Indexing and Retrieval of Multimedia Metadata on a Secure DHT

Walter Allasia and Francesco Gallo
Research and Innovation Department
EURIX Group
26, Via Carcano, Torino, Italy
E-mails: allasia@eurixgroup.com, gallo@eurixgroup.com

Marco Milanesio and Rossano Schifanella
Computer Science Department
University of Torino
185, Corso Svizzera, Torino, Italy
E-mails: milane@di.unito.it, schifane@di.unito.it

Keywords: multimedia metadata, digital rights, secure distributed hash table, peer-to-peer

Received: August 31, 2008

This paper proposes a decentralized, distributed and secure communication infrastructure for indexing and retrieving multimedia contents with associated digital rights. The lack of structured metadata describing the enormous amount of multimedia contents distributed on the the web leads to simple search mechanisms that usually are limited to queries by title or by author. Our approach is based on structured peer-to-peer networks and allows complex queries using standard MPEG-7 and MPEG-21 multimedia metadata. Moreover, security aspects limit the development of general purpose real applications using a peer-to-peer routing infrastructure for sharing digital items with an associated license. Accordingly, we propose a framework made up of a secure Distributed Hash Table layer based on Kademlia, including an identity based scheme and a secure communication protocol, providing an effective defense against well known attacks.

Povzetek: Predstavljen je sistem za učinkovito indeksiranje in doseganje digitalnih vsebin.

1 Introduction

Nowadays the growing of digital items exchanged on the web increases the need of their accurate description. We can define metadata as the description of the data. Even if it is possible to share multimedia items, it is very difficult or impossible to search them without appropriate description provided by content metadata. Usually people making use of web-sharing systems do not provide detailed metadata information, which in most cases is only limited to the title or the author. This lack of information determines the growth of unstructured information. Using metadata it is possible to structure the information and thus, on one side, to enhance and enrich the information related to a content and, on the other, to search and retrieve digital items. It is clear that more detailed are the metadata, more complex is the structure which they are inserted on.

Moreover, in order to reach a common understanding of metadata, it is important to adopt standards. The adoption of MPEG-7 [1] for describing metadata related to the digital items and of MPEG-21 [2] for describing metadata related to a governed content (i.e., with an associated license), as proposed in this paper, is a common approach used by an increasing number of scientific communities. The use of the standards mentioned above can improve the expressiveness of the query language for the multimedia items and can make governable the content distribution.

The enormous amount of media available on the web promotes the adoption of completely decentralized infrastructures, such as peer-to-peer (P2P) content sharing systems, that minimize the impact of a single point of failure fostering scalability, reliability and efficiency. Unfortunately, such approaches introduce a large spectrum of security flaws that limit the adoption in a real scenario. In fact, if it is true that digital contents are growing up very fastly especially in such distributed environments, it must be noticed that such systems usually offer poor functionalities for indexing and retrieving structured information. The main issue comes from the flat indexing space that affects these systems: the lack of a central entity offering a complete representation of complex information (i.e., the set of the metadata characterizing the digital items) results in a poorly expressive query language (e.g., parsing of the query string and pattern matching). Moreover most of these topologies are not providing any kind of content government and in the worst case they are not taking into account any digital rights associated to the exchanged resources.

We propose a decentralized, distributed and secure communication infrastructure for the indexing and the retrieval of governed as well as ungoverned multimedia contents. Our approach, based on Distributed Hash Tables, allows complex queries to the system by means of complex multimedia metadata indexing. Moreover, the sharing of digital items on the basis of the associated license (either free or not), enables the usage of the P2P routing infrastructure for real applications, where a particular care has to be devoted to security aspects.

The main contributions of our work are summarized in the following:

- a decentralized scheme to index and retrieve structured metadata related to multimedia contents,
- a policy to manage digital rights expressed by MPEG-21 Rights Expression Language (REL) [3] profiles that enables the governed sharing of digital items along with the protection of the intellectual property,
- a secure structured overlay network that assures the basic security functionalities providing an effective defense against well known attacks,
- a Java-based prototype implementation that shows the feasibility of our approach.

The remainder of this paper is structured as follows. Section 2 presents an overview of the related studies available in the literature, while Section 3 describes the general model developed for the proposed framework. The indexing and retrieving schemes are discussed in Section 4, while Section 5 introduces a secure structured overlay infrastructure built on top of Kademlia that provides a defense against well known attacks. Moreover, Section 6 presents a Java-based implementation of the proposed system. Finally, Section 7 discusses some concluding remarks and future works.

2 Related Works

In this Section we focus on a general overview about the building blocks that compose the proposed framework. For network topology we adopted a structured P2P network based on a Distributed Hash Table, described in Section 2.1, where the fundamental properties are briefly discussed. In Section 2.2 an overview of security concerns related to DHTs, based on the available literature, is presented.

For metadata representation we adopted the MPEG-7 [1] and MPEG-21 [2] standards, which are outlined in Section 2.3. Concerning the governed content management, we adopted the solutions developed by the *Digital Media Project* (DMP) [4]. Accordingly, Section 2.3.1 describes the overall architecture of *Chillout* [5], the reference software implementation of the ISO/IEC 23000-5 (Media Streaming Application Format) standard.

2.1 Distributed Hash Tables

Distributed Hash Tables (DHTs) [6, 7, 8] are a class of distributed algorithms that provides the same functionality of a traditional hash table, by making available the mapping between a *key* and a *value*. DHTs are typically designed to scale to large numbers of nodes and to handle continual node arrivals and failures. The basic functionality provided by a DHT is the `lookup(key)` operation that returns the identifier of the node responsible for the `key`. In a DHT, nodes and objects are assigned with random identifiers (called node IDs and keys, respectively) from a large ID space. Given a message and a key, the DHT routes the message to the node with the node ID that is numerically closest to the key in a logarithmic number of hops with respect to the size of the network. In order to route a message, each node maintains a local routing table that contains information on a logarithmic subset of the entire system, granting scalability to the structured P2P system.

Even if DHTs offer a very good level of scalability and robustness, they suffer also from various drawbacks. First of all, in order to locate the node that stores a key, one needs to know in advance the exact identifier, but this cannot be always assumed at the application level. This phenomenon is known as the *exact match lookup* problem. As a consequence, distributed applications based on structured peer-to-peer overlay networks have to set up an interface to communicate with the P2P network providing the keys used for both routing messages and searching resources. A typical solution allows the insertion of meta-information and meta-keys extracted from the query string (such as in eMule[1] with Kademlia support).

Another relevant issue arises when a new node joins the network: it needs to know at least one living peer that is contacted in order to gather the necessary information to build the peer's state and the related routing table. Obviously, this node (called *bootstrap node*) represents a single point of failure: if it is off-line, the oncoming node can not enter correctly the system. However, usually the new peer holds a list of existing peers and it contacts each of them until an on-line node is reached. Of course, the presence of the bootstrap node raises also some security issues since the correctness of information provided is necessary to ensure a valid join mechanism. Furthermore, in most DHTs every information is replicated and cached in the system, to improve reliability and performance: this leads to the problem of balancing the trade-off between consistency and communication overhead between peers that need to update their cache. Finally, the DHT paradigm assumes that all peers equally participate to the system without any difference in terms of bandwidth, computational power or resource availability of nodes. In such a scenario, it is possible that low-capacity peers act as a bottleneck in terms of system performance.

[1]http://www.emule-project.net. Last visited: 15 Nov 2008.

2.2 Security Flaws on DHTs

Recently, a lot of effort has been put on securing DHTs [9] and the applications built on them. The usual robustness and efficiency of a DHT-based system can be overwhelmed by the malicious behavior of groups of peers that do not follow properly the DHT protocol. Examples of attacks that a DHT-based application has to face with can be divided regarding the targets: the overlay routing (e.g., eclipse attack, sybil attack, churn based attacks, and adversarial routing) or the applications (e.g., DDoS, attacks on Data Storage). In the following we will give a brief overview of all the attacks which are tipically carried against DHTs.

2.2.1 Threats

A popular family of attacks is known as *routing poisoning*. As active nodes' routing tables are maintained and renewed through a push-based approach (i.e., unsolicited messages, such as the publication of route tables of neighboring nodes or lookup messages sent from unknown nodes, supply an information that is used to update table's entries), it is possible for a malicious peer to inject random routing data into victim nodes, (e.g., during bootstrapping).
When carried against nodes, a particular form of routing poisoning is the so-called *eclipse attack* which aim is to separate a set of victim nodes from the rest of the overlay network, mediating most overlay traffic and effectively eclipsing correct nodes from each other's view. When carried against the stored contents on a DHT (i.e., making inaccessible the values of the DHT), the Eclipse attack is called *node insertion attack*: a vast number of nodes marked with identifiers numerically close to the target identifier are initiated, intercepting thus most of the lookup requests and answering with fake contents or not replying at all, effectively hiding the content.

Since typically there exists no verifiable link between the participating entity (human user or machine) and its identity (the nodeId), it is possible for any entity to show multiple identities to the system. The generation of multiple identities under a single entity is called Sybil attack and it undermines the redundancy property of a P2P system, because it enables the gathering of a large number of nodes on few machines, centralizing unsafely many keys' responsibilities and content replicas. The Sybil entities are usually exploited to increase the effectiveness of other attacks (e.g., Eclipse, DDoS) without needing huge computational resources or without the help of other colluding entities.

An index poisoning based attack [10] consists in inserting corrupted contents among the storages of a group of index nodes. A corrupted content might be something not related to the key for which it was stored, or even a fake information, like a reference to the wrong source. An attacker can make a bogus content highly visible by flooding fictitious records under 'strategic' indexes (e.g., among nodes responsible for "hot" keys), flushing legitimately stored content. In file-sharing applications, the most similar attack is the *content pollution*, that inserts on the DHT fake meta-data (i.e., meta-data that should be correct but that point to corrupted resources).

A distributed denial of service attack consists in inducing a large number of nodes of the overlay to generate a huge amount of messages to be sent to a target entity located internally or externally the P2P network. It can be achieved with a redirect technique [11], carried out through an index poisoning attack. In file-sharing systems, the attacker can insert meta-data related to a very popular content, pointing to the target IP address as a source of such a file: the victim will be overflowed by connection requests until the 'polluted' content will be kept in index nodes' storage.

Concluding this overview on the attacks, it is worth notice that some studies [12] show that in the Kad network at least half of the network is prone to a Man In The Middle attack. To avoid this, communicating must be sure about the integrity of messages and about the identity of the sender. An authenticated channel between endpoints can instantly exclude a third malicious entity.

2.2.2 Defenses

Most of the overlay routing attacks countermeasures are given in terms of routing protocol changes or access control policies. An exhaustive overview of the commonly used distributed access control mechanisms is given in [13], stressing the difference between the different threshold signatures. Authors underline that the use of RSA for generating keys in a distributed environment leads to an high communication and computation overhead, particularly harmful for mobile and ad-hoc networks. Saxena et al. [14] developed an identity-based group admission control technique that overcomes the drawbacks of previous certificate-based approaches, presenting ID-GAC (ID-based Group Admission Control), based on the threshold version of BLS signature scheme, an identity-based mechanism since the membership token used to prove membership is derived from the group member's identity. The use of a super singular elliptic curve influences the overall cost of the scheme. The whole scheme comes along with a distributed membership revocation mechanism based on the membership revocation lists.

A possible approach to locate Sybil nodes is periodically sending a different challenge to each node: requiring a high computational effort to be solved, one machine cannot solve a challenge for each Sybil node it hosts within a specified short time interval. The main issue within this approach is the difficulty to practice it in an heterogeneous domain [15]. A central authority that assigns a certified nodeId only after a user registration process might limit this phenomenon, because the time required to the creation of a new node would be considerably longer.
An exhaustive overview of the different behaviors of peers in the KAD network is given in [16]: among other results, it's clear that node identifiers are not necessarily persistent as was assumed in previous works. In [17], authors con-

sider the vulnerability of KAD against Sybil Attack and point out that a solution is to prevent a peer from choosing its own ID and avoiding a peer to obtain a large number of IDs. For doing so, they sketch out a centralized solution that makes it impossible for an attacker to obtain arbitrary KAD IDs: a central agent binds the KAD ID to a cellular phone number.

Sybil Attack is also the core of the work in [18] and [19]. In the first work, a resistant routing strategy is introduced on a variant of Chord, assuring that lookups are performed using a diverse set of nodes, and thus that at least a subset of the nodes involved in the lookup process is not malicious. As a consequence, the lookup process makes forward progress, not only converging fast to the destination, but also minimizing the number of trusted bottlenecks: when choosing the next node in the path, the variant will take into account the sources of information about the previous hops, and strive to avoid relying on a single trusted bottleneck. In [19] an admission control system for structured P2P systems is given. The system constructs a tree-like hierarchy of cooperative admission control nodes, from which a joining node has to gain admission. The admission control system is implemented by the nodes, and it examines joining nodes via client puzzles. The burden of self-organization and admission control is placed on the peer-to-peer nodes themselves. For this reason, the computational load of these activities must be low. Analysis shows that these costs are vanishingly small for all nodes in the network. Admission Control System (ACS) defends against Sybil attacks by adaptively constructing a hierarchy of cooperative admission control nodes. A node wishing to join the network is serially challenged by the nodes from a leaf to the root of the hierarchy. Nodes completing the puzzles of all nodes in the chain are provided a cryptographic proof of the examined identity.

A tool which could effectively combat the content pollution and the index poisoning attacks is the use of credentials, bound to the content, provided by the owner of the content during the insertion phase: if the content is bound to the identity of an owner, when a fake resource is found, it is possible to trace back to content creator. If the application implements a reputation system, it could be possible to penalize or even to ban a malicious node.
Credentials and reputation systems can also be used against DDoS: as it would be too costly to oblige replica nodes to verify the authenticity of each inserted content, it is necessary to adopt a reputation system so that peers who have made incorrect insertions are recognized as soon as possible and banned from the network.
Against the Eclipse attack, an anonymous auditing technique is proposed in [20], but still it is shown to be ineffective against Node insertion attack: the introduction of a third party trusted certification service that assigns randomly generated certified identifiers to nodes seems to be an effective solution to prevent this attack.

S/Kademlia [21] is a secure key-based routing protocol based on Kademlia [22] that has a high resilience against common attacks by using parallel lookups over multiple disjoint paths, limiting free nodeId generation with crypto puzzles and introducing a reliable sibling broadcast, needed to store data in a safe replicated way. In order to make Kademlia more resilient they suggest limiting free nodeId generation by using crypto puzzles in combination with public key cryptography, extending the Kademlia routing table by a sibling list, reducing the complexity of the bucket splitting algorithm and allowing a DHT to store data in a safe replicated way, and finally a lookup algorithm which uses multiple disjoint paths to increase the lookup success ratio.

In [23] periodic routing table resets, unpredictable identifier changes and a rate limit on routing table updates are given, in order to make attackers unable to entrench themselves in any position that they acquire in the network, and also to make them unable to fix an appropriate strategy for addressing some specific nodes. Authors propose also a practical defense against the eclipse attack, extending the Bamboo DHT[2].

A distributed node ID generation scheme would limit the rate in which an attacker can obtain IDs. The former authors of Pastry [24] require prospective nodes to generate a private/public key pair such that the hash of the public key has the first p bits equal to zero [25]. They also suggest binding the IP address of the node with its ID. To overcome the possibility of an attacker to accumulate node IDs they suggest periodically to invalidate node IDs and using different setting for the hash initialization. However, this would require legitimate nodes to obtain new IDs every time this happens. Authors show how the use of secure routing can be reduced by using self-certifying application data.

Finally, an admission control framework suitable for different flavors of peer groups nd match them with appropriate cryptographic techniques and protocols is presented in [26].

2.3 Multimedia Metadata Representation

MPEG-7 [1], formally named Multimedia Content Description Interface, provides a rich set of standardized tools to describe multimedia contents. It mainly focuses on description of the digital items, without considering how and where this information is used. In particular, the MPEG-7 descriptions of content may include (1) information describing the creation and production processes of the content (director, title, short feature movie), (2) information related to the usage of the content (copyright pointers, usage history, broadcast schedule), (3) information of the storage features, (4) on spatial, temporal or spatio-temporal components or about low level features (colors, textures, sound timbres, melody description) and many others.

MPEG-7 standard has been included in several metadata language, such as ODRL (Open Digital Rights Language[3]) and has been coupled with other important TV ontologies

[2]http://bamboo-dht.org. Last visited: 15 Nov 2008.
[3]http://www.w3.org/TR/odrl. Last visited: 15 Nov 2008

(e.g., TVAnytime RMPI [27]). Concerning digital rights, MPEG-7 provides a standard XML schema and the metadata to define conditions for accessing the content (including links to a registry containing intellectual property rights data and price) and additional information about the content (copyright pointers, usage history, broadcast schedule). An MPEG-7 Query Format reached the Final Committee Draft, during the MPEG meeting on October 2007. Moreover several query frameworks based on MPEG-7 are still under investigation [28].

MPEG-21 [2] differs from MPEG-7 because it aims to define a normative open framework to be used by all the players in the delivery and consumption value chain. This framework will provide an open market to content creators, producers, distributors and service providers. The goal of MPEG-21 is the definition of a standard technology needed to support users in order to exchange, access, consume, trade and otherwise manipulate digital items in an efficient, transparent and interoperable way. In particular, part 5 of MPEG-21 defines a Rights Expression Language (REL) to be used in the description of customized rights applied to any digital item, since it is seen as a machine-readable language that can declare rights and conditions defined in the Rights Data Dictionary (also standardized by MPEG-21). Rights metadata are expressed by means of MPEG-21 REL, which describes the license associated to a specific resource, along with several available rights (*play*, *copy*, *modify*, *print*, etc.). According to the schema shown in Figure 1 [29] we can imagine the license as made up of an issuer (with multiplicity 0 or 1), an undefined number of grants (multiplicity 0 or more), and a principal (multiplicity 0 or 1). The issuer is the owner of the rights associated to a given content (eventually coincident with the creator or distributor of the resource) and can assign a given right (e.g., the authorization to copy or modify the content) to the principal. For example, in the wide commonly used *CreativeCommons* [30] licenses the principal is not specified since this kind of license is intended for everyone.

2.3.1 Chillout

Chillout [5] is the reference software of the Digital Media Project (DMP) [4]. DMP is a no profit organization that has recently approved a version 3.0 of its specification, called Interoperable DRM Platform (IDP-3.0). Chillout is also the reference implementation of ISO/IEC 23000-5 Media Streaming Application Format [31], addressing the distribution of governed content over streaming channels. The most important technologies adopted by Chillout are: (a) a data structure capable of hosting different data types accompanying a resource (e.g., audio, video, image, text, etc.), (b) a content identification system, (c) a set of technologies for content protection, (d) the Rights Expression Language, (e) a file format for storing digital items and resources and (f) a technology to transmit digital items in streaming mode.
Two file formats for managing digital contents are used as depicted in Figure 2: DCI (DMP Content Information) and DCF (the DMP Content File) [32]. The DCI is a standard XML-based format which is intended mainly to express the license metadata and is compliant with two MPEG-21 REL profiles: the Open Access Content (OAC) profile [33], for expressing equivalent CreativeCommons licenses, and the Dissemination And Capture (DAC) profile [34], mapping the TV Anytime RMPI [35] licenses, used in the broadcasting domain. The specification of the DCI allows also to include the MPEG-7 representation for the content. The DCF file has been conceived as a container of the DCI and the resources as well and we extract a subset of the metadata contained in the DCF for indexing. The resources can be stored within the DCF file or can be referred to by means of pointers.

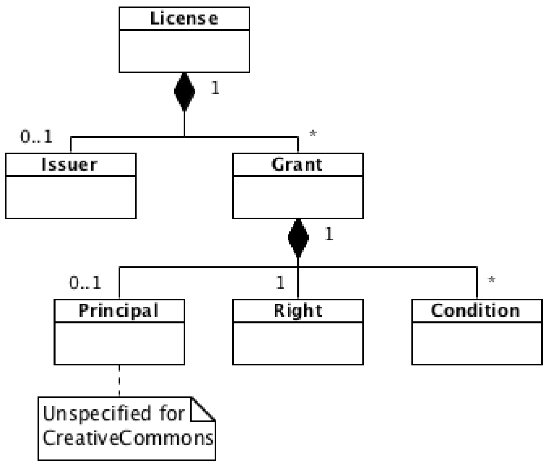

Figure 1: Simplified diagram of a REL license.

3 Model

In the previous Sections we have presented the building blocks (secure DHT, MPEG multimedia metadata and Chillout) that have been used in our solution in order to create a prototype system that is able to *share governed contents on P2P networks*, where *share* here means the possibility to publish, index, search, retrieve and consume a digital item and *governed* refers to the fact that each digital content distributed on such system is governed according to its associated license. It is worthwhile pointing out that a DRM system could use the proposed solution as the underlying software to manage (create, index, retrieve) govenred contents, demanding to an other application software placed on top of it to manage or not the associated digital rights. This solution allows also the integration of the proposed prototype with proprietary DRM solutions, where the content representation is based on MPEG standards. Moreover, despite the common feeling about P2P networks in relationship with abuse or violation of digi-

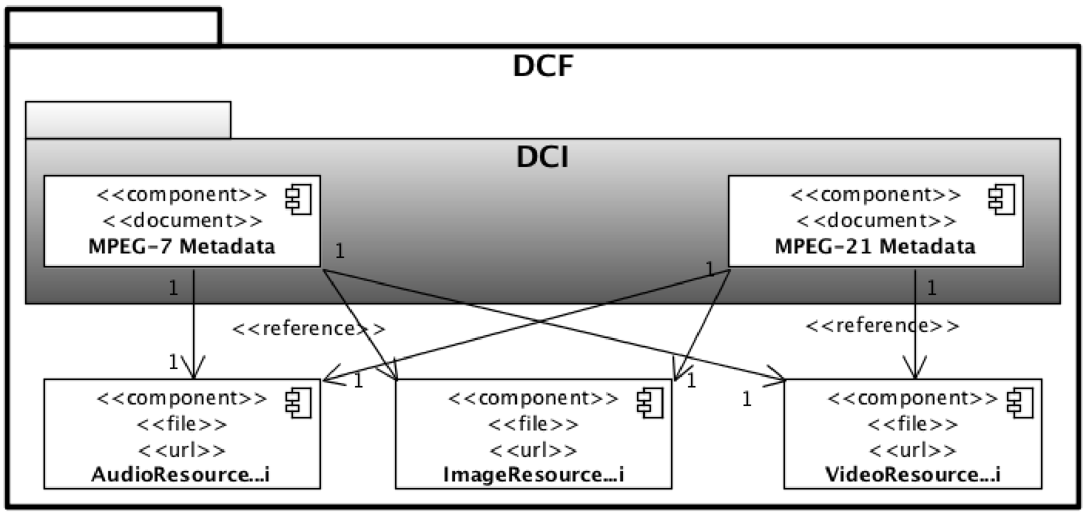

Figure 2: DCF and DCI structure diagram.

tal rights and intellectual property rights in general, mainly due to the sharing of copyrighted or otherwise licensed content, the software solution proposed in this paper proofs that it is possible to have content government on these popular networks and is also possible to make them secure.

We make use of the MPEG-7 standard for expressing the metadata related to the digital content itself, describing the user metadata (e.g., title or author) as well as the metadata describing the content as visual descriptors (e.g., ScalableColor or HedgeHistogram). We have adopted the MPEG-21 standard for expressing licenses because MPEG-21 REL provides several profiles for specific environments and purposes (broadcasting, mobile applications,...), which guarantee high interoperability with other rights languages and therefore it is able to express most of the possible licenses. As described in Section 2.3.1 Chillout can manage governed content using MPEG-7 and MPEG-21 representation, which is contained into a DCI structure, specified by ISO/IEC 23000-5. Hence the proposed solution is able to share on a secure DHT the DCF files and to index the metadata stored in the DCI files.

As shown in Figure 3, our approach is made up of three logic layers:

- *User Interaction Layer*, where the several user software components communicate with the application layer providing and consuming digital contents.

- *Application Layer*, which is in charge for extracting the information to be indexed and for communicating with the DHT layer in order to index the related keys.

- *Overlay Layer*, which is responsible for the management of the overlay network and the routing of messages.

On top of the layered architecture, the *User Interaction Layer* describes the way an entity can interact with the application exploiting the provided functionalities. As shown by the components depicted in Figure 3, a user device can play different roles:

- *Content Creator*, which is the component responsible for the creation of governed content (in DCF format), making use of user resources and the associated licenses (expressed in the DCI file).

- *Content Provider*, which is the component responsible for providing governed contents that can be created by the same user as well as by others.

- *Player (End User)*, which is the component that can consume the resources according to the associated licenses. When the user asks the system to consume a resource, it recognizes which rights are guaranteed to the current user (e.g., copy, play, modify, distribute) and can enforce them.

The *Application Layer* is made up of three main components: *Retrieving*, *Indexing* and *Exchanging*, as shown in Figure 3. The *Retrieving component* provides functionalities for (a) extracting a defined subset of MPEG-7 and MPEG-21 metadata from the DCF file associated to the governed resource, (b) computing the identifiers associated with the extracted metadata, (c) querying the underlaying DHT with the computed identifiers and (d) collecting and merging the lookup results.

The *Indexing component* provides functionalities for (a) extracting a defined subset of MPEG-7 and MPEG-21 metadata from the DCF file associated to the governed resource, (b) computing the identifiers associated with the extracted metadata, (c) inserting the governed resource in a storage layer and (d) inserting the relative mappings in the DHT.

A user can search for resource related metadata (e.g., the title in MPEG-7), license related metadata (e.g., the issuer in MPEG-21 REL) or a combination of the two. A detailed description of the *Retrieving* and *Indexing* components can be found in Section 4.

The *Exchanging component* communicates with the *Transport component* by mean of two socket connections (represented in Figure 3 using UML 2.0 [36] conventions), one for exchanging metadata information that are basically DCI documents and the other for exchanging the real digital content, for example as byte array. This communication is asynchronous and completely separated. The user can make use of the metadata exchanging component looking for several resources and can decide to download only one of them, making use of the other exchanging component, the one for accessing the actual contents.

Finally, the *Overlay Layer* is made up of the *DHT* and the *Transport* components. The former is responsibile for the DHT management and is described in Section 5 while the latter is responsible for exchanging/downloading the contents between peers and also for exchanging the full metadata available in the DCF and contained in the DCI. In order to provide a system open to further extensions, we layered the application core functionalities of the DHT component under a *facade design pattern* which can be considered as a bundle of interfaces widely used by different DHT implementations. This choice improves the system flexibility, allowing the choice of other DHT implementations with no (or only minor) changes.

4 Indexing and Retrieving

A goal of the proposed approach is to provide a fully distributed system that exploits the scalability, resiliency, and efficiency properties of DHTs in order to index and retrieve audiovisual contents through their descriptions.

In all DHTs, to each node and resource, an identifier computed from the same space is given. This means that there is no way, starting from an identifier, of knowing if this is the index of a node, of a resource or anything else. What can be exploited within these distributed algorithms is the key consistency and the collision avoidance of the used hashing functions. Once computed several identifiers, at an application level it will be possible to address any domain specific data structure complexity. The DHT layer allows, in any case, the convergence of the routing mechanism and the scalability of the system, as the diameter of the system will never be bigger than $O(logN)$, for N nodes in the network.

Accordingly, it is clear that there is a discrepancy between metadata representation and the way in which information are stored in a DHT-based infrastructure. In the first case, the content is described by a structured XML-based formalism, e.g., MPEG-7 and MPEG-21 documents, in the second case, the information are codified in a flat set of $\langle key, value \rangle$ relations. To fill up this gap, we proposed an iterative indexing and retrieving scheme [37] that is similar to the hierarchical indexing scheme described in [38].

Let's consider a generic audiovisual content R that is associated with a set of metadata. Such metadata are extracted during the indexing phase from the MPEG-7 and/or MPEG-21 documents related to R. As described in Section 2.3 both MPEG-7 and MPEG-21 standards contain a large spectrum of data describing multimedia contents and digital rights. Therefore, it is evident that to index the complete metadata knowledge could represent a very expensive computational and spatial cost. To lighten the load of each node, we decided to not index all the metadata: we chose a subset of the overall tags, used for a first step of the query process. To refine the result set it is possible to query locally the retrieved resources against the complete schema through well-established approaches. This hybrid strategy can lead to a good trade-off between efficiency, scalability and query expressiveness.

In more details, given the resource R, we define the subset of medatata $M_R = \{m_0, m_1, ..., m_n\}$ where the generic m_i has the form $m_i = (tag_0|[attribute_0]| \ldots |tag_m|[attribute_m]|value)$. In other words, each metadata item is composed by chaining the tags and the optional tags attributes with the corresponding value. For sake of simplicity, assume a MP3 audio file that has to be indexed on the system. Figure 4 shows the MPEG-7 document related to the song. In this scenario, we can describe the title by way of the metadata item $m_i = (title|songtitle|Times\ Like\ These)$ or the genre through $m_j = (genre|name|Acoustic\ Rock)$. After the selection of the metadata, we compute the identifier id_R (calculated applying the hashing function $hash()$ of the resource itself) and the set of identifiers $I_{M_R} = \{id_{m_0}, id_{m_1}, ..., id_{m_i}\}$ where the generic id_{m_i} is equal to $hash(m_i)$. Each identifier must reference id_R, in order to allow metadata based queries. We insert on the DHT a set of $\langle key, value \rangle$ pairs in the form $\langle id_{m_i}, id_R \rangle, \forall\ id_{m_i} \in I_{M_R}$, along with the relation $\langle id_R, R \rangle$. This basic scheme allows a user to retrieve the resource associated to a metadata m_i. In fact, during the retrieving phase the system calculates the identifier id_{m_i} and, by means of a $lookup(id_{m_i})$, the related resource identifier id_R. Then, R itself is obtained. For instance, let's suppose that a user wants to find the song entitled *"Times Like These"*. She submits the request to the system that computes m_i following the rules depicted above, then it calculates id_{m_i} and, by means of a $lookup(id_{m_i})$, it retrieves the identifier of the corresponding resource. At last, a subsequent lookup is able to get (directly or indirectly) the requested resource.

A key aspect of our approach is the ability to index and retrieve audiovisual items with associated digital rights. We can divide contents between governed and ungoverned. Ungoverned items do not have licenses associated and the keywords to be indexed are just MPEG-7 elements. Gover-

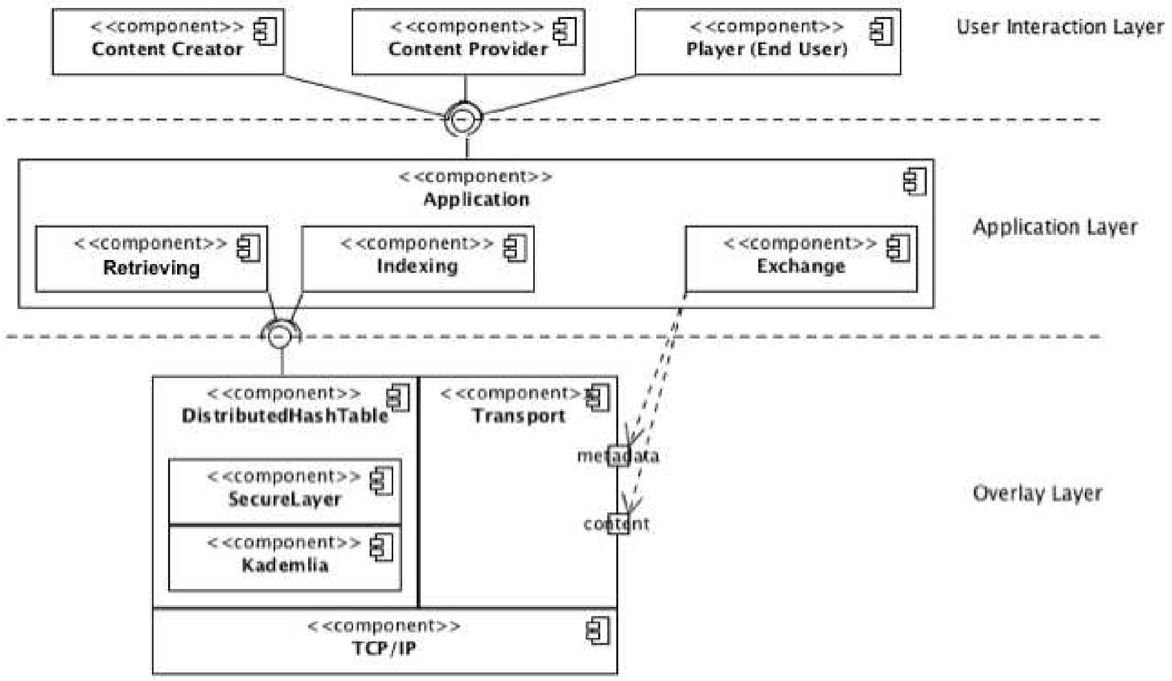

Figure 3: System overview

```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7  xmlns="urn:mpeg:mpeg7:schema:2001"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001
     http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-7_schema_files/mpeg7-v2.xsd">
     <Description xsi:type="CreationDescriptionType">
         <CreationInformation id="jj-2005-onon-track-01">
             <Creation>
                  <Title type="songTitle">Times Like These</Title>
                  <Title type="albumTitle">On and On</Title>
                  <Creator>
                       <Role href="urn:mpeg:mpeg7:RoleCS:2001:PERFORMER"/>
                       <Agent xsi:type="PersonType">
                           <Name>
                                      <FamilyName>Johnson</FamilyName>
                                      <GivenName>Jack</GivenName>
                           </Name>
                       </Agent>
                  </Creator>
                  <CreationCoordinates>
                       <Date><TimePoint>2003</TimePoint></Date>
                  </CreationCoordinates>
             </Creation>
             <Classification>
                  <Genre href=" urn:id3:cs:ID3genreCS:v1:80"><Name>Acoustic Rock</Name></Genre>
             </Classification>
        </CreationInformation>
   </Description>
</Mpeg7>
```

Figure 4: Example of a MPEG-7 description for a MP3 audio file.

ned items have a license and we have defined the following structure to be indexed: for each right described in the license we index three MPEG-21 REL tags: *issuer*, *right*, *principal* (see Section 2.3). Although typical licenses contain one or more grants, we assume in the following a single issuer and a single principal for each right and for every grant expressed in the license we index the bundle of issuer, right and principal linking the associated content. Hence, the DHT contains the indexes of the general purpose metadata and in addition, for governed resources, the bundle of grants linking the digital item.

Let's consider again the resource R with an associated MPEG-21 REL license as shown in Figure 5. We extract the following metadata elements:

$$\begin{aligned} m_{issuer} &= (issuer|keyholder|keyname|value) \\ m_{right} &= (grant|value) \\ m_{princip} &= (principal|keyholder|keyname|value) \end{aligned}$$

For instance, we have that $m_{issuer} = (issuer|keyholder|keyname|Jack\ Johnson's\ key)$ and $m_{right} = (grant|play)$. The principal is not defined since the item is governed by a *Creative Commons License*. Afterwards, a key is calculated for the metadata, i.e., id_{issuer}, id_{right} and $id_{princip}$ respectively, and the mappings $\langle id_{issuer}, id_R\rangle$, $\langle id_{princip}, id_R\rangle$, and $\langle id_{right}, id_R\rangle$ are put on the DHT as explained in the MPEG-7 scenario.

In order to allow complex queries, all possible combinations of those three metadata are inserted. In other words, we derive the following relations:

$$\langle hash(m_{issuer}|m_{right}), id_R\rangle$$
$$\langle hash(m_{right}|m_{princip}), id_R\rangle$$
$$\langle hash(m_{issuer}|m_{right}|m_{princip}), id_R\rangle$$

In this scenario, a user could search for "all the digital items issued by someone", or could submit composite queries like "all the contents with a grant of copy issued by someone", beyond looking for titles and authors.

In summary we are indexing rights metadata on a structured overlay network, allowing users to search governed resources looking for specific issuers, grants or principals. It is worth noting that our system easily enables keyword-based queries like eMule with Kademlia support does. In this case, we index keywords extracted from the file name combining them to allow complex queries as described above.

5 Secure DHT Layer based on Kademlia

As previously underlined, one of the main concern that limits a broad adoption of a DHT-based content sharing platform is the security aspect. In this Section we will describe a communication protocol and an identity management scheme that provide a secure layer on which general purpose applications can be built.

The adversary model considered here is composed by nodes in the DHT system (with reference to Kademlia) that do not properly follow the protocol. We assume that a malicious node is able to generate packets with arbitrary contents (including forged source IP addresses) and, furthermore, to overhear or modify communications between other nodes.

Kademlia [22] is a structured P2P system featured by the use of a XOR metric for computing distance between points in the identifier space. In Kademlia every node has a random 160-bit *nodeId* and maintains a routing table consisting of up to 160 k-buckets. Every k-buckets contains at most *k* entries with <IP address, UDP port, NodeId> triples of other nodes, with *k* as a redundancy factor for robustness purposes. Buckets are arranged as a binary tree and nodes get assigned to buckets according to the shortest unique prefix of their nodeIds.

Kademlia combines provable consistency and performance, latency minimizing routing, and a symmetric, unidirectional topology.

The Kademlia protocol is vulnerable to all the attacks introduced in Section 2.2, even if it can mitigate the harmfulness of some of them. Nodes' identifiers are not certified and they can be generated at will on the local node, so it's possible to quickly instantiate a large number of Sybil nodes with arbitrary Ids in order to complete a node insertion attack. There is no credential associated with contents maintained in storages and no control is performed by replica nodes over the information stored in the DHT thus allowing the index poisoning and derivative attacks. There is no authentication protocol between nodes. Nevertheless, k-buckets provide resistance to certain DoS and index pollution attacks; in fact, one cannot flush nodes routing state by flooding the system with new nodes. Kademlia nodes will only insert the new nodes in the k-buckets when old nodes leave the system. Unfortunately, it is very easy to inject into a route table information relating to contacts whose identifier is very close to the victim node Id, because of the bucket splitting procedure.

Finally, it is possible to affect the lookup procedure to lead the searching node to contact a set of replica peers controlled by the attacker. The Kademlia lookup procedure for a key χ starts selecting α nodes whose ids are the nearest to the local id and sending to each of them a FIND-NODE(χ) RPC. The response to these messages are list of triples $< IP, Port, NodeId >$ that locates the contacts closest to among all the entries of the queried nodes' routing tables. The lookup initiator selects, among all received triple, α contacts whose nodeId is closest to χ and iterates the same procedure until it gets responses from the k nodes closest to χ it has seen. If a malicious node receives a FIND-NODE RPC, it responds with k triples that identify colluding nodes whose id is claimed to be close to the lookup key. The searching peer has no way for verifying messages and it will trust every response.

5.1 Protocol

In order to nullify or to reduce the impact of the DHTs' vulnerabilities, we define a framework that includes an identity based scheme and a secure communication protocol that may provide an effective defense against well known attacks. For further details refer to [39]. The proposed approach is layered on Kademlia and its architecture is based

```
<?xml version="1.0" encoding="UTF-8"?>
<license xmlns="urn:mpeg:mpeg21:2003:01-REL-R-NS"
  xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS" xmlns:m3x="urn:mpeg:mpeg21:2006:01-REL-M3X-NS">
  <grant>
     <mx:play/>
     <digitalResource licensePartId="jj-2005-onon-track-01">
         <nonSecureIndirect URI="urn:newspaper:news:2005_07_10-12H-00M"/>
     </digitalResource>
     <m3x:copyrightNotice noticeType="ShowBeforeExercise">
         <m3x:copyrightString> Written by Jack Johnson, 2005</m3x:copyrightString>
     </m3x:copyrightNotice>
  </grant>
  <issuer>
     <keyHolder>
        <info>
           <dsig:KeyName>Jack Johnson's key</dsig:KeyName>
        </info>
     </keyHolder>
  </issuer>
</license>
```

Figure 5: Example of a MPEG-21 REL license for the audio file described in Figure 4.

on the presence of a Certification Service (CS). The CS can be a centralized or decentralized authority whose task is to generate random nodeIds and to certify the link between nodeIds and users' identities by signing peculiar tokens. To accomplish this, we suppose that a classic public key cryptography scheme is used: in this section we assume that the CS is a centralized authority owner of a public key known to every Kademlia node, and holder of its private counterpart. Similarly, we assume that each user who intends to take advantage of the network services should be in possession of a key pair. The following notation is used throughout:

A, B	:	nodes
$NodeId_A$	:	A's Kademlia Id
$UserId_A$	:	A's user identifier
K_A^+, K_A^-	:	A's public and private key
K_{CS}^+, K_{CS}^-	:	CS public and private key
$Sign(m, k)$	:	message m signed with key k
$H(o)$	:	hash code of the object o
$AuthId_A$	:	node A's authenticated id
$Auth_{AB}$	:	authentication by A for B
ts, TTL	:	timestamp, time to live
$a\|\|b$	:	concatenation of strings

The proposal enhances the join procedure, the node interaction protocol and the content storage procedure defined by Kademlia. In a preliminary initialization phase a node applies to the Certification Service for a certified NodeId and for bootstrap information; since the certified NodeId has an extensive temporal validity, initialization is not executed at every bootstrap but only periodically. After the initialization, the node performs the network join procedure to take part to the overlay. In order to correctly interact with other nodes, the newly joined one must follow a communication protocol for incoming and outcoming messages; especially, the node must produce special credentials related to every content to be inserted in the DHT.

5.1.1 Initialization

Node A must obtain its own certified id, in order to interact with other peers. To this aim the node sends a request to the CS containing an identifier and its public key:

$$NodeIdReq = UserId_A, K_A^+$$

The $UserId_A$ is the identity by which user A presents himself to the network community. It is an identifier of a generic account of user A and whose validity must be verifiable by the same CS. It may be assumed that the $UserId$ is an existing and verifiable identity, (e.g., an OpenID URL or an email address), in which case the CS should initiate an interaction with an external authority (e.g., an Identity Provider, a mail server) to verify its effectiveness. Otherwise the same CS could be able to maintain user accounts and verifying the identity with a password request.

The CS makes the $UserId$ verification procedure (whose steps depend on the nature of the $UserId$ itself), and then binds the user identity with his public key and with a $NodeId$ by producing the following token:

$$AuthId_A = Sign(NodeId_A||UserId_A||K_A^+||exp_A, K_{CS}^-)$$

The $NodeId$ is randomly chosen; exp_A is a timestamp that establishes the expiration date of the signed $NodeId_A$. The CS keeps track of the association between $UserId$ and $AuthId$, so that all subsequent $NodeIdReq$ received by the same users receive in response the same $AuthId$ passed earlier, unless it is expired or close to expiration. This is a precaution to avoid the CS producing useless signatures. Then, the CS sends to the client a response message structured as follows:

$$CS \rightarrow A : AuthId_A, Sign(bootstrapList, K_{CS}^-)$$

The $bootstrapList$ is a list of triple $< NodeId, IP, port >$ that points to a set of nodes that the CS assumes active; by contacting at least one of these nodes, the peer can join the network. The way the CS obtains the entry of bootstrap list is described in Section 5.1.5.

5.1.2 Join

Once initialization step is completed, the node may initiate the network join procedure as described by the Kademlia protocol, namely sending a lookup request for its own $NodeId$ to one of the bootstrap contacts. However, once obtained an $AuthId$, it is important that the nodes avoid contacting the certification service, unless if necessary. After making the first join using information obtained from the $bootstrapList$, each node should get in a different way a list of nodes to be contacted for subsequent join operations. For example a node can maintain its own list of trusted bootstrap nodes, or the same CS could periodically insert a signed $bootstrapList$ in the DHT, so that every active node could download it before disconnection and use it for its next join. Only if all the known nodes are off-line the CS will be contacted again to request a new $bootstrapList$.

The messages sent by the nodes during the join procedure must follow, like any other message, the protocol described in the next paragraph.

5.1.3 Nodes interaction

A node A can successfully send a RPC (join primitive included) to a node B and obtain a proper response only if both A and B observe the following communication protocol:

I $A \rightarrow B : NodeId_A, N1$

II $B \rightarrow A : NodeId_B, N2$

III $A \rightarrow B : AuthId_A, Auth_{AB},$ RPC-REQ

IV $B \rightarrow A : AuthId_B, Auth_{BA},$ RPC-RES

We call this four way exchange a $session$ between A and B. RPC-REQ and RPC-RES fields are respectively the request and response RPC defined in Kademlia; $N1$ and $N2$ are randomly generated nonces. Messages sent at steps I and II must be somehow marked differently (e.g., different opcode), to distinguish the request from the response.

Authentication tokens are structured as follows:

$$Auth_{AB} = Sign(NodeId_B||N2||H(\text{RPC-REQ}), K_A^-)$$
$$Auth_{BA} = Sign(NodeId_A||N1||H(\text{RPC-RES}), K_B^-)$$

In step III (and IV), the receiving node checks signatures (in $AuthId$ and $Auth$), expiration times validity, equalities between nonces, and equalities between $NodeId$ in step I (and II), in $Auth$, and in $AuthId$.

In steps III and IV, the receiving node performs the following controls:

1. Validity of $AuthId$ signature
2. Validity of $AuthId$ expire time
3. Validity of $Auth$ signature
4. Equality between the $NodeId$ contained in the $Auth$ and the receiver's $NodeId$
5. Equality between the nonce contained in the $Auth$ and the nonce sent previously
6. Equality between the $NodeId$ contained in the $AuthId$ and the $NodeId$ received at step I or II
7. Check of the RPC hash

Signature and expiration time validity checks on $AuthId$ demonstrate the existence of a valid and randomly generated $NodeId$, associated with an $UserId$ and with a public key; validity of signature in $AuthId$ and equality check on $NodeId$ assures that the sender is the same entity certified by $AuthId$ and that the present node is the correct recipient of the message. Equality checks on nonces in $Auth$ and the ones received previously protect against replay attacks. A's verification of $NodeId_B$ included in $AuthId_B$ assures that B is really the node that A wanted to contact; B's verification of $NodeId_A$ included in $AuthId_A$ proves that the RPC has been called by the same node that started the session. Finally, both peers execute an integrity check on the RPC hash to verify that no attacker has replaced the original RPC with a bogus one. The reader should observe that nonces are used against man in the middle attacks instead of exchanging timestamps because we cannot assume that hosts are synchronized to a common clock.

5.1.4 Content storage system

RPCs follow Kademlia's definitions, except for the store RPC. Let A be a node, owner of a content Obj. If A wants to store Obj in the DHT it locates via lookup the k nodes closest to the content key and then sends to them a store message structured as follows (suppose that B is a generic replica node):

$$A \rightarrow B : AuthId_A + Auth_{AB} + StoreRPC$$
$$StoreRPC = k||Obj||Cred$$
$$Cred = Sign(UserId_A||k||H(Obj)||ts||TTL, K_A^-)$$

$Cred$ binds the $UserId$ to the key for which the content was inserted and to the hash code of the content, so that is subsequently possible to prove that the owner had inserted the content Obj at the key k. $Cred$ includes also a timestamp and a time to live to specify the content submission time and its persistence period. During the periodic content spreading procedure, all replica nodes send store messages keeping the original credentials associated with each content. A node performing a lookup for contents related to a key χ receives all the objects marked with χ from replica nodes responsible for that key; before passing the content to the application, the node must verify the credentials signature and the object hash and must discard the object if the check fails.

If the application ascertain that the content is somehow polluted (e.g., the key that marks the content is not related with it), it can benefit from the information included in the credentials to penalize the owner of the content. This could be simply accomplished by instructing the underlying node to blacklist the cheater user in order to refuse all the incoming requests marked with the malicious node's $AuthId$. The description of a reputation service that can manage feedbacks from the users and the details concerning a possible revocation policy for the identifiers of misbehaving users, are beyond the goal of this paper. However, it is important to say that an effective reputation manager, that can be external to the network, as well as integrated in the application, can help to exclude more rapidly the polluter from the whole network. Nevertheless, the propagation of polluted content is largely limited due to credentials' verification.

5.1.5 Bootstrap list construction

The bootstrap node selection is a problem inherent to the fully distributed nature of P2P networks. The bootstrap information acquisition process must prevent an attacker to manipulate bootstrapping information to let a victim join a malicious parallel network. Kademlia does not face the bootstrap node selection problem.

The CS maintains a list of active peers in a cache, where a generic entry stores the following information:

$$CacheEntry = (NodeId, IPaddress, UDPport, ts)$$

The CS probes nodes in the cache, controlling a DHT node, marked with a self signed $AuthId$, that runs a sequence of FIND-NODE RPCs for random generated keys. The CS adds to its cache the pointers to the nodes that replied to the FIND-NODE RPCs, then it can iterate the procedure until it gathers enough contacts for cache replacing. A least-recently cache replacement policy is implemented, except that active nodes are never removed from the list: if the cache is full then the least-recently seen node is pinged. If it fails to respond, it is replaced with a newly discovered one. Otherwise, if the least-recently seen node responds, it is moved to the tail of the list, and the new contact is discarded.

5.2 Discussion

In this section we discuss how this proposal strongly limits dangerousness of attacks described in Section 2.2.1.

Routing attacks In Kademlia, the sender contact of every incoming message is added to the route table if there is enough room in the buckets. The contacts with a nodeId close to the local id are always added to the route table due to the splitting procedure. To effectively put off a routing attack, the attacker must inject bad routing information in the target node by sending him messages that report sender ids near to the victim's id. Combined usage of $AuthId$ and $Auth$ makes the communication between nodes authenticated, so the attacker can inject only its own contact into the target route table, and because the ids are randomly chosen by the CS, the attacker cannot generate its id "ad hoc". Routing attacks (including eclipse) are unfeasible. Moreover, it is unfeasible for an attacker to hide a content marked with a given key k by way of a node insertion attack, because the malicious node cannot register a substantial number of nodes with IDs close to k: in fact, he cannot control id generation by his own.

Kademlia's lookup vulnerability is corrected by authenticated message exchange and random id generation. If the malicious FIND-NODE RPC receiver responds with a set of references to invalid nodes (i.e., devoid of $AuthId$s), the victim node is not able to contact any of them because the authentication protocol fails in signature verification. If the attacker responds with a set of valid colluding nodes, its attack results ineffective because the colluders' ids are scattered along the keyspace, so the lookup procedure proceeds properly.

Sybil attack Every user can have multiple identities (e.g., many email addresses), so a user can bind each of his identities to a different node by sending many $NodeIdRequest$ to the CS, and then he can run all those nodes on the same machine. So the Sybil attack is not completely wiped out with this scheme. Nevertheless, each node corresponds to a different user account and the node initialization requires a verification procedure for that account. If the user authentication procedure requires a human interaction it would be difficult for an attacker to create many different nodes in an automated way, actually lowering the risk of Sybil Attacks. For this reason, we strongly suggest to adopt OpenId verification methods, that redirect the user agent to an identity provider, and that returns to the CS when the submitted identity has been correctly authenticated.

Storage attacks Every storage entry in the DHT is bound with its $Cred$, created by the content owner with an unforgeable signature. A node performing a lookup operation returns to the application only those results that are bound with some $Cred$, and that has been previously verified. Therefore, the consumer application (or the human user himself) can interact with a reputation system to reward or penalize the owner of the consumed object depending on the quality of the content. The underlying node can be then instructed to exclude from network traffic those nodes whose reputation is too bad. The use of $Cred$ can contrast attacks like index poisoning, content pollution or even DDoS attacks based on redirection by punishing the malicious users who attempt these attacks.

Man in the middle attack An attacker who's able to intercept and alter the messages flowing between two nodes has no way to act as one of the endpoints or to fool correct nodes into accepting forged messages. $AuthId$ and $Auth$ cannot be modified since they are signed, and the RPC cannot be altered or replaced because the Authenticator contains the RPC hash code. An attacker cannot effectively replay an intercepted $Auth$ because it includes a nonce which validity is limited to a node interaction session; moreover authenticators are addressee-specific, because they include the recipient node ID. Finally, the nonce based two-way authentication scheme grants protection against common interleaving attacks as Oracle session attacks, parallel attacks and offset attacks.

6 Prototype

In this Section we describe the main functionalities of the proposed application, according to the system architecture depicted in Figure 3. In the implemented prototype the main application interacts with the user components described in Section 3. As already mentioned above, we assume that the content indexed and retrieved in the P2P network is always governed, requiring the adoption of an appropriate format which is able to provide a full description of the content. We used the MPEG-7 metadata for the multimedia content representation and MPEG-21 metadata to express the digital rights. Moreover we also consider protected contents, obtained by applying encryption tools for DRM. According to Chillout reference implementation [5], we make use of the DCI and DCF formats, already discussed in Section 2. The user is able to search for a subset of relevant MPEG-7 and MPEG-21 metadata extracted from DCF and at the same time the whole DCF can be accessed in a completely distributed way, by means of a *DCFMetadataService* protocol provided by the *Transport* component (see Figure 3), implemented for building the mapping between the subset of indexed keys, inserted in a structured way, and the DCF it refers to.

We built our system upon Kademlia but, exploiting the separation level between the DHT implementation and the applications, the framework provides the possibility of using other DHTs. This level exports to upper modules the insertion of new mappings and the retrieval of the *key's root* functionalities. In order to communicate with the DHT module (see Section 2.1),we made use of Java *Future* objects for non-blocking asynchronous insertion and querying operations, that were introduced in the Java Development Kit from version 5[4]. We used a *CollectorParameter* design pattern in order to collect the results provided by the Future objects. The main reason is that the communication works asynchronously because it has to take into account the network latency and topology reconfiguration due to the peers joining and leaving the overlay. The *Transport* component is not put directly on top of the DHT (see Figure 3): they communicate in order to get the information about the two (or more, e.g., for multisource download) endpoints of the direct connection established for downloading the DCF. This module is composed by two subcomponents and the Figure 3 is showing the two socket listeners: one is responsible for transferring the resources (multimedia files) and the other is responsible for transferring the related metadata, actually a Java object which wraps the DCF file.
The *Application* component exports a set of high level functionalities in order to join/leave the system and to insert/retrieve the DCF files. As already described, it makes use of a DCI and DCF wrapper for parsing the digital content files and for extracting the metadata to be indexed.
The insertion of a content proceeds as follows: the *Content Creator* component is responsible for creating the DCI and the DCF. The user can choose one or more resources to be published in the P2P network in a single DCF file and can associate to each resource a different license, which can be completely customized for different purposes. It is worthwhile noticing that some resources in the DCF file could be also encrypted to ensure that even if they are retrieved from the P2P network the consumption of the content is possible only to the principal specified in the license. Once the DCF (or simply the DCI) is created, it can be shared on the structured peer-to-peer network. Concerning the content retrieval, the lookup operation on the DHT could be done by simple keywords or structured bundle of MPEG-21 REL tags, resulting, at low level, in the index of the content whose DCF (DCI) is fulfilling the request. The Application module contacts then the publishing sources (peers) asking for more information about the content. Every user can check the license conditions associated to a given content before downloading it. The *Transport* component communicates by means of a *DCFMetadataService* on a separate channel (socket), with a specific protocol which is able to exchange the wrapper of the DCF. In this way we can provide to the user all the available metadata related to the searched keywords and grants. The results of the query are collected by means of the *CollectorResults*, which generates a separate thread looking for the asynchronous return messages. The user can select the specific content from the result list and the *Application* component will contact the specific owner source (the <IP address,port> pair), through the *FileTransportation* component (see Figure 3) which communicates using a separate channel (socket), with a specific protocol for exchanging files. Our first approach has been the adoption of a simple file transportation but a possible improvement could come from making use of more sophisticated solutions enabling the multi-source download, as the BitTorrent [40] exchange protocol.

7 Conclusions and future works

We have described a decentralized, distributed and secure communication infrastructure for indexing and retrieving multimedia contents with associated digital rights. We have discussed a feasible approach to share digital items accord-

[4]http://java.sun.com. Last visited: 15 Nov 2008

ing to the associated license, making use of a P2P routing infrastructure based on DHT. Complex queries on standard MPEG-7 and MPEG-21 multimedia metadata are supported.

Concerning the future works, in order to express queries in a standard format, we will evaluate the use of MPEG Query Format (MPQF) [41], part 12 of the MPEG-7 standard, whose reference software implementation is currently under development by people involved in the MPEG consortium. MPQF lets us also investigate novel approaches for searching digital contents on peer-to-peer infrastructure, as *range* and *by feature* queries that could be introduced into a future prototype. Moreover, we plan to evaluate the Java implementation upon a live large-scale testbed like PlanetLab[5] in order to test the efficiency, scalability and reliability properties in a real scenario.

8 Acknowledgements

This work was partially supported by the SAPIR project[6], funded by the European Commission under IST FP6 (Contract no. 45128) and by the Ministero Italiano per l'Università e la Ricerca (MIUR) within the framework of the PRIN "PROFILES" project[7].

References

[1] MPEG-7 - ISO/IEC 15938 - Information Technology Multimedia Content Description Interfaces. `http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm`. Last visited: 15 Nov 2008.

[2] MPEG-21 - ISO/IEC 21000 - Information Technology Multimedia Framework. `http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm`. Last visited: 15 Nov 2008.

[3] MPEG-21 Rights Expression Language - ISO/IEC 21000-5 - Information Technology Multimedia Framework. `http://www.chiariglione.org/mpeg/technologies/mp21-rel/index.htm`. Last visited: 15 Nov 2008.

[4] Digital Media Project (DMP). `http://www.dmpf.org`. Last visited: 15 Nov 2008.

[5] Chillout. The Reference Software for DMP Interoperable DRM Platform. `http://chillout.dmpf.org`. Last visited: 15 Nov 2008.

[6] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[7] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *IPTPS '02: Proceedings of 1st International Workshop on Peer-to-Peer Systems, Cambridge, MA, USA*, 2002.

[8] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany*, pages 329–350, 2001.

[9] Klaus Wehrle, Stefan Götz, and Simon Rieche. Distributed Hash Tables. In *Peer-to-Peer Systems and Applications*, pages 79–93, 2005.

[10] J. Liang, N. Naoumov, and K. W. Ross. The index poisoning attack in p2p file sharing systems. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[11] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. *SIGMETRICS Perform. Eval. Rev.*, 33(1):38–49, 2005.

[12] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A global view of kad. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 117–122, New York, NY, USA, 2007. ACM.

[13] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Access control in ad hoc groups. In *HOT-P2P '04: Proceedings of the 2004 International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 2–7, Washington, DC, USA, 2004. IEEE Computer Society.

[14] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Identity-based access control for ad hoc groups. In *Information Security and Cryptology (ICISC)*, pages 362–379, 2004.

[15] John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[16] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Analyzing peer behavior in KAD. Technical Report EURECOM+2358, Institut Eurecom, France, Oct 2007.

[5]http://www.planet-lab.org. Last visited: 15 Nov 2008
[6]http://www.sapir.eu. Last visited: 15 Nov 2008
[7]http://dit.unitn.it/profiles. Last visited: 15 Nov 2008

[17] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting kad: possible uses and misuses. *SIGCOMM Comput. Commun. Rev.*, 37(5):65–70, 2007.

[18] George Danezis, Chris Lesniewski-Laas, Frans M. Kaashoek, and Ross Anderson. Sybil-resistant dht routing. In *ESORICS*, volume 3679 of *LNCS*, pages 305–318. Springer, 2005.

[19] Hosam Rowaihy, William Enck, Patrick Mcdaniel, and Thomas La-Porta. Limiting sybil attacks in structured peer-to-peer networks. Technical Report NAS-TR-0017-2005, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, 2005.

[20] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[21] I. Baumgart and S. Mies. S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing. In *Proc. of P2P-NVE 2007 in conjunction with ICPADS 2007, Hsinchu, Taiwan*, volume 2, December 2007.

[22] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *IPTPS*, pages 53–65, 2002.

[23] Tyson Condie, Varun Kacholia, Sriram Sankararaman, Joseph M. Hellerstein, and Petros Maniatis. Induced churn as shelter from routingtable poisoning. In *In Proc. 13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.

[24] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[25] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):299–314, 2002.

[26] Yongdae Kim, Daniele Mazzocchi, and Gene Tsudik. Admission control in peer groups. In *NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, page 131, Washington, DC, USA, 2003. IEEE Computer Society.

[27] Fotis G. Kazasis, Nektarios Moumoutzis, Nikos Pappas, Anastasia Karanastasi, and Stavros Christodoulakis. Designing Ubiquitous Personalized TV-Anytime Services. In *CAiSE '03: Proceedings of the 15th Conference on Advanced Information Systems Engineering, Klagenfurt/Velden, Austria*. CEUR-WS.org, 2003.

[28] Rubén Tous and Jaime Delgado. L7, An MPEG-7 Query Framework. In *AXMEDIS '07: Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, Barcelona, Spain*, pages 256–263. IEEE Computer Society, 2007.

[29] Walter Allasia, Francesco Gallo, Filippo Chiariglione, and Fabrizio Falchi. An Innovative Approach for Indexing and Searching Digital Rights. In *AXMEDIS '07: Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, Barcelona, Spain*, pages 147–154. IEEE Computer Society, 2007.

[30] CreativeCommons. `http://creativecommons.org`. Last visited: 15 Nov 2008.

[31] MPEG-A - ISO/IEC 23000-5 - Information Technology Multimedia Application Format, Part 5: Media Streaming Application Format. `http://www.chiariglione.org/mpeg/standards/mpeg-a/mpeg-a.htm`. Last visited: 15 Nov 2008.

[32] Digital Media Project (DMP). Approved Document No. 3 - Technical Specification: Interoperable DRM Platform, Version 3.0 - 1003/GA15. `http://www.dmpf.org/open/dmp1003.zip`. Last visited: 15 Nov 2008.

[33] MPEG-21 Rights Expression Language - ISO/IEC 21000-5 Amendment 3 : the OAC (Open Access Content) profile.

[34] MPEG-21 Rights Expression Language - ISO/IEC 21000-5 Amendment 2: the DAC (Dissemination And Capture) profile.

[35] TV-Anytime. `http://www.tv-anytime.org`. Last visited: 15 Nov 2008.

[36] Unified Modeling Language. UML 2.1.1 - UML 1.4.2 (ISO/IEC 19501). `http://www.uml.org`. Last visited on Nov 15th 2008.

[37] W. Allasia, F. Gallo, M. Milanesio, and R. Schifanella. Governed content distribution on dht based networks. *Internet and Web Applications and Services, 2008. ICIW '08. Third International Conference on*, pages 391–396, June 2008.

[38] Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. A Totally Distributed Iterative Scheme for Web Services Addressing and Discovery. In

PDCS '07: Proceedings of the 19th IASTED International Conference on Parallel and Distributed Computing Systems, Cambridge, MA, USA, pages 85–90. ACTA Press, 2007.

[39] L. M. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella. Tempering kademlia with a robust identity based system. In *Proc. of 8th International Conference on Peer-to-Peer Computing 2008 (P2P'08)*, 2008.

[40] BitTorrent. `http://www.bittorrent.com`. Last visited: 15 Nov 2008.

[41] MPEG Query Format. `http://dmag.upf.edu/mpqf`. Last visited: 15 Nov 2008.