors.

3.1 Domain Devices

We require that Domain Devices, including the domain-specific mobile phone, are trusted platforms (TPs). These are computing platforms with the property that their state can be remotely tested, and which can be trusted to store security-sensitive data in ways testable by a remote party. A Domain Device could be a PC, laptop, PDA, mobile device, etc. Each Domain Device is assumed to possess a DRM agent, which must be trusted to perform the DRM scheme correctly. Each Domain Device can verify that the DRM agent is running correctly in another device. In addition, each Domain Device is assumed to possess an asymmetric encryption key pair. The corresponding private decryption key is bound to a particular environment configuration state. We assume that the DRM software agents that are authorised to read data encrypted using this key will not release the data outside the Domain Device; even the domain owner should not be able to retrieve these data in clear. The TP must provide a protected execution environment, in which applications run in isolation, free from being observed or compromised by other processes running in the same protected partition, or by software running in any insecure partition [9].

A TCG compliant platform meets all these requirements; see, for example, section 7. TCG compliant platforms are not expensive, and are currently available from a range of PC manufacturers, including Dell, Fujitsu, HP, Intel and Toshiba [10].

3.2 Mobile Phones

Our DRM scheme requires that a domain-specific mobile phone owned by the domain owner, is registered by the mobile network operator and is equipped with a special application. We also suppose that the UICC in the phone is capable of supporting the DRM scheme. The domain-specific mobile phone should be compatible with the 3GPP specifications [5], and is responsible for: authenticating the domain owner using a shared secret, e.g. a PIN, which is stored securely in the UICC; mutually authenticating itself with the mobile network operator; creating and maintaining a domain unique secret key K_D that is used to encrypt content encryption keys; and authorising devices to join its domain by ensuring that they in physical proximity to itself, and that their processing environment is trusted. The key K_D is not available in the clear, even to the domain owner, and only the mobile phone is able to copy this key to a device joining the domain.

Each mobile phone maintains two sequentially incremented domain counters, both of which are initially set to one. The first counter, C_t, represents the total number of devices that have joined the domain. The second counter, C_p, represents the number of devices currently present within a domain. Both counters have domain-specific limits, denoted by L_t and L_p (for C_t and C_p, respectively). These limits help control domain membership, and are set and maintained by the Network Application Function.

For the purposes of our scheme we assume the Network Application Function provides the following functions: initialising and maintaining domain-specific limits, i.e. L_t and L_p, and backup and recovery of the domain-specific secrets, K_D, C_p and C_t. The phone obtains L_t and L_p from the Network Application Function. Both limits can be increased by the mobile network operator. Consumers could be charged more for higher maximum values.

3.3 Content Distributor and Rights Issuer

The Content Distributor distributes items of protected digital content to consumers. Each item of protected content is encrypted using a content-specific secret key (K_T), which is stored inside an associated Rights Object (RO), together with other rules applying to the content. The Rights Issuer is in charge of issuing the RO. The Domain Device that renders protected digital content enforces the rules inside the associated RO.

4. PROCESS WORKFLOW

The workflow of the proposed system is divided into four main stages: domain establishment, joining devices, exchanging content, and content backup and recovery. Before performing any of these processes, an initialisation procedure must be performed between the mobile network operator and the mobile phone. We now describe each of these processes.

4.1 Initialisation procedure

The initialisation of a mobile phone by a mobile network operator involves the authentication of the phone and the establishment of a secure session between the phone and the Network Application Function. Before creating a domain, or adding or removing devices from a domain, the domain-specific mobile phone M authenticates the domain owner by instructing the domain owner to provide a secret key, e.g. a PIN, shared by the domain owner and M. Once the domain owner has been authenticated, M and the Bootstrapping Server Function perform mutual authentication. As described in section 2, this process establishes session keys C_K and I_K, later fetched by the Network Application Function F from the Bootstrapping Server Function, and used to secure communications between F and M.

4.2 Domain Establishment

This phase applies when a consumer wishes to create a domain (and simultaneously add the domain-specific mobile phone M to the domain). We assume the initialisation procedure described in section 4.1 has already been performed. M securely generates the domain-specific secret key K_D using a random number generator, and initialises counters C_p and C_t to one. M then sends a Create_Domain request to the Network Application Function F, via the pre-established secure session, to establish a domain. The request has the form:

(1) $M \rightarrow F$: $e_{C_K}(K_D||IMSI)||m_{I_K}(e_{C_K}(K_D||IMSI))$

where $e_{C_K}(Y)$ denotes the symmetric encryption of data Y using key C_K, $m_{I_K}(Y)$ denotes a MAC computed on data Y using key I_K, and IMSI is the International Mobile Subscriber Identity that is used to uniquely identify the domain. When a user changes his/her IMSI, e.g. by changing network operator, the new network operator obtains the user domain information from the old network operator.

Next, F verifies the integrity of the received message and decrypts it, and then initialises the domain, i.e. sets the domain limits L_p and L_t, securely associates them with K_D and IMSI, and stores them in its protected storage. F then transfer these limits to M in the following message:

(2) $F \rightarrow M$: $e_{C_K}(L_p||L_t)||m_{I_K}(e_{C_K}(L_p||L_t))$.

M verifies the integrity of the received message and decrypts it, and then securely associates L_p and L_t with K_D, C_p, and C_t, and securely stores them in its protected storage.

The domain key K_D is generated inside the secure environment of M, and is then securely stored by M. Given the assumptions in section 3.1, this means that it is not available in the clear even to the domain owner, and cannot be copied between Domain Devices. In addition, it is unique per-domain, shared amongst all domain devices, and does not change during the life of the domain. K_D is used to encrypt ROs, each of which contains a content-specific key K_T that is used to encrypt the associated item of content.

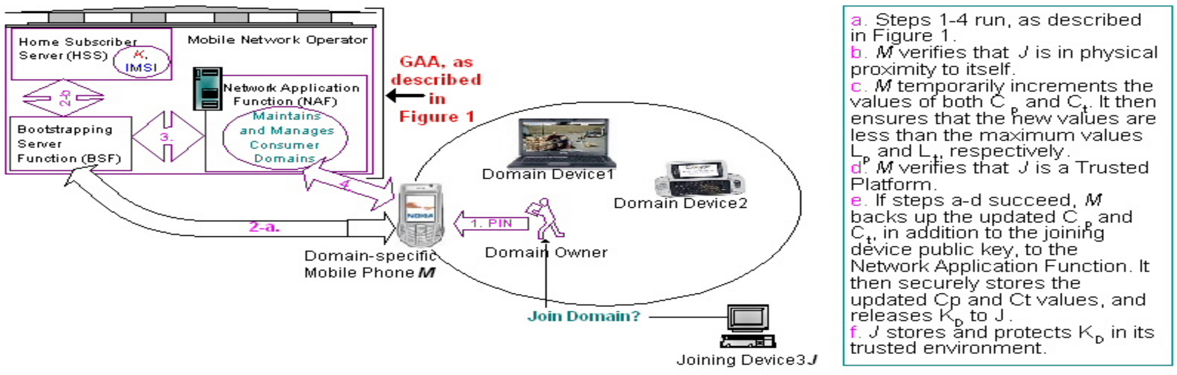
4.3 Joining and Leaving Domains

This phase applies when a device joins or leaves a domain; we assume that the procedures described in sections 4.1 and 4.2 have already been performed. In order for a device J to join a domain, it must communicate with the domain-specific mobile phone M, as shown in Figure 2. J first sends a Join_Domain request to M. The request has the form:

(1) $J \rightarrow M$: $P_J||S_J\ ||N_1||\text{Cert}_{A_J}||\text{Sign}_J(P_J||S_J||N_1)$

where P_J is the public encryption key of J, S_J is the execution status of the DRM agent on J (the exact nature of S_J is implementation-dependent, see, for example, section 7), N_1 is a nonce, Cert_{A_J} is a certificate signed by the CA for the public key A_J, where A_J is the signature verification key of J, and $\text{Sign}_J(Y)$ denotes a signature on data Y created using the private signing key of J (note that we use P_X, S_X, Cert_{A_X}, A_X and $\text{Sign}_X(Y)$ throughout to denote the corresponding objects associated with entity X).

Next, M checks that J is in physical proximity to itself, e.g. by using the Near Field Communication (NFC) protocol or measuring the Round-Trip Time (RTT) between M and J, see, for example, [11, 12, 15]. M verifies Cert_{A_J}, extracts A_J, and checks that it has not been revoked, e.g. by querying an Online Certificate Status Protocol (OCSP) service, [19].


Home Subscriber Server (HSS)
K, IMSI
Mobile Network Operator
2-b
Network Application Function (NAF)
Maintains and Manages Consumer Domains
Bootstrapping Server Function (BSF)
3.
4.
2-a.
GAA, as described in Figure 1
Domain Device1
Domain Device2
1. PIN
Domain-specific Mobile Phone *M*
Domain Owner
Join Domain?
Joining Device3 *J*

a. Steps 1-4 run, as described in Figure 1.
b. *M* verifies that *J* is in physical proximity to itself.
c. *M* temporarily increments the values of both C_p and C_t. It then ensures that the new values are less than the maximum values L_p and L_t, respectively.
d. *M* verifies that *J* is a Trusted Platform.
e. If steps a-d succeed, *M* backs up the updated C_p and C_t, in addition to the joining device public key, to the Network Application Function. It then securely stores the updated Cp and Ct values, and releases K_D to J.
f. *J* stores and protects K_D in its trusted environment.

Figure 2: DRM system workflow

M then verifies J's signature using A_J, and verifies that the DRM agent is running correctly in J by checking the value of S_J. How this verification occurs is implementation-dependent, see, for example, section 7. M then generates a nonce N_2, and sends the following message to J:

(2) $M \rightarrow J$: $\mathrm{ID}_J||N_1||N_2||S_M||Cert_{A_M}||\mathrm{Sign}_M(\mathrm{ID}_J||N_1||N_2||S_M)$

where ID_J is the identifier for J included in Cert_{A_J}. Note that we use ID_X throughout to denote the identifier of entity X. J verifies $Cert_{A_M}$, extracts A_M from $Cert_{A_M}$ and verifies that it has not been revoked, e.g. by querying an OCSP service. J then verifies M's signature by using A_M, verifies ID_J to ensure that it is the intended recipient, and verifies message freshness by comparing N_1 with the value sent in (1). J then verifies that the DRM agent is running correctly in M by checking the value of S_M. As above, how this verification occurs is implementation-dependent. J then sends the following reply to M:

(3) $J \rightarrow M$: $N_2||\mathrm{ID}_M||\mathrm{Sign}_J(N_2||\mathrm{ID}_M)$

M verifies J's signature, verifies ID_M to ensure that it is the intended recipient, and checks message freshness by comparing N_2 with the value sent in (2). Steps 1–3 conform to the three-pass mutual authentication protocol described in [14]. M temporarily increments the values of both C_t and C_p. If they are greater than the maximum permitted value (L_t and L_p, respectively) held by M, the agent running on M exits with an appropriate error message. M now sends F the following backup request for the values of C_t, C_p and P_J:

(4) $M \rightarrow F$: $e_{C_K}(C_t||C_p||P_J||\mathrm{IMSI})||m_{I_K}(e_{C_K}(C_t||C_p||P_J||\mathrm{IMSI}))$

F verifies the integrity of the received message and decrypts it, and then securely associates the new values with the domain identified by IMSI. Subsequently, F sends an acknowledgment back to M as follows:

(5) $F \rightarrow M$: $e_{C_K}(\mathrm{Result}||C_t||C_p)||m_{I_K}(e_{C_K}(\mathrm{Result}||C_t||C_p))$

M verifies the integrity of the received message and decrypts it. M checks message freshness by comparing C_p and C_t with the values sent in (4). M then checks the value of Result. If it indicates success, M stores the values of C_t and C_p in its trusted storage, and sends the key K_D to J in the following message:

(6) $M \rightarrow J$: $E_{P_J}(K_D)||\mathrm{Sign}_M(E_{P_J}(K_D))$

where $E_{P_J}(K_D)$ denotes the asymmetric encryption of K_D using public key P_J, and where we assume that the asymmetric encryption primitive in use provides non-malleability, as described in [16]. Note that we use $E_{P_X}(Y)$ throughout to denote the asymmetric encryption of data Y using public key P_X. When J receives this message, it verifies the signature, and then decrypts $E_{P_J}(K_D)$. The key K_D is then securely stored by J, as described in section 4.2.

In order for a Domain Device to leave a domain, the domain owner follows a similar process, except that the physical proximity check is not required, the C_t value does not change, and the C_p value is decremented. Before a Domain Device leaves a domain, M authenticates and attests to the state of the leaving Domain Device in the same way as described above. This is to ensure that the leaving Domain Device can be trusted to delete K_D and the key used to protect K_D from its protected storage. If a Domain Device is hacked, the domain owner must inform the CA, which will then include the hacked Domain Device public key in its revocation list. M checks whether the Domain Device public key has been revoked, e.g. by querying an OCSP service, before decrementing C_p.

4.4 Exchanging Content

This phase covers the communication process between Content Distributors/Rights Issuers and Domain Devices, between devices in the same domain, and between devices in different domains. A variety of methods could be integrated into the scheme for downloading digital content C and the associated Rights Object R from a Content Distributor/Rights Issuer to a Domain Device V; see, for example, [21]. Describing these methods is outside the scope of this paper; however, it generally involves the following steps. V sends a Get_Content message to the Content Distributor to request C, as identified by id. The request includes payment details, e.g. credit card details, which are used to pay for access to C. The Content Distributor and V mutually authenticate each other, and the Content Distributor attests to the state of V, as described in section 4.3, steps 1–3. If these verifications succeed, the Content Distributor securely generates a content-specific secret key K_T, symmetrically encrypts C using the key K_T, and sends a Generate_RO request to the Rights Issuer, as follows:

(1) Content Distributor $\rightarrow$ Rights Issuer: $P_V||K_T||h(G_{K_T}(C))||id$

where we assume that a secure session has been pre-established between the Content Distributor and Rights Issuer, h denotes a globally agreed cryptographic hash-function, $G_{K_T}(C)$ denotes the symmetric encryption of C using key K_T, and where we assume that G provides authenticated encryption, i.e. provides data confidentiality, integrity, and origin authentication, as described in [17]. Note that we use $G_X(Y)$ throughout to denote the symmetric encryption of data Y using key X. The Rights Issuer generates the Rights Object R, incorporating the string:

$$R = \ (C\text{'s consumption rules } ||E_{P_V}(K_T)||h(G_{K_T}(C))||id)$$

where $h(G_{K_T}(C))$ is used to bind R to the content C. Subsequently, the Rights Issuer sends the following message to the Content Distributor:

(2) Rights Issuer $\rightarrow$ Content Distributor: $\text{Cert}_{I_{RI}}||R||\text{Sign}_{RI}(R)$

The Content Distributor then sends the following message to V:

(3) Content Distributor $\rightarrow V$: $\text{Cert}_{I_{RI}}||R||\text{Sign}_{RI}(R)||G_{K_T}(C)$

V verifies that R is bound to C by recomputing $h(G_{K_T}(C))$, verifies $\text{Cert}_{I_{RI}}$, extracts I_{RI}, checks its validity, e.g. by querying an OCSP service, and then checks the Rights Issuer's signature. Finally, V decrypts K_T using its private decryption key, re-encrypts it using K_D, and retains $R||\text{Sign}_V(R)||G_{K_T}(C)||G_{K_D}(K_T)$ in its storage.

Transferring content from a source device V_s to a destination device V_d in the same domain involves the following steps. In the description below we implicitly assume that V_d and V_s have a real-time communications link; if such connectivity is not available, then the same messages can be exchanged using a portable storage medium, e.g. a USB memory stick. We also assume that only V_s needs to query an OCSP service; however, this does not require all domain devices to be connected to the Internet whist exchanging content. For example, a domain could contain one or more devices with Internet access that are used to download content from a Content Distributor, to access an OCSP service, and for distributing the downloaded content to other devices that do not have a real-time communication link. V_d first sends the following Get_Content message to V_s:

(1) $V_d \rightarrow V_s$: $\text{Cert}_{I_{V_d}}||P_{V_d}||id||\ \text{Sign}_{V_d}(P_{V_d}||id)$

V_s verifies $\text{Cert}_{I_{V_d}}$, extracts I_{V_d}, checks its validity, e.g. by querying an OCSP service, and then checks V_d's signature. If the verifications succeed, V_s sends C, as identified by id, and the associated Rights Object, in the following message:

(2) $V_s \rightarrow V_d$: $G_{K_T}(C)||E_{P_{V_d}}(R||G_{K_D}(K_T))$

V_d now decrypts $E_{P_{V_d}}(R||G_{K_D}(K_T))$ using its private key, and verifies that C is bound to R, as described above. If the verification succeeds, V_d retains $G_{K_T}(C)||G_{K_D}(K_T)||R||\text{Sign}_{V_d}(R)$. Before using the content, K_D is used to decrypt K_T, which can be used to decrypt the content.

The main goal of our scheme is to stop content decryption keys that are stored inside Rights Objects from being transferred to devices in different domains. This is achieved, as described earlier, by encrypting each content-specific key K_T with the domain-specific key K_D. As explained in section 4.3, K_D and the key used to protect it are removed from a device when it leaves a domain, which prevents protected Rights Objects from being used by devices in multiple domains. However, this does not stop legitimate controlled content sharing; protected content can move between devices belonging to different domains. A consumer could, for example, use protected content by contacting the corresponding Rights Issuer, and downloading a trial Rights Object enabling him/her to temporarily use the protected content. If the consumer is interested, he/she could then buy a full usage licence, as explained above. This concept is known as ***super-distribution***, and has been proposed by OMA [21] as a means of allowing consumers to obtain digitally protected content from anywhere, and to use it with a restricted licence. This allows consumers to use content for a limited period, with lower quality, and/or limited features. When the consumer is happy with the protected content and decides to get a full licence, only then will he/she need to download the Rights Object, which is much smaller than the encrypted content.

4.5 Backup and Recovery Procedure

A DRM solution must be capable of recovering digital content in the event of system failure. For backup purposes, digital content encrypted using the domain key K_D can be stored in an offline medium, for example, a tape or CD-ROM. If the domain key K_D is lost and cannot be recovered, it follows that the domain content on the backup cannot be decrypted. Thus, backup provisions for K_D are needed. The domain-specific mobile phone M is required to communicate with the Network Application Function regularly, as described in section 4.3, and backup K_D, C_t and C_p.

If the mobile phone M cannot be recovered, for example, because it has been lost or stolen, the domain owner must inform the mobile network operator. The mobile network operator will blacklist M and disable M's UICC, and then issue a new UICC. The replacement mobile phone will need to contact the Network Application Function and restore the domain settings stored by the Network Application Function. The Network Application Function then releases K_D, C_p, C_t, L_p and L_t, to the new domain-specific mobile phone. Subsequently, other devices in the domain that cannot recover K_D must re-join the domain, as described in section 4.3 (before M increments the values of the domain counter it checks with the Network Application Function whether the joining device public key is already a member of the domain; if so it does not increment the counters).

5. CONTROLLING DOMAIN MEMBERSHIP

The domain-specific mobile phone M controls domain membership in the following way.

1. It limits the number of devices that can be in a domain, hence limiting the number of devices that can simultaneously access domain content.

2. It limits the total number of devices that can join a domain, which stops domain owners abusing the system by allowing multiple devices to join and then leave a domain.

3. It stops piracy using digital media such as the Internet, because, as described in section 3.2, the content protection key K_D is securely stored inside M, is not available in the clear, and can only be transferred from M to other devices after their physical proximity has

been checked; i.e. the physical location check, in conjunction with the use of counters, addresses the Root Distribution problem.

4. It imposes stringent restrictions on piracy using physical media. As the content protection key K_D is not available in the clear even to the domain owner, this prevents the domain owner from transferring this key to other users. In addition, as described earlier, M must be used to transfer this key to other devices, which can only occur after the domain owner has been authenticated to M, and mutual authentication has been performed between M and the mobile network operator; i.e. the scheme prevents Leaf Distribution.

Most other schemes focus primarily on point (1); see, for example, section 8. Our solution stops illicit content proliferation; the only way a domain owner could transfer the content protection key to another user's device is by transferring the encrypted domain content, the domain-specific mobile phone M, and the domain owner's authentication credential for M, e.g. a PIN. Whilst possible in principle, such a procedure is unlikely to be attractive to the domain owner. Such a process would also mean that the other user's device would become part of the domain controlled by M, which would mean that fewer of the domain owner's devices could be added to the domain. Most importantly, devices which have joined this domain using M would not be able to re-transfer the domain key (as described above, only M can transfer this key to other devices).

6. SECURITY ANALYSIS

We now consider the security threats, services, and mechanisms that apply to the storage, execution and transmission of K_D, C_t, C_p, L_t, L_p, and digital content. In addition, we address the security threats, services, and mechanisms between the mobile device and the Network Application Function; however, we do not address threats to the 3G security system. Security threats of this type are addressed elsewhere; see, for example, [1, 2, 4].

6.1 Security Threats

Security threats related to processing and storing the domain-specific values K_D, C_t, C_p, L_t and L_p in a mobile phone M are: (1) unauthorised manipulation of the domain-specific values during use in M; and (2) unauthorised manipulation of the domain-specific values whilst stored in M.

Security threats related to processing and storing K_D whilst in transit between a mobile phone M and a joining device J are: (3) unauthorised reading or updating of K_D whilst in transit; (4) M unwittingly sending K_D to a malacious entity; (5) J unwittingly receiving K_D from a malicious M; and (6) replay of communications between M and J.

Security threats related to transferring content C and Rights Object R between Domain Devices are: (7) unauthorised reading or alteration of C, and unauthorised alteration of R, while in transit; and (8) transfer of C and R to an unauthorised entity.

Security threats related to storing and executing C and R in Domain Devices are: (9) unauthorised reading or updating of C, and unauthorised updating of R, whilst stored in a Domain Device; and (10) unauthorised reading or updating of C, and unauthorised updating of R while being accessed on a Domain Device.

Security threats related to exchanging messages between between a mobile phone M and the Network Application Function F are: (11) unknowingly, M communicating with a malicious entity, or F communicating with a malicious entity; (12) replay of communication between M and F; and (13) unauthorised reading or updating of exchanged messages.

6.2 Security Services and Mechanisms

The security services required to counteract threats 1, 2, 4, 9, and 10 (listed above) can be provided using trusted platform functionality, as discussed in section 3.1. In section 7.2 we illustrate how such trusted platform functionality can be implemented using a platform confirming to the TCG specifications. Threats 3, 5–8 and 11–13, are addressed using standard cryptographic mechanisms. A direct mapping exists between the threats outlined above and the services and potential mechanisms outlined below:

1. *Confidentiality and integrity of the domain-specific values during execution on M.* Providing this service requires process isolation techniques, as discussed in section 3.1.

2. *Confidentiality and integrity of the domain-specific values whilst stored in M.* Providing this service requires protected storage, as discussed in section 3.1.

3. *Confidentiality and integrity of K_D whilst in transit.* This service is provided by the use of symmetric encryption and a MAC, see section 4.2, or asymmetric encryption and a digital signature, see section 4.3.

4. *Entity authentication of a joining device J to the domain-specific mobile phone M.* The provision of this service is implementation-dependent, and involves a protocol exchange between J and M; see, for example, section 7. It is initiated when M and J mutually authenticate each other — see section 4.3. This mutual authentication attests to the DRM agent execution status, i.e. S_J, and whether the platform is trusted, as discussed in section 3.1.

5. *K_D origin authentication.* The joining device J checks the origin of K_D by checking M's signature on the received encrypted value of K_D — see section 4.3.

6. *Prevention of replay of communications between M and a device.* This is provided by the inclusion of nonces in messages — see section 4.3.

7. *Confidentiality and integrity protection of content C, and and integrity protection of Rights Object R whilst in transit.* This service is provided by encrypting C using an authenticated encryption technique, encrypting K_T using asymmetric encryption technique that provides non-malleability, and signing R; see section 4.4.

8. *Entity authentication of the destination device V_d whilst transferring content C and Rights Object R.* The source device V_s validates V_d's public key to make sure it has not been revoked, and then encrypts R and the encrypted K_T using V_d's public key — see section 4.4.

V_s does not need to verify whether V_d is trusted, because the transferred K_T is protected using K_D, which is known only by devices in the same domain and is revealed only in a trusted environment, as discussed in section 3.1.

9. *Confidentiality and integrity of content C, and integrity of Rights Object R in Domain Devices.* The integrity of R is protected using a digital signature. C is encrypted using the secret key K_T that is itself encrypted using a secret key K_D. As described in sections 4.4, the symmetric encryption technique in use is assumed to provide authenticated encryption — see section 4.4. Also, the encryption key K_D is bound to the device's trusted environment, as discussed in section 3.1.

10. *Confidentiality and integrity of content, and integrity of Rights Object during execution* is provided exactly as discussed in (1) above.

11–13. These threats are counteracted by the use of the secure session established between M and F, as described in section 2; mutual authentication between M and F counteracts threat 11; threat 12 is counteracted by the inclusion of nonces — see section 4.3; and threat 13 is counteracted by the use of symmetric encryption and a MAC — see section 4.2 and 4.3.

7. IMPLEMENTING THE PROTOCOLS USING TCG

In this section we describe how a system conforming to the TCG specifications [24, 25, 26] can satisfy the requirements described in section 3.1. In addition we illustrate how the TCG-based design counteracts the security threats listed in section 6.1.

7.1 Trusted Platform Requirements

A TCG compliant platform provides the following features, which meet the requirements for a TP given in section 3.1.

1. The TP has a hardware component referred to as Trusted Platform Module (TPM) that is physically and cryptographically bound to the TP. It is a self-contained processing module with specialist security capabilities such as random number generation, asymmetric key generation, digital signing, encryption capabilities, hashing capabilities, an HMAC [18] engine, monotonic counters, as well as memory, non-volatile memory, power detection and I/O. Support for platform integrity measurement, recording and reporting is also provided. The TPM is typically implemented as a processing engine that is separate from the TP's main processing environment.

2. An asymmetric encryption Storage Root Key (SRK) pair is securely associated with each TPM. The SRK private key is statistically unique, and is created and stored inside the TPM. The public part of the SRK acts as the root for encrypting sub-tree key objects that are used as data or signing key objects. The private part of the SRK is used for decrypting sub-tree objects.

3. A certification authority, the privacy-CA, certifies public identity keys. The generated certificate binds an identity of the TPM to a public key used for the verification of digital signatures. Domain Devices request these certificates from the privacy-CA by providing their platform credentials. These credentials are generated by the Trusted Platform Module Entity (TPME), that vouches that a TPM is a genuine TPM, the Platform Entity (PE), that attests to the correct incorporation of a particular TPM into a platform, and the Conformance Entity (CE), that attests that the design of the TPM in that class of platform meets the TCG specifications, and that the way that the platform incorporates that type of TPM also meets the TCG specifications. The corresponding private identity key, that is protected by the TPM, is used as a signing key for entity authentication.

4. A TPM can generate two types of keys, known as migratable and non-migratable keys. Migratable keys can be transmitted to other TPs if authorised by both a selected trusted authority and the TPM owner. A non-migratable key is bound to the TP that created it. Data encrypted under non-migratable keys can leave the TP if and only if the software agent authorises the release of the data to other platforms. We assume that software agents that are authorised to read data encrypted using non-migratable keys will not release the data outside the TP boundaries.

5. The TP can perform integrity challenge and response exchanges with other TPs. One TP can verify the trustworthiness of the state of another TP by computing an expected set of trustworthy integrity metrics and comparing them with the current platform software state obtained from the integrity response, which we referred to as S_X in section 4. This enables a TP to verify that the DRM agent is running correctly on a remote platform.

6. The TP protects all secret keys required by Domain Devices by encrypting them using a non-migratable key, as described in point (4). The TP associates the current platform software state, which is stored in the Platform Configuration Registers (PCRs), with the non-migratable key, and then protects them using the SRK. Stored secrets are only released after the platform's PCRs have been compared with the values associated with the stored key. Reporting, storage, and retrieval are carried out by the TPM. Therefore, if a process relies on the use of secrets, it cannot operate unless it and its software environment are correct. The system assumes that if the operating system and application are as expected, then the integrity and secrecy of data is subsequently guaranteed.

7.2 Security Services and Mechanisms

This section describes how security requirements 1, 2, 4, 9, and 10, discussed in section 6, can be met using TCG functionality.

1. *Confidentiality and integrity of K_D, C_t, C_p, L_p and L_t during execution.* A challenger can verify that a platform is trusted by validating the platform integrity

metrics. The TP measures the integrity of software executed from platform start-up and stores the result in the platform's PCRs; this provides assurance to the challenger that the expected version of the OS, and of any other measured software, is running on the platform. The private key that is used to decrypt K_D, C_t, C_p, L_p and L_t will only be released to the DRM software agent if the PCR values are as expected. The system assumes that if the OS and the application are as expected, then the integrity and secrecy of data is subsequently guaranteed.

2. *Confidentiality and integrity of K_D, C_t, C_p, L_p and L_t whilst stored in Domain Devices.* The use of asymmetric encryption provides the confidentiality service. When stored in a Domain Device, K_D, C_t, C_p, L_p and L_t are encrypted using a non-migratable key that is bound to the Domain Device TPM, and only released when the Domain Device integrity metrics are in a state that matches the values stored with the key. Integrity is provided by inserting an authorisation value into the key object (under the assumption that the method of encryption used has appropriate properties) that is required in order to reveal the key. If the encrypted key object has been altered, the decrypted authorisation value will not match the stored authorisation value.

4. *Entity authentication of a device J to the domain-specific mobile phone M.* This service is achieved using the TCG challenge-response authentication protocol. This is initiated when M generates a nonce N_1 and sends it as a part of an integrity challenge to J. J's trusted platform agent (TPA) replies with an integrity response that includes M's identity, N_1, and J's integrity metrics, i.e. S_J, all signed by J's TPM. In addition, it provides the measured software logs from the Trusted Platform Measurement Store (TPMS), as well as certificates for the measured software. The software measurements and the certificates enable M to verify the current state of J [9]. M verifies the signature and checks that the necessary properties hold for the platform associated with the identity.

9. *Confidentiality and integrity of content C and integrity of Rights Object R in a Domain Device V.* R integrity is protected using a digital signature. C is encrypted using the secret key K_T that is itself encrypted using the secret key K_D. As described in section 4.4, the symmetric encryption technique in use is assumed to provide authenticated encryption. The key K_D is protected using a non-migratable key that is bound to V's TPM and access control information. The protected storage mechanism is used to ensure that the non-migratable key is only accessed when the device's execution environment state matches that associated with this key.

10. *Confidentiality and integrity of C, and integrity of Rights Object during execution* is provided exactly as discussed in (1) above.

8. RELATED WORK

We now briefly review some of the more widely discussed schemes for managing an *authorised domain* for DRM (note that in the analyses below we focus primarily on how each scheme implements an authorised domain; more detailed analyses of these schemes can be found in [7]). There are other DRM schemes; however, many such schemes do not address the authorised domain concept, and only focus on binding a licence to a single device. Such schemes are not considered here, as they do not address the core theme of this paper. Nevertheless, such schemes (including some of the schemes discussed in this section) could be integrated with the proposed scheme for downloading content from content distributors to an authorised domain, as outlined in section 4.4.

The *Digital Rights Management in a 3G Mobile Phone and Beyond* scheme [8], has the following main problems. The domain key is an asymmetric key pair that changes every time a device joins or leaves the domain, or when a device is revoked; this step requires the domain owner to connect all devices to the Internet to retrieve the new key. In addition, each time the domain key is changed it requires all licence files to be re-encrypted, each of which holds a content encryption key. Each Domain Device must therefore keep track of all these licence files. This requires extra administration, in addition to requiring greater storage and processing costs. This could result in a significant overhead if the number of items of content is large. There is also no binding between the domain asymmetric key and the domain owner. This means that any device could be added to the domain, regardless of device ownership, as long as the device has not been revoked and a domain trusted authority authorises it. The scheme described in this paper addresses all these issues.

Other schemes have the same problems, apart from the scheme of Popescu et al. [22], that does not require domain content to be re-encrypted whenever a device joins or leaves a network. Moreover, most schemes fail to address content and domain key backup and recovery. Finally, other solutions have their own specific problems.

In the *OMA-DRM*[3] *system* [21], each device must securely store the domain keys, domain identifiers, and domain expiry times, provided by each Rights Issuer, for each domain that it joins. Devices require secure storage to store these keys. In addition, each Rights Issuer must create and manage all domains, and control which and how many devices are included or excluded from each domain. In order for a device to use the content in a domain, it must join all Rights Issuers from which the domain owner has downloaded content. This is because, as above, each Rights Issuer protects its content within a domain using a domain specific protection key provided by the Rights Issuer. This is not user friendly, as it imposes a significant administrative overhead.

The *DRM Security Architecture for Home Networks* [22] uses secret keys shared between pairs of devices for authentication, and a master key shared between each device and the domain manager. Each device is required to store the list of shared secret keys and the device master key, which increases the hardware costs. Extending the domain depends on the ability of all Domain Devices to increase their storage; it also requires the domain to be re-initialised and all domain content to be re-encrypted. In addition, the system

[3]http://www.openmobilealliance.org

requires each downloaded digital asset to be associated with a global device revocation list. This list will grow with time, and potentially results in a large increase in download time and size.

The *xCP Cluster Protocol* [13] encrypts content encryption keys using a master key, which is changed every time devices leave or join the home network. This step requires all devices to be online, which is inconvenient for some devices such as car CD players, MP3 players, etc. In addition, every time a device is hacked, a new media key block is released. This is a large data structure that imposes a significant overhead when moving it between devices and generating the master key, especially on devices that have limited capabilities.

The *SmartRight system* [23] requires the presence of smart card readers on all devices, and at least two smart cards per device: one for content access and the other for presentation functionality. This increases the total cost, especially bearing in mind the smart card maintenance costs. In addition, in order to increase the domain size, the system has to be re-initialised with a new domain key. Moreover, if all Terminal cards are lost or fail, then all existing content will be unusable, and if the Terminal card that most recently joined the domain is lost or stolen, then no other devices can be added.

9. CONCLUSION

This paper presented a novel solution for the protection of proprietary digital content against illegal copying. The basis of the solution is a means for identifying device ownership using the 3GPP GAA. This gives consumers the flexibility to transfer their digital rights to devices they own, and at the same time ensures content is protected against illegal copying. Our proposed solution meets most of the requirements for a DRM system (see, for example, [7]), in addition to avoiding problems found in other proposed solutions.

10. REFERENCES

[1] 3rd Generation Partnership Project. *3GPP TS 21.133 — 3G Security; Security Threats and Requirements. Specification version 4.1.0 Release 4*, December 2001.

[2] 3rd Generation Partnership Project. *3GPP TS 33.120 — 3G Security; Security Principles and Objectives. Specification version 4.0.0 Release 4*, March 2001.

[3] 3rd Generation Partnership Project. *3GPP TS 33.908 — General Report on the Design, Specification and Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms. Specification version 4.0.0 Release 4*, September 2001.

[4] 3rd Generation Partnership Project. *3GPP TS 33.102 — 3G Security; Security architecture. Specification version 7.0.0 Release 7*, December 2005.

[5] 3rd Generation Partnership Project. *3GPP TS 33.220 — Generic Authentication Architecture (GAA) — System Description. Specification version 7.0.0 Release 7*, March 2006.

[6] 3rd Generation Partnership Project. *3GPP TS 33.919 — Generic Authentication Architecture (GAA) — Generic Bootstrapping Architecture. Specification version 7.4.0 Release 7*, June 2006.

[7] Imad Abbadi. Digital asset protection in personal private networks. In *8th International Symposium on Systems and Information Security (SSI 2006), Sao Jose dos Campos, Sao Paulo, Brazil*, November 2006.

[8] E. A. Dabbish and T. S. Messerges. Digital rights management in a 3G mobile phone and beyond. In Joan Feigenbaum, Tomas Sander, and Moti Yung, editors, *Proceedings of the 3rd ACM workshop on Digital Rights Management*, pages 27–38. ACM Press, NY, 2003.

[9] Eimear Gallery and Allan Tomlinson. Secure delivery of conditional access applications to mobile receivers. In Chris J. Mitchell, editor, *Trusted Computing*, chapter 7, pages 195–237. IEE, 2005.

[10] Trusted Computing Group. Trusted platform module FAQ.

[11] André Günther and Christian Hoene. Measuring round trip times to determine the distance between WLAN nodes. In Raouf Boutaba, Kevin C. Almeroth, Ramn Puigjaner, Sherman X. Shen, and James P. Black, editors, *Proceedings of 4th International IFIP-TC6 Networking Conference, Waterloo, Canada*, volume 3462 of *Lecture Notes in Computer Science*, pages 768–779. Springer-Verlag, Berlin, May 2005.

[12] Bradley Huffaker, Marina Fomenkov, Daniel J. Plummer, David Moore, and K. Claffy. Distance metrics in the Internet. In *IEEE International Telecommunications Symposium*, 2002. http://www.caida.org/publications/papers/2002/Distance/.

[13] IBM Research Division Almaden Research Center. xCP cluster protocol, 2003. http://www-03.ibm.com/solutions/digitalmedia/doc/content /bin/xCPWhitepaper_final_1.pdf.

[14] International Organization for Standardization. *ISO/IEC 9798-3, Information technology — Security techniques — Entity authentication — Part 3: Mechanisms using digital signature techniques*, 2nd edition, 1998.

[15] International Organization for Standardization. *ISO/IEC 21481: Information technology — Telecommunications and information exchange between systems — Near Field Communication Interface and Protocol -2 (NFCIP-2)*, 2005.

[16] International Organization for Standardization. *ISO/IEC 18033-2, Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*, 2006.

[17] International Organization for Standardization. *ISO/IEC 2nd CD 19772, Information technology — Security techniques — Authenticated encryption mechanisms*, 2006.

[18] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: keyed-hashing for message authentication. RFC 2104, Internet Engineering Task Force, February 1997.

[19] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — OCSP. RFC 2560, Internet Engineering Task Force, June 1999.

[20] A. Niemi and J. Arkko. Hypertext transfer protocol (HTTP) digest authentication using authentication and key agreement (AKA). RFC 3310, Internet Engineering Task Force, September 2002.

[21] Open Mobile Alliance. *DRM Specification — Version 2.0*, 2006.

[22] B. C. Popescu, F. L. A. J. Kamperman, B. Crispo, and A. S. Tanenbaum. A DRM security architecture for home networks. In Joan Feigenbaum, Tomas Sander, and Moti Yung, editors, *Proceedings of the 4th ACM workshop on Digital Rights Management*, pages 1–10. ACM Press, NY, 2004.

[23] Thomson. SmartRight technical white paper, 2003. http://www.smartright.org/images/SMR/content/SmartRight_tech_whitepaper_jan28.pdf.

[24] Trusted Computing Group. *TPM Main, Part 1, Design Principles. Specification version 1.2 Revision 94*, 2006.

[25] Trusted Computing Group. *TPM Main, Part 2, TPM Structures. Specification version 1.2 Revision 94*, 2006.

[26] Trusted Computing Group. *TPM Main, Part 3, Design Principles. Specification version 1.2 Revision 94*, 2006.