ARES project CONSOLIDER-INGENIO 2011 CSD2007-00004 Workpackage 2 - Task 2 (WP2.T2) Completion of the design and implementation of random node identity assignment techniques in structured p2p networks

Oscar Esparza, Jose L. Muñoz, Juan Caubet, Juan Vera del Campo, Carlos Gañán, Sergi Reñé, Joan Arnedo-Moreno, Joaquín García-Alfaro, Helena Rifà-Pous

Universitat Politècnica de Catalunya. Departament Enginyeria Telemàtica.
1-3 Jordi Girona, C3 08034 Barcelona (Spain)
Universitat Oberta de Catalunya. Estudis d'Informàtica, Multimèdia i Telecomunicació.
Rambla de Poblenou, 156 08018 Barcelona (Spain)
{oscar.esparza, jose.munoz, juan.caubet, juanvi, carlos.ganan, sergi.rene}@entel.upc.es
jarnedo@uoc.edu joaquin.garcia-alfaro@acm.org hrifa@ouc.edu

March 22, 2011

Abstract. This report summarizes the advances carried out for task WP2.T2 within the ARES project to improve the identity assignment in structured p2p networks. The report describes the works that have been performed by the ISG group of the UPC and the KISON group of the UOC to justify the task WP2.T2 "Completion of the design and implementation of random node identity assignment techniques in structured p2p networks". These works are mainly related to identity issues in p2p networks. In addition, both research groups have made some extra research regarding other topics related to identity issues, like providing anonymity and privacy, or deploying trust and reputation systems. We also present the prototype of the Secure Cooperative File System (SCFS), accessible under the GPL license. A list of published works and conclusions is provided.

Publications

The contributions by members of ARES related to task WP2.T2 that have been published or that are currently accepted to be published are listed below.

ISI-JCR Journals

1. J. Arnedo-Moreno and J. Herrera-Joancomartí. JXTA resource access control by means of advertisement encryption. *Future Generation Computer Systems*, 26(1):21–28, 2010.
2. Marcela Mejia, Néstor Peña, José. L. Muñoz, and Oscar. Esparza. A review of trust modeling in ad hoc networks. *Internet Research*, volume 19(1):88–104, 2009.
3. Marcela Mejia, Néstor Peña, José. L. Muñoz, Oscar. Esparza, and Marco-Aurelio Alzate. A game theoretic trust model for on-line distributed evolution of cooperation in manets. *Journal of Network and Computer Applications*, volume 34(1):39–51, 2011.
4. Muñoz, J. L. and Esparza, O and Aguilar, M and Carrascal, V and Forné, J. Rdsr-v. reliable dynamic source routing for video-streaming over mobile ad hoc networks. *Computer Networks*, 54(1):79–96, 2010.
5. H. Rifa and J. Herrera. A fair and secure cluster formation process for ad hoc networks. *Wireless Personal Communications*, 56(3):625–636, 2010.

Lecture Notes in Computer Science

1. S. Castillo-Perez and J. Garcia-Alfaro. Anonymous resolution of dns queries. In *International Workshop on Information Security (IS'08), International OTM Conference*, volume 5332 of *LNCS*.

Conferences and non ISI-JCR Journals

1. J. Arnedo-Moreno and J. Herrera-Joancomartí. XML-based security for JXTA core protocols. *International Journal on Autonomic Computing, Special issue on Efficient Techniques for autonomic resources scheduling and management in Grid and P2P systems*, 1(4):447–469, 2010.
2. S. Castillo-Perez and J. Garcia-Alfaro. Resolucion de consultas anonimas sobre dns. In *X Reunión Espanola sobre Criptología y Seguridad de la Información (RECSI'08)*, 2008.
3. S. Castillo-Perez and J. Garcia-Alfaro. Evaluation of two privacy-preserving protocols for the dns. In *In* 6th International Conference on Information Technology: New Generations (ITNG 2009)*, pp. 411-416.*, 2009.
4. Juan Vera del Campo, J. Hernández-Serrano, and J. Pegueroles. Análisis de seguridad de un sistema de archivos distribuido. X Reunión Espanola sobre Criptología y Seguridad de la Información (RECSI'08), 09/2008 2008.
5. Juan Vera del Campo, J. Hernández-Serrano, and J. Pegueroles. Scfs: Towards design and implementation of a secure distributed filesystem. International Conference on Security and Cryptography (SECRYPT), International Conference on Security and Cryptography (SECRYPT), 07/2008 2008.

6. L. Barolli F. Xhafa J. Arnedo-Moreno, K. Matsuo. Persistent interoperable security for JXTA. In *Proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC)*, pages 354–359, 2008.
7. L. Barolli F. Xhafa J. Arnedo-Moreno, K. Matsuo. Securing a java p2p framework: The JXTA-Overlay case. In *Proceeedings of the 11th International Conference on Integration and Web-based Applications & Services (iiWAS'09)*, pages 158–165, 2009.
8. L. Barolli F. Xhafa J. Arnedo-Moreno, K. Matsuo. A security-aware approach to JXTA-Overlay primitives. In *Proceeedings of the 3rd International Workshop on Advanced Distributed and Parallel Network Applications (ADPNA'09)*, pages 431–436, 2009.
9. M. Barbeau J. Garcia-Alfaro and E. Kranakis. Evaluation of anonymized ons queries. In *Workshop on Security of Autonomous and Spontaneous Networks (SETOP 2008)*, pages 47–60, 2008.
10. J.L. Munoz M. Rodriguez-Perez, O. Esparza. Surework: Un sistema de reputación para redes p2p basado en super-peers. In *VI Jornadas de Ingeniería Telemàtica (JITEL'07)*, 2007.
11. J.L. Muñoz M. Rodriguez-Perez, O. Esparza. Surework: a super-peer reputation framework for p2p networks. In *23rd Annual ACM Symposium of Applied Computing (SAC)*, 2008.

PhDs and Books

1. Marcela Mejia. *Evolución Genética de Estrategias para Modelos de Confianza en Redes Móviles Ad Hoc Basados en Teoría de Juegos.* PhD thesis, Universitat Politècnica de Catalunya (UPC, Spain) and Universidad de los Andes (Colombia), 2011.

1 Introduction

Peer to peer (p2p) systems emerged as an incipient paradigm of communications which allow sharing resources without the need of centralized servers. First generation p2p networks were also known as *centralized p2p networks* because they still depend on centralized entities to perform some functions, for instance searching resources. Some examples of this kind of networks are the initial music sharing service Napster [1], and the file sharing network eDonkey2000 (ed2k) [2]. One of the problems of centralized p2p networks is that they have a single point of failure. To alleviate this problem, second generation p2p networks were designed completely decentralized. An example of such networks is Gnutella [3]. These networks are characterized by the arbitrariness of the links between nodes, and the use of flooding of messages to search resources. These decentralized unstructured networks are easy to implement and very robust. However, the use of broadcasting techniques limits their scalability. The next step in p2p systems was to structure their resources, so that searches can obtain positive results in a certain number of hops. This third generation p2p networks are called *structured p2p networks* or *p2p overlays*, and they do not allow random connections between nodes. These systems create an *overlay*, that is, a logic high-level layer built on other existent networks, and which is used to structure the nodes and the connections between them. The routing protocol uses logical identifiers of nodes (*nodeIDs*) to decide the routing on the overlay, instead of using directly the IP addresses of the nodes.

Most typical structured p2p overlays are implemented using a Distributed Hash Table (DHT) [4]. The DHT is a distributed table that associates pairs of *keys* and *values* and creates a virtual space that contains all these possible pairs. A *value* can be a certain resource (for instance a file), or the way to reach this resource in the overlay. *Keys* are used to locate these resources into the network. The DHT is divided in subtables, which correspond to a zone of the virtual space. These subtables (and their corresponding zones) are assigned to the nodes of the overlay. This way, each node is responsible of one zone, and hence it is responsible for the key/value pairs included in the subtable that have been assigned to this zone. Usually, a zone is assigned to the node whose *nodeID* is close to the key values stored in a certain subtable of the DHT.

Some examples of typical p2p overlays are CAN [5], Chord [6], Pastry [7], Tapestry [8], Kademlia [9] and JXTA [10].

Unlike earlier systems, some mechanisms of structured p2p overlays have been analyzed in depth to guarantee the scalability and efficiency. However, less attention has been paid to security mechanisms. Most p2p overlays presuppose that peers behave honestly, but this assumption is not acceptable in open environments. The existence of anonymous peers and the lack of a centralized authority capable of monitoring or punishing make these systems more vulnerable against dishonest or malicious behaviors. These improper usages cannot be faced with classical security solutions. P2p overlays should follow the secure routing primitives described by Wallach in [11], which are: (1) secure maintenance of routing tables, (2) secure routing of messages, and (3) secure identity assignment to nodes. In fact, the first two problems depend in some way on the third one. For instance, routing tables are constructed using routing information exchanged between nodes, but if we are not able to assign stable identifiers to nodes, it is difficult to tell the difference between honest from malicious routing information. Other problems and solutions have been stated in the literature [12, 13].

Therefore, it is important to prevent uncontrolled identity changes. Users must be sure that the others' identities are valid and stable; a user must always have the same identifier and not be able to change it. However, in practice, the overlays are vulnerable to *Sybil attacks* [14] if they do not implement the proper identity assignment and access control mechanisms. A Sybil attack involves managing multiple identifiers by the same (malicious) node, which increases its presence in the network simulating several nodes. The node that applies the Sybil attack can degrade the system performance or it can cause serious routing problems. For instance, an attacker can use multiple identities in order to encircle a victim node and control the communications to-and-from it, or even to censor a target document. In [14], Douceur shows the impossibility to discern whether the same entity controls several identifiers or not. He also concludes that a trusted entity that certifies identities is the only solution to avoid Sybil attacks.

Some authors have addressed the problem of assigning robust identifiers to nodes. Some of these proposals have focused their efforts on the context of anonymity and reputation [15–18]. Some others try to detect Sybil attacks, or to limit the scope of such attacks [19, 20]. However, less attention has been paid to the way that the identifiers of nodes should be constructed in a p2p overlay, or how to make access control mechanisms more robust. In our opinion, a robust identification system is necessary

to guarantee that the overlay provides the main security services, and in particular secure routing mechanisms.

One of the main goals of the ARES project related to the p2p networks was to address these two problems: identity and access control. In this deliverable we summarize the works that have been performed by the ISG group of the UPC and the KISON group of the UOC to justify the task WP2.T2 "Completion of the design and implementation of random node identity assignment techniques in structured p2p networks". These works are mainly related to identity issues in p2p networks. In addition, both research groups have made some extra research regarding other topics related to identity issues, like providing anonymity and privacy, or deploying trust and reputation systems.

To be more specific, the objective of both groups was to improve the security of a generic p2p overlay. In particular, JXTA [10] has been chosen as this overlay because it is a project that has been under active work and continuous revisions by the scientific community. JXTA has still major lacks regarding security, and for that reason the KISON group has designed a new security layer, common to all of JXTA's core protocols. This new security layer allows peers which do not support signature to process messaging in a transparent way. Apart from this, extra efforts have been made to secure resource access control and distribution in the JXTA network. Once the security of JXTA has been improved, both groups have addressed the problem of assigning secure identities to peers. The KISON group has designed identities based on CBIDs (Crypto-based IDs), which avoids the intervention of a TTP. Also, the ISG group has designed a new protocol to deliver anonymous identifiers by means of two TTPs.

Other extra works related in some way to identity issues were made by both research groups. The KISON group has evaluated the risk of privacy leaks in p2p services as a consequence of the dependency of p2p communications to the current Internet Domain Name resolution System (DNS) protocol. The ISG group has designed and simulated reputation and trust models for the p2p and MANET scenarios.

The ISG group of the UPC also presents the prototype of the Secure Cooperative File System (SCFS), a secure architecture for the storage of personal data in distributed peer-to-peer networks. The prototype is accessible under the GPL license.

2 Securing a generic p2p middleware

In this Section we explain the works that have been made to improve the overall security of JXTA and to address the problem of assigning secure identities to peers.

2.1 The JXTA Project

The main research line at UOC on regards to p2p networking has focused on the JXTA open protocol specification [10], developed by SUN in 2001. Since then, it has been under active work and continuous revisions, the latest update, version 2.7RC, being dated January 2011.

JXTA Overview The fundamental JXTA architecture is divided into three distinct layers, as shown in Figure 1.

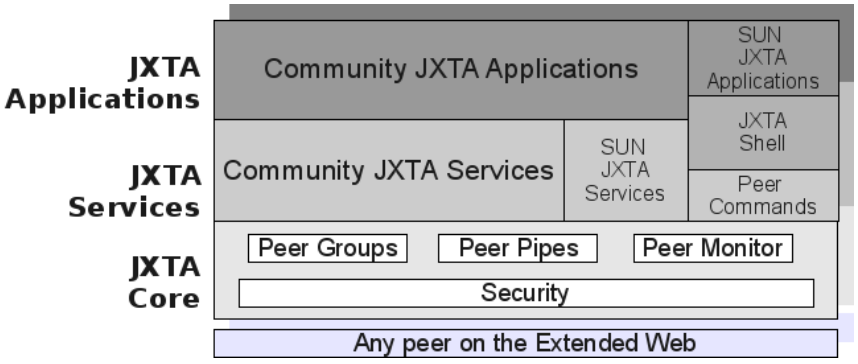


Fig. 1: JXTA's layered architecture

The *Core* layer contains the minimum and essential characteristics, common to all p2p networks. Such operations include peer creation, discovery and communication (even when behind firewalls or NAT), as well as basic security services.

The *Services* layer includes all network services which are not absolutely necessary, but provide desirable capabilities such as resource search, indexing, storage and sharing.

The top layer is the *Applications* one, which includes any application deployed using the JXTA framework, such as instant messaging, file sharing or content management.

Notice that the distinction between services and final applications may not always be clear, since what a client may consider an application may be considered a service by another peer. For that reason, the system is designed in a modular way, letting developers to choose the set of services and applications which most satisfy their needs. All JXTA components are within these three layers.

JXTA differs from other p2p protocols because the concept of *peer group* is introduced, one of the main characteristics of its architecture. Usually, p2p environments are conceptualized as a global overlay network without any kind of logical segmentation or segregation as far as resource availability is concerned. However, in JXTA, the global overlay network is segmented into hierarchical peer groups, which offer a context for accessing services. Peers may access a specific resource only if they previously join the group that offers such service. The concept of peer groups is very important, since all JXTA security services are ultimately related to peer group membership management.

Protocols. JXTA defines a set of protocols (six, specifically) which enable the deployment of p2p networks. Using these protocols, peers may collaborate in a fully autonomous manner by publishing and discovering available resources within the network. All JXTA protocols are asynchronous and based on a request/response model, which means that for any given request, zero, one or many responses may be received. A brief description of each protocol is given:

- The *Peer Discovery Protocol (PDP)* allows peers to publish their own resources and make them available to other peers. Any kind of peer may send PDP messages. This protocol is the default discovery protocol, but it is possible for applications to implement and deploy their own protocols.
- In order to obtain information about other peers, the *Peer Information Protocol (PIP)* is used. Using this protocol, it is possible to query peer capabilities or monitor its behavior.
- Peers use the *Peer Resolver Protocol (PRP)* in order to send requests to one or several peers and manage responses. The PRP protocol uses a unique ID associated to every request, which is included in messages. Other core protocols, such as PDP, make use of PRP in order to create its own requests.
- The *Pipe Binding Protocol (PBP)* establishes virtual communication channels between peers named *pipes*, acting as abstract endpoints above any physical network.

- Routes between peers are found with help of the *Endpoint Routing Protocol (ERP)*. Whenever some peer is about to send a message to a destination but does not know any path, an ERP message is sent to other peers asking whether they know a path.
- Finally, the *Rendezvous Protocol (RVP)* is responsible for the efficient propagation of messages within a group of peers, allowing peers to connect to services and exchange messages.

Core JXTA protocols shortcomings. the current security layer in core protocol messaging has three main shortcomings which could be improved:

- Lose of interoperability: Even though JXTA makes heavy use of XML in its protocols, no XML standard is used for signature generation at the messaging layer, which may become a hurdle if application interoperability has to be maintained. Signatures are generated by serializing XML as plain text, which is not ideal for XML processing. Furthermore, under the current secure protocols, endpoints must support and agree to use the provided security layer in order to communicate. It is not possible for a peer which chooses not to deploy the security layer (for example, because of computational limitations) to understand received messages.
- No confidentiality at the messaging layer: Currently, the only way to achieve data confidentiality is using TLS. However, since TLS is a wire transport protocol, it imposes a constraint that cannot be ignored: no other transport protocol may be used underneath. That means that it is not possible to transparently secure neither other current JXTA wire transport protocols such as message propagation via multicast or HTTP proxying, nor any future ones (for example, UDP or RTP).
- TTP-based trust model: The use of TLS forces peer group management via the PSE Membership Service. PSE provides an integrated secure environment in JXTA, but for some applications it may become too restrictive by constraining the peer group to use X509 certificates and a TTP based trust model. This is not always desirable in a dynamic and decentralized environment such as peer-to-peer, especially when trying to maximize peer equality and self-organization. Furthermore, the use of a TTP inherits additional problems which increase system's complexity like, for instance, certificate chain management and revocation.

Work on JXTA protocol improvements. In order to solve these issues, we extended the protocols with a security layer, common to all of JXTA's

core protocols. We defined a data encryption and signature format for message elements based on XML standards, making use of the standard JXTA messaging signature capabilities at the Messaging layer. Using this method, the message format, as defined in the JXTA v2.0 protocols specification [21], is maintained.

This approach allows peers which do not support signature to process messaging in a transparent way. As a result, each peer may choose its own degree of security without being constrained by other peer's decision on that regard. Due to JXTA's layered protocol architecture, deploying security at the core layer ensures that this proposal may easily integrate with existing applications, since this layer is completely transparent to application data. This approach also allows different secure protocols to coexist, both current ones, as well as future proposals, so applications may choose the one that suits its specific needs.

The results of the work were published in [22].

The JXTA-Overlay subproject Several subprojects have been developed under the umbrella of the JXTA specification, improving its breadth of services or performance at different layers. An important project which tries to achieve both goals is JXTA-Overlay [23]. This project was deemed a good testbed. Therefore, the UOC research team worked on helping improve this project, in close collaboration with the Fukuoka Institute of Technology (FIT).

JXTA-Overlay, increases the reliability of JXTA-based distributed applications and supports group management and file sharing. However, the design focus on JXTA-Overlay was completely concerned with system performance, but not at all with security, a situation which may become a great constraint under today's standards. Even though JXTA provides some basic security mechanisms, they were not taken into account.

Network Entities overview. The improvements are attained by further structuring the JXTA network according to the following entities:

End-users connect to the JXTA-Overlay network by authenticating using a username and password. Once the authentication process is successfully completed, they are organized into different overlapping groups, so only members of the same group may interact. It is also important to take in to account that JXTA-Overlay end-users are mobile, they may connect at different times using different client peers, as well as may be

connected through several client peers at the same time.

A *client peer* represents an application, which end-users use to communicate and share resources between themselves, effectively acting as end-user proxies within the JXTA-Overlay network. They forward end-user data to client peers that belong to end-users of the same group and authentication data to a broker. A client peer is assumed to belong to the same groups as its current end-user.

Brokers control access to the network, requesting end-user authentication, and help client peers interact between themselves by propagating their related information. Brokers are very important since they exchange information about all client peers, maintaining a global index of available resources, thus allowing all peers to find network services. Brokers also act as beacons which client peers which have recently gone online use to join the network. For that reason, they usually have well-known identifiers, such as a DNS name or a static IP address.

All the information related to user configuration (username, password and group membership) is stored in a special single entity within the JXTA-Overlay network: a *central database*. Only brokers may access the database data, in order to check end-user authentication attempts and organize them into groups. It is assumed that some *administrator* takes care of properly configuring the database, registering new end-users. Nevertheless, JXTA-Overlay does not impose any constraint on the database architecture.

Vulnerabilities. JXTA-Overlay's design is not concerned with security, with the only exception of end-user network access control via a username and password. As a result, it is vulnerable to different security threats which may jeopardize the network. A security study must take into consideration the fact that not only entities external to the JXTA-Overlay network may try to subvert it, but also malicious legitimate users.

Some of the greatest security concerns in JXTA-Overlay are the following ones:

– Transmitted data may be easily eavesdropped, since no data confidentiality is provided. Even though it may be argued that data confidentiality in message exchanges between end-users is just an optional feature, there are some cases where confidentiality should not be optional, namely the initial authentication username and password. Currently,

both fields are sent as plain text. Therefore, any device at broadcast range may read this information with a network protocol analyzer.

– Any legitimate user may forge advertisements with no fear of reprisal. No integrity or source authenticity is maintained. False fields, such as the source client peer identifier or any statistics information, may be added into the advertisement, which will be automatically distributed by the broker and accepted by all group members, unaware of the false data it contains.
– Client peers connect to a self-proclaimed broker, but never check if it is a legitimate one. Even in the case that client peers are connecting to the proper broker address, there are no guarantees that the broker is a legitimate one, since it may be that traffic is being redirected to a fake one via methods such as DNS spoofing [24].

As can be seen, some of the current JXTA-Overlay vulnerabilities are quite obvious ones, such as transmitting sensitive data with no real confidentiality. Therefore, it could be concluded that JXTA-Overlay did not provide a security baseline and needed a serious improvement on this regard. At UOC, a security layer was developed in order to protect the system from those attacks.

Work on JXTA-Overlay improvement. A security layer was proposed for JXTA-Overlay. The results were published in [25, 26]. An overview of security framework proposal is presented in Figure 2. All modules are located at different layers within JXTA-Overlay and JXTA's own architecture.

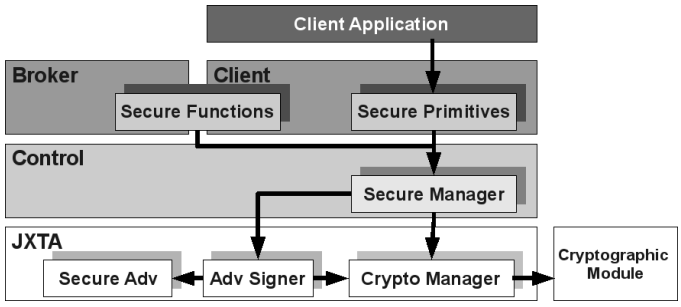


Fig. 2: Security architecture for JXTA-Overlay.

The following modules are not JXTA-Overlay specific, but an extension of JXTA protocols. As such, they may be used in any JXTA-based application:

- The *Crypto Manager* provides an abstraction layer for any cryptographic module and key management. It enables the integration of security services with any kind of module such as hardware cryptographic tokens. This approach takes into account the fact that not all such modules are accessed via plain text passwords (for example, some may use biometrics). Furthermore, even in cases were passwords are used, sometimes, each module has its methods for accessing private keys (such as the special GUI integrated into the operating system in the Windows CryptoAPI [27]). The Crypto Manager is completely modular and accepts different implementations, according to the needs of the specific cryptographic module being used for any particular deployment of JXTA-based application.
- The *Secure Adv* defines the secure advertisement format, providing a method for key transport and additional fields related to its signature.
- The *Adv Signer* manages JXTA advertisement signature and validation by interacting with the Crypto Manager. Secure advertisement management is provided in a modular way to both JXTA and JXTA-Overlay without the need to modify the standard JXTA libraries.

The JXTA-Overlay specific modules follow:

- The *Secure Manager* is the common interface to all additional security capabilities within the JXTA-Overlay Control Module, operating as a single entry point for all secure services. This module provides and initializes the Crypto Manager implementation for each instance of JXTA-Overlay.
- Finally, the *Secure Functions* and *Primitives* Modules just extend the base Broker and Client entity Modules, providing a set of additional primitives and functions which take into account security considerations. As far as secure framework management is concerned, the end-user application developer just has to choose which primitives to use, just as in standard JXTA-Overlay. The secure version of JXTA-Overlay's primitives do not replace JXTA-Overlay's original non-secure versions, but just complement them, leaving the final choice on which primitives to use to the end-user application developer.

2.2 Secure resource access control

Apart from security at the protocol level, extra efforts were made to secure resource access control and distribution in a JXTA network. The proposed technique was based on a CBID approach which made avoided reliance on a TTP (one of the shortcomings listed in Section 2.1). Even though the work focused on JXTA and was tested on the JXTA-Overlay subproject testbed, the proposal may be easily ported to other middlewares or applications.

An overview of JXTA resource publication Any kind of resource available within the JXTA network, including peers, peer groups, pipes or services, is described by an *advertisement*. Advertisements are metadata records used by JXTA protocols to describe any available resource in the network, from peers to group services. Peers cannot access a resource without previously retrieving its associated advertisement. The most important types of advertisements in JXTA are the following ones:

- *Peer Advertisement*: Describes a peer and the resources it provides. Each peer is responsible for its own Peer Advertisement publication.
- *Peer Group Advertisement*: Describes a peer group, its specific resources and its offered services' parameters.
- *Module Class Advertisement*: Provides the description of a Module Class ID. A Module Class ID is used by any service code running on JXTA to designate the modules it depends on.
- *Rendezvous Advertisement*: Describes a peer that acts as a *rendezvous peer* for a given peer group. Rendezvous are super-peers that route and index messages.
- *Pipe Advertisement*: Describes a *pipe*, the JXTA core mechanism for exchanging messages between two applications or services. Pipes provide a simple, unidirectional and asynchronous communication channel.

All advertisement types are codified using XML and passed between peers using the JXTA core protocols. Using XML codification allows programming language/platform independence, it offers a self-described specification and it is able to enforce correct syntax. As an additional feature, XML can easily be translated into other encodings, such as HTML, which may allow to access resources to peers that do not support XML.

Resource publication and discovery Advertisements are XML documents containing a unique ID and all information regarding that resource and how it may be accessed and exchanged between peers using the JXTA protocols. Peers cannot access a resource without previously retrieving its associated advertisement. Every peer maintains a local cache where all received advertisements are stored for a later use. The local cache directly makes use of the peer's file system in order to organize its content via directories and files.

Any peer may publish an advertisement to announce the availability of a particular resource by using two different methods: local and remote publication. It must be remarked that during the publication process, the original advertisement is always kept in the peers' local cache, only its index is distributed. This means that in case the peer goes offline, the advertisement will become unavailable. That makes sense, since also the resource the advertisement publicizes will be unreachable.

In *remote publication* not only indexes are distributed, but the full advertisement itself via the JXTA propagation mechanism. This method is useful in case that the advertisement must be reachable even when the publishing peer is offline. However, under the remote publication method, no assumptions can be made about which peers will really store the advertisement and for how long.

In both methods, when the expiration date is reached, the advertisement is considered stagnant and flushed from the cache, unless the same advertisement is received again, which renews its expiration date. Advertisements may be periodically retransmitted in order to attain permanency or update parameter changes.

Resource access security The current JXTA v2.0 specification guarantees message confidentiality at transport level via its own implementation of generic Transport Layer Security (TLS), providing private, mutually authenticated, reliable streaming communications. However, using this method, information security is only provided during transit by protecting the JXTA transport protocol at a lower layer. There is no method in order to specifically provide confidentiality to advertisements when stored in the local cache.

Furthermore, in the JXTA implementation of TLS, both endpoints require X.509 [28] certificates in order to provide mutual authentication and to establish a private channel. Key authenticity for such certificates is provided by a common Trusted Third Party (TTP). This information is stored locally in the peer's local cache under the protection of

a passphrase. When a peer contacts another peer group member, they can be mutually authenticated using the TLS handshake's certificate request/response and certificate validation.

Work on resource access control improvements We proposed a set of improvements to the advertisement distribution methods in order to secure them at the local cache. The proposal was published in [29, 30].

Advertisement encryption. In our proposal, it is assumed that all peers within the peer group own a private/public key pair. Advertisements are selectively encrypted using a wrapped key encryption scheme. For each advertisement field to be encrypted, a symmetric key is generated to encrypt the field. The symmetric key is then encrypted (wrapped) using each recipient's public key, obtaining a set of encrypted keys, each one of which can only be accessed by one group member. The encryption layer is maintained intact even when the advertisement is stored at the local cache, thus providing persistent security, not only during transit.

This scheme is applied to peer advertisements according to the profile shown in Figure 3. The advertisement's publisher may choose which advertisement fields will be encrypted. The encryption profile takes into account that all advertisements may contain three distinct types of fields, depending on the type of metadata stored within. Some constraints must be followed when selecting which fields are encrypted.

Mandatory fields must always exist in an advertisement, usually resource identifiers. These fields are necessary in order to locate the advertisements via the Discovery Service and for that reason they should not be encrypted.

Optional fields provide additional information about the resource, but are not mandatory. They can be freely encrypted (or only some subset of them), but then the Discovery Service will not be able to index a resource using that field. Nevertheless, an encrypted field will not disrupt Discovery Service operations on unencrypted fields (encrypted fields will be ignored).

Service fields define specific services or resources and they are used to access them. If the service field is encrypted, only the selected subset of peers which may decrypt the data will be able to properly locate and access the resource. They usually take the form of `Svc` XML elements in Peer and Peer group Advertisements, containing Pipe Advertisements or Module Class Advertisements with executable code locations.

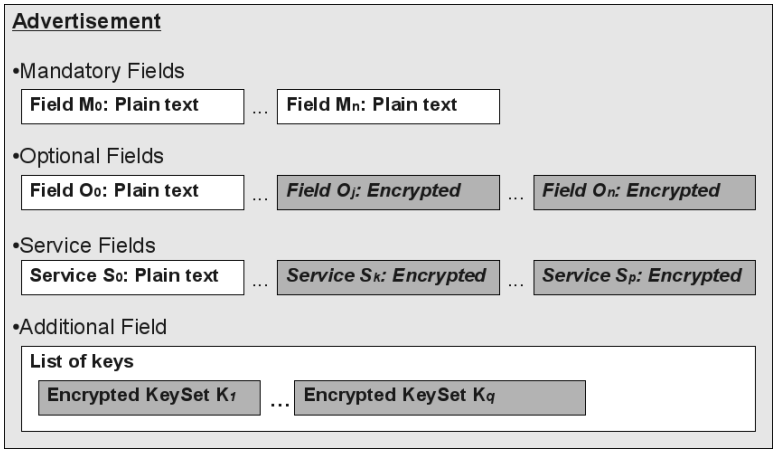


Fig. 3: Advertisement encryption schematics

Wrapped keys are included into the encrypted advertisement by introducing additional data, a list of keys. This data is encoded as a standard field named `KeyList`, following the same syntax as any other field in an advertisement. Such field contains an entry for each destination peer which should be able to access an encrypted field. Each encrypted field in the advertisement is associated to a set of a wrapped keys included in the `KeyList` field.

A wrapped key is defined by an XMLenc `EncryptedKey` element, which contains all cryptographic information necessary to decrypt such field. With this approach, XMLenc data can be added to advertisements without changing the basic elements in its schema definition, obtaining a transparent integration with JXTA and maintaining interoperability even when encrypted advertisements are used.

Encrypted fields within an advertisement are defined by XMLenc `EncryptedData` elements. Figure 4 shows the encryption profile we define in order to link each `EncryptedData` element to its corresponding `EncryptedKey`. Peers that receive the advertisement may identify which `EncryptedKey` fields may be decrypted with its own private key by searching for its Peer ID in the `KeyInfo` field of each `EncryptedKey` element. Every `EncryptedData` element may have several linked `EncryptedKey` elements (one for each peer which may access the service).

By using this XML profile and advertisement schematics, it is possible to accommodate selective field encryption as well as supporting advertisement propagation. In addition, keeping mandatory fields as plain text

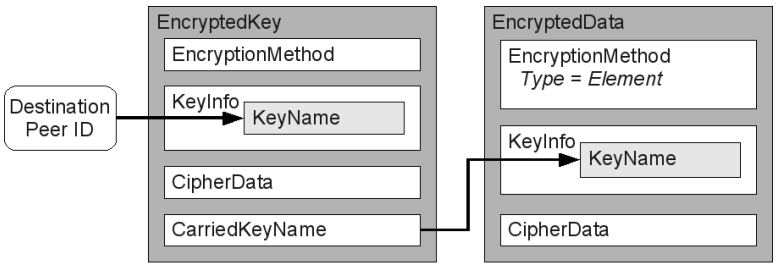


Fig. 4: XMLenc encryption profile

allows to achieve a tradeoff between data confidentiality and the capability to locate resources using the JXTA Discovery Service, since encrypted fields become invisible to the basic service operations.

With this approach, apart from standard data confidentiality, it is now possible to provide access control to group services by encrypting specific advertisement fields: those which hold service definition and configuration parameters. Only those peers which can decrypt such fields will be able to ultimately locate the services. By using selective encryption, it is possible to choose which subset of peers may access each service.

2.3 Works on node identity assignment.

The works that are described in this Section are the main goal of task WP2.T2. Both groups have addressed this same problem of assigning identities to peers in different ways. The KISON group has designed identities based on CBIDs (Crypto-based IDs), which avoids the intervention of a TTP. Meanwhile, the ISG group has designed a new protocol to deliver anonymous identifiers by means of two TTPs.

CBID-based identities. Before a key may be used for encryption (or any other cryptographic operation, for that matter), its authenticity must be guaranteed. In our proposal, public key authenticity is achieved by using CBIDs (Crypto-based IDs), which avoids the intervention of a TTP. This concept was initially conceived for IPv6 addressing in order to solve the issue of address ownership, to avoid router supplantation attacks and bind update packet spoofing [31].

Under a CBID scenario, each peer's key pair is uniquely bound to its network identifier, providing key authenticity. In the absence of a TTP,

this binding is created via applying a pseudo-random function to the public key and using the result (or part of it) as the peer identifier. There are other methods to bind key pairs to identifiers without the need of a TTP, such as an id-based approach [32] or self-certifying keys [33]. However, since JXTA IDs must follow a very specific format, they are not feasible.

A JXTA Peer ID should canonically, uniquely and unambiguously refer to a peer, being represented as 64 bytes according to the following fields: Peer Group UUID (16 bytes long), Peer UUID (16 bytes long) and identifier type (1 byte long, its value being set at 03 for Peer ID's). The rest of bytes are unused. Notice that, according to this format, only 16 bytes, the Peer UUID, are really unique to each peer within the same peer group. For that reason, those are the only values that can be manipulated in order to generate CBIDs.

We generate CBIDs from the public key by applying the SHA-1 hash algorithm on it. The result is then considered the Peer UUID in order to construct the full JXTA ID. Since SHA-1 produces 20 bytes but only 16 are needed, the first 4 most significant bytes are discarded. This process is summarized in Figure 5.

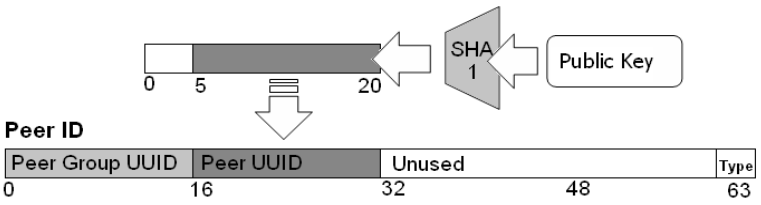


Fig. 5: CBID Generation

Using this method of CBID generation, no certificates are needed. In fact, the whole system may be based on raw private-public key pairs. In the case that certificates must be used for some reason, self-signed certificates are enough. An additional advantage of this method for generating CBIDs is that since it is based on the SHA-1 hash algorithm, it applies under the UUID type 5 specification [34]. This method also allows to integrate peers which use CBIDs with those who do not, since Peer IDs which are not CBIDs in a JXTA network are random numbers anyway.

It must be taken into account that a single ID is generated from a specific public key, so it could be a problem if two different public keys

produce the same UUID field. The probability of such occurrence can be seen as a collision problem, similar to the Birthday Paradox [35]. The probability of a collision in such scenario is $p(n;d) \approx 1 - e^{-(n(n-1))/2d}$, where n is the number of values drawn from the distribution and $[1,d]$ the distribution range ($d = 2^{128}$ for a 16 bytes long, 128 bits, UUID). However, according to [36], a typical JXTA peer group should not have more than 32 members, since above this number performance starts to degrade. With 32 members, the probability that two of them will have the same ID providing a different public key is about $1/20^{36}$, which is low enough.

In order to validate CBID ownership, whenever a peer advertisement is retrieved, the contained public key is extracted and then used to generate the source address, *i.e.* the CBID.

Anonymous identities and access control. In the previous explanation, the KISON group of the UOC has designed a system that allows to construct identities of peers using CBIDs, which are based on network identities. However, there are some systems that require the anonymity of users, and hence they cannot use such type of identities. Anonymity is a security service that is quite usual in most structured p2p networks. However, it is very important to have a mechanism to prevent new users to choose where they want to put themselves in the virtual space of the overlay. This would limit the capability of users to create plots or perform some malicious actions. Just recall that in a p2p overlay the identity of the nodes is directly related to the location of the nodes in the virtual space, and hence to the services/resources they manage. Therefore it is necessary to have a robust system to construct and assign identities to nodes, even in case that anonymity is required. In addition, a robust access control mechanism is necessary to prevent that a user joins the network with different identities, which is known as the Sybil attack [14]. This one would prevent a user can create her own set of malicious nodes and use them to lower the network performance or isolate another user, among others.

For this reason, the ISG group of the UPC has designed a new protocol to construct, assign and deliver anonymously secure nodeIDs in structured p2p networks. The protocol provides unique, stable, anonymous and non-self-managed identifiers.

Identity and access control in existing p2p overlays. Here we summarize how some of the most typical overlay networks implement access control and identity assignment.

In CAN (Content-Addressable Network) [5], nodes are directly identified by their assigned zone. This assigned zone is supposed to be chosen at random at the bootstrap phase, but in fact the newcomer can choose the area of the virtual space to be placed. Regarding the access control, nodes can enter or leave the system as many times as they want, changing the zones they manage on each arrival. Therefore, nodeIDs in CAN are self-managed, not random, and not stable. This can cause multiple security problems. For instance, a set of malicious nodes (or one node performing a Sybil attack) may censor a target document by choosing the zone that contains the key of the target document. This malicious node will have the possibility to deny the access to the target document, or even to share a modified version of it. It is possible to carry out a similar attack against nodes by controlling several zones and corrupting the routing system to encircle a target node, control its communications or isolate it. Reputation systems are known mechanism to control such misbehaviors. However, reputation systems are effective in case that identities are stable. In CAN, a node with bad reputation can leave the network and enter again managing another (or the same) zone.

In Chord [6], newcomers calculate their nodeIDs as the hash (e.g., SHA-1) of their IP addresses. The network only checks that nodeIDs are unique. In Chord, nodeIDs cannot be considered stable because there are users that use dynamic addresses. For the same reason, nodeIDs have some degree of self-management in case the node can choose its IP address (or spoof the address of another node). Despite Chord obtains the nodeID as the hash of the IP address, this provides a poor level of anonymity. Regarding access control, the authors in [6] mention that this task could be delegated to an external server, or even to an internal node, but they do not define any mechanism.

In Pastry [7], the authors state that nodeIDs must be generated randomly by a high-quality uniform random number generator or a secure hash function. As an example, the hash of the IP address of the node can be used. So then, as in Chord, this means that nodeIDs are self-managed, are not stable, and are not anonymous. Users can be anonymous if a random number is used as nodeID. However, in this case the system cannot prevent the user to change its nodeID each time she joins the network. The authors do not describe how access control is performed, and any

newcomer is just accepted as a valid peer becoming part of the Pastry network. Therefore Sybil attacks are possible.

In Tapestry [8] and Kademlia [9] newcomers generate their own identifiers, so nodeIDs are self-managed and not stable. Like previous overlays, they do not include any access control mechanism, so they suffer from the same security weaknesses. Users can perform Sybil attacks, and reputation systems have little utility.

Just to summarize, binding the identity of users to their IP addresses may compromise the stability and anonymity of nodeIDs. This is due to IP addresses are usually dynamic, and due to they are vulnerable to IP-spoofing. If nodeIDs are not stable, it is difficult to control the past behavior of nodes (reputation).

Some networks allow nodes to use self-managed identities, so that it is possible to prevent IP-spoofing attacks. However, this kind of identities is even less stable than the IP-based ones. Self-managed identities may also permit nodes to locate themselves in a certain zone of the virtual space of the overlay. In addition, if there is no access control a malicious node can generate a new identity each time it behaves badly, or it can control several identities (Sybil attack), so it is possible to censor resources within the overlay or isolate target nodes.

Work on identity and access control. The ISG group of the UPC has designed a new protocol to construct, assign and deliver anonymously secure nodeIDs in structured p2p networks. This protocol is similar to the one described in [37], but adapting it to the structured p2p overlays scenario. The protocol introduced by Friedman and Resnick provides unique and anonymous pseudonyms to users. However, nodes can still choose their own identifiers (self-managed identities) and locate themselves in a certain region of the overlay. In addition, assigned identities are so stable that they cannot be changed, even in the case of compromised keys.

The protocol designed by the ISG group of the UPC group also assures that nodes cannot self-manage their nodeIDs, and that identities can be revoked and re-issued in case of compromised keys. The mechanism is based on the use of two Trusted Third Parties (TTPs) which are responsible for the access control, and the collaboration of another node of the overlay which is who chooses a part of the nodeID. This work has been submitted for publication in [38].

This identity and access control protocol meets the following requirements:

- Uniqueness: each node should only manage one nodeID in the system. This requirement is necessary to avoid Sybil attacks.
- Stability: nodes should not have the possibility to change their nodeIDs. This requirement is necessary to implement efficient reputation systems.
- Anonymity: the rest of nodes and the two TTPs should not relate the real identity of a node to its nodeID.
- Self-management: nodes should not be able to choose their nodeIDs, avoiding them to place themselves in a certain location of the overlay.
- Uniformity: nodes should have nodeIDs uniformly distributed over the entire overlay. This is necessary to achieve the proper load-balancing among all nodes of the overlay.

This protocol allows revocation of real-world certificates and the certificates used within the network. For this purpose, two Trusted Third Parties are used. The first entity controls users by real-world certificates, and the other one issues valid certificates for the network. Both parties maintain a list of active users and share a Serial Number (SN) for each of them. This SN relates the real-world identities with the nodeIDs, which makes it possible to revoke any of the certificates while maintaining control over the other.

Through the use of real-world identities to join the network, the TTP1 can check if a node had already requested a nodeID (it protects structured p2p networks against Sybil attacks) and ensures the stability of the identifiers. However, assignation of the nodeIDs is performed jointly by the node itself and a bootstrapping node (a different one each time). Thanks to this bootstrapping node, the protocol does not allow self-managed nodeIDs and also ensures the uniformity and uniqueness of the identifiers. The bootstrapping node is chosen by both the TTP1 and the newcomer. This node and the TTP2 do not know the real-world identity of the new node. In this way we preserve anonymity of the newcomer.

3 Other identity related issues

In this Section we will introduce the work made in some other research areas that are related to identity. These works are focused on trying to provide anonymity, privacy and reputation in some scenarios, which include p2p and MANET networks.

3.1 Anonymity and privacy

The work of the KISON group of the UOC in [39–42] evaluates the risk of privacy leaks in p2p services as a consequence of the dependency of p2p communications to the current Internet Domain Name resolution System (DNS) protocol. Let us recall that DNS queries generated by p2p applications may reveal — to one or more DNS servers — the origin and the target of such queries. Moreover, the rise of new p2p services, especially those related to the integration of people and physical objects to the Internet, may expose complementary private/business sensitive data that can be eventually sold to untrusted third parties with malicious purposes (e.g., profiling, spam, scams, and untruthful marketing). Indeed, privacy leaks originated from the use of DNS queries stem from fundamental features of traditional Internet protocols and are not easy to solve by simple applying cryptography or new patches. Indeed, privacy threats targeting the DNS service have been overlooked for several years. Even the DNS security extensions (known as DNSSEC) have avoided to address this problem. The objective of our work was intended to improve the current state-of-the-art in privacy-preserving solutions for Internet-based services, respecting important properties like usability and pervasiveness. That means that the research must lead to suitable and implementable solutions, improving the current level of security offered by some of today's DNS-dependent services.

Some of the threats addressed in our work are: (1) association of source-target instances of a query; (2) identification of the origin of a query; (3) identification of the target of a query; (4) amount of queries issued by a given origin; (5) amount of queries issued for a given target; (6) differences between the data volume originated by two (or more) different sources; (7) differences between the data volume originated for two (or more) different targets. Notice that some of these threats, e.g., origin-specific threats, can be addressed by enforcing source anonymity. More specifically, origin-specific threats can be addressed by simple using general-purpose anonymization solutions, such as those based on Onion Routing. The work presented in [39] evaluates the use of the low-latency anonymity system Tor (The second generation Onion Router) to enforce DNS source-anonymity. Tor is a popular low-latency anonymity system that can be installed as an end-user application on a wide range of operating systems to redirect the traffic through a series of anonymizing proxy circuits. The construction of these circuits determines both the performance and the anonymity degree of the Tor anonymity system. While some circuit construction strategies lead to delays which are totally tol-

erated for the resolution of DNS queries, they can make the system vulnerable to de-anonymization attacks. The strategies analyzed in [39] are only those based on uniformity-selection and bandwidth-based selection. We are actually evaluating new strategies for the construction of Tor circuits, with respect to their de-anonymization risk and latency, regarding its use for anonymizing DNS queries. The results are expected to be soon published in a forthcoming extended version.

Threats targeting at both the origin and destination of the DNS queries require some more intrusive solutions (i.e., requiring modifications of the DNS protocol). In this sense, the work presented in [40, 41] discusses the advantages of combining *enforcement of source anonymity* and *random set queries*. The use of random set queries expects that, each time a client wants to query a domain name N_i, it constructs a query set $\mathcal{N}$, of size n, comprising of N_i (i.e., the real target) and $n-1$ randomly picked names (i.e., confusing targets). The client then queries every domain name in $\mathcal{N}$. If we combine this strategy with the use of infrastructure-based anonymity (i.e., use of TOR), then a malicious host eavesdropping DNS queries cannot apply neither origin discovery nor target tracking. The main advantage of this solution is that it reduces the adversary space, since now, only local name servers controlled by a malicious adversary may be seen as a serious risk.

In [42] we evaluated two state-of-the-art proposals presented by Zhao et al. in [43, 44] based on random set queries and Privacy Information Retrieval (PIR) strategies. The preservation of privacy of these two proposals is achieved by introducing random noise during the execution of DNS queries. Both works were presented in the literature without providing specific practical results. Motivated by the benefits and limitations that these two proposals, combined with the use of infrastructure-based source anonymity, may suppose, we implemented and tested them over a GNU/Linux setup. Our evaluation confirms that the main benefit of the use of random set queries is its simplicity and easy implementation; while its main drawback is the increase of latency and bandwidth during the execution and resolution of queries. The second approach, which aims at reducing these limitations, gets inspiration from Privacy Information Retrieval (PIR) techniques. We observed that although the second approach successfully reduces bandwidth consumption, it significantly modifies the DNS protocol. It therefore requires the modification of both DNS client and server implementations. Moreover, we pointed out to serious security flaws on both proposals if active attackers can target those mechanisms. We addressed these flaws on an improved version of the two proposals

which, moreover, builds upon DNSSEC protocol extensions to preserve authentication and integrity of queries. Our solution still requires, however, additional improvements to be fully effective.

3.2 Reputation in p2p overlays

Reputation systems are proved mechanisms to prevent malicious behaviors. Reputation systems encourage honest cooperation, discourage peers from changing their identities and can be used to punish malicious or selfish behaviors. Reputation is measured from direct or indirect knowledge of previous interactions between peers. Individual peers can rely on their direct experiences or can obtain reputation information from other peers.

State-of-the-art. Several distributed reputation schemes specifically designed for p2p systems have been proposed in the literature. Here we mention some of the most significative proposals by terms of how they specify the framework for data and communication management. The local reputation system is the simplest distributed reputation model. After each service transaction, the peers rate it according to their experience. These peers use this rating information to update the reputation score of the other peer. In this model, no reputation information is shared with other peers.

Reputation systems based on polling mechanisms [45–49] extend the local reputation system and provide an enhanced knowledge of the reputation of the peers. This is because, apart from the local collected experiences, peers can ask other peers about their opinions. In these systems, peers exchange directly reputation scores, which are locally computed from the accumulated experiences. An analysis of local and polling reputation systems can be found in [50].

Reputation schemes based on transaction certificates [51–53] provide a different view to the problem of distributing reputation information. In these systems, peers exchange reputation information by means of digital certificates. Usually, these certificates contain opinions about individual experiences that can be use as feedback to compute the reputation scores of other peers.

EigenTrust [54] is an algorithm for trust management where some other peers compute the reputation score of a peer. Although this algorithm is designed as a secure algorithm it presents some limitations, especially against collusion attacks.

Work on reputation in p2p overlays. Some of the benefits of fully distributed p2p systems are scalability, resource aggregation and interoperability without any administration cost or centralized infrastructure support. However, *pure* p2p architectures tend to be inefficient [55], because they consist of a set of peers with different capabilities [56] (e.g. bandwidth or CPU). On the other hand, *hybrid* networks [57] take advantage of this heterogeneity, distributing tasks among peers according to their capabilities. Super-peer systems are hybrid networks where peers form clusters around Super-peers. Peers send information and/or queries to their Super-peers and receive results from them. Hence, Super-peer networks are hybrid p2p systems formed by Super-peers and their associated client peers. These hybrid systems are normally more efficient than pure p2p systems. However, if the specific tasks of the Super-peers are not properly distributed among peers, Super-peers can also become bottlenecks in the system.

The ISG group of the UPC presented in [16, 58] a distributed reputation scheme called Surework, which is based on the Super-peer paradigm. Surework promotes that peers with high reputation scores become Super-peer for reputation, acting as *reputation servers*. In Surework, these special peers are called Sure-peers. Peers form clusters around Sure-peers and they share their reputation information. Trust management in p2p systems is usually partitioned into three subproblems [59]: (1) define global model of trust; (2) determine the local algorithm that each peer applies to determine trust; and (3) specify the framework for data and communication management. Surework was focused on the latter third subproblem, that is, how reputation information is stored and shared among peers. Surework is flexible in the way that peers compute reputations and in the semantics of reputation. Surework has been designed to be a scalable, self-managed and fault tolerant. In addition, Surework introduces incentives to promote reciprocity by encouraging peers to provide better services to most reputable peers. There are other reputation systems based on Super-peers [60, 53, 61], but they normally require an *a priory* established Super-peer architecture, meanwhile in Surework the hybrid structure is established dynamically.

Surework is based on clusters, but the way in which clusters are formed can be improved. Just next, we introduce some works of the KISON group of the UOC that addresses this problem.

3.3 Cluster formation

Part of the study of the KISON group of the UOC has been focused in p2p networks build over an ad hoc network [62]. The best way to organize ad hoc networks, overall if they are large in area and number of nodes, is dividing the nodes into multiple clusters and designating a clusterhead for each cluster which is responsible of traffic management and control the access capability of nodes to reach other peers. Moreover, clusterheads possess valuable information about the location and contacts of the network users, which is susceptible of disclosing private details of a user's profile. Thus, the election of the clusterhead is critical. From the viewpoint of building a scalable and efficient network, they shall be the nodes that can guarantee a backbone at a minimum cost. From the point of view of establishing a reliable and robust network, they should belong to honest users. Hence, in order to prevent malicious nodes take control of the group in a fraudulent way and avoid selfish attacks from suitable nodes, the clusterhead needs to be elected in a secure way.

The election of the appropriate node to be a clusterhead is not exempt of vulnerabilities. We identify two types of attacks that can generally be performed by liable clusterheads:

1. **Selfish attack.** Nodes cheat on the election process in order to not be elected clusterheads and keep energy and bandwidth for themselves.
2. **Greedy attack.** Nodes cheat on the election process in order to be elected clusterheads and have access to valuable information of the network topology and control the traffic of the members of their cluster. A malicious clusterhead could use its position to execute the following attacks.
   - *Denial of service attack.* The aim is to disrupt the cluster and isolate its members.
   - *Topology dominance attack.* A clusterhead can abuse of its topology control to address the cluster members to use some determinate network resources which can benefit it.
   - *Location disclosure attack.* A location disclosure attack can give information about the node's location or the structure of the network to unwanted users.

In order to prevent the stated attacks, some requirements must be considered.

- **Fair election of the clusterhead.** By fairness we mean that the best node (based on a specific metric) to serve as a clusterhead is

the one that has to be elected (unless it has previously occupied this position and other nodes have still not done it).
- **Rotation of the clusterhead role.** Rotation avoids, on the one hand, that a node gets burdened with more duties than others, and on the other hand that it can freely dominate the cluster.

We have proposed a protocol that follows the stated requirements and provides fairness and security to the cluster formation process. To do it, it uses a distributed algorithm that monitors the system and checks if the data claimed by a node is coherent and reliable. We assume the majority of network users are honest. In order to authenticate the information provided by each user and be able to track its behavior, a mechanism based on symmetric ciphers, hash functions and digital signatures has been developed.

The cluster formation process is composed of three subprotocols as follows:

1. The **Cost Discovery Protocol** provides reliable information about the measure used to evaluate the suitability of a node to be a clusterhead. Our protocol uses a utility function based on the degree, i.e., the number of one hop neighbors. In this phase, nodes publish the number of neighbors they have and exchange their particular perspective of the network to create a consensual view.
2. The **Clusterhead Designation Protocol** selects the best suited node in a neighborhood to be the clusterhead based on some verifiable data. The information provided by each node in the Cost Discovery Protocol is correlated with data from other nodes.
3. The **Cluster Management Protocol** manages the cluster formation and actualization once the clusterhead is set up. Cluster nodes endorse the role of the elected clusterhead by signing a claim. Then, the clusterhead is set up as a trusted authority for that cluster. The clusterhead controls the authorized nodes that can work in its neighborhood through the issuance of public key certificates that attest it.

The designed protocols use different cryptographic algorithms. Symmetric ciphers are used to construct commitments. Digital signatures are the basic tool for providing authenticity when two peers first interact. Finally, hash functions are employed to create hash chains that efficiently produce one time authenticity tokens.

The proposed protocols assure that the selected clusterhead is the node with the highest degree in a neighborhood. This role is temporal

to prevent nodes gathering too much sensitive information on the cluster members they control, and moreover, to avoid the exhaustion of these selected nodes. The process is fair in the sense that the elected node is the best one to be the clusterhead and no fakes can be made to become the clusterhead or avoid the role. Nodes that become clusterheads are securely appointed so that they can be recognized by any node of the network.

Furthermore, the proposed cluster formation process can be applied in flat networks that want to create its first topology, or in clustered networks that have to update some clusterhead roles.

3.4 Trust and reputation models in MANET

Mobile Ad Hoc NETworks (MANETs) are infrastructureless networks formed by wireless mobile devices with limited resources. Source/destination pairs that are not within transmission range of each other must use intermediate nodes as relays [63]. The cooperation among nodes is fundamental for the operation of MANETs, since nodes that contribute with their own limited resources, such as battery, memory and processing capacity, should be able to use the resources contributed by other nodes. However, in this environment, there could be free-riders or selfish nodes, i.e., users that want to maximize their own welfare by using resources from the network to send their own packets without forwarding packets on behalf of others [64]. Thus, it is important to encourage nodes to participate in essential network functions such as packet routing and forwarding, because the higher the cooperation the better the network performance. In this sense, several trust models have been proposed as mechanisms to incentive node participation within the network [65]. A trust model is a conceptual abstraction to build mechanisms for assigning, updating and using trust levels among the entities of a distributed system. In other words, the trust model allows the establishment of trust relationships among different entities with some degree of credibility, for a given action [66].

State-of-the-art. Models for cooperation enforcement in MANETs can be broadly divided in two categories according to the techniques they use to enforce cooperation [67]: credit-based models and trust models. The former category is based on economic incentives, whereas the later is based on building reputation to enforce cooperation.

In credit-based models, the network tasks are treated as services that can be valued and charged. These models incorporate a form of virtual

currency to regulate the dealings among nodes. Some of these models are the one introduced by Buttyan and Hubaux [68], Sprite [69], Express [70], OCEAN [71] or ad hoc-VCG [72].

On the other hand, models in which trust is the base for cooperation are envisioned as the most promising solutions for MANETs because these models do not have the restrictions of credit-based models. Trust models can frustrate the intentions of selfish nodes by coping with observable misbehaviors. If a node does not behave cooperatively, the affected nodes, reciprocally, may deny cooperation. Several trust models have been proposed in the literature for improving the performance of MANETs. In [65], there is classification of trust-based systems based on the theoretical mechanisms used for trust scoring and trust ranking. Following this classification, we can further divide the trust-based proposals in approaches based on social networks [73–76], information theory [77], graph theory [78] and game theory [79–86].

Work on trust models on MANET The ISG group of the UPC has designed and simulated DECADE [87, 88] (Distributed Emergent Cooperation through Adaptive Evolution in Mobile Ad Hoc Networks), a new trust models based on game theory, and more specifically in a non-cooperative game. DECADE uses genetic algorithms for the evolution of strategies, in particular, we take as reference model the centralized one presented in [89]. This model is interesting because the evolution algorithm achieves promising results regarding cooperation and energy saving. However, we are still concerned about the highly centralized nature and the slow convergence of its evolution algorithm. Optimal strategies are obtained by using a centralized entity that runs a conventional genetic algorithm in an off-line way after a large number of interactions between nodes. For these reasons, we propose a non-cooperative game theoretic trust model in which strategies evolve on-line according to a distributed genetic algorithm without requiring too much data exchange among nodes. Our model exploits the distributed nature of a MANET by using a local genetic information exchange. Our algorithm works much like plasmid migration in bacterial colonies [90, 91] combined with a parallel cellular genetic algorithm [92–95]. Through this procedure, individual nodes select the strategies that locally maximizes their payoff in terms of both packet delivery and resource saving.

The foremost characteristics of our evolution algorithm are its distributed implementation and its convergence speed. These features make our algorithm readily implementable in a MANET since we exploit the

natural structure of the network in two ways: (1) the evolution is carried out in a distributed way through local neighborhoods; and (2) The genetic information is quickly disseminated all over the network after a few time since these neighborhoods highly overlap through mobility.

Work on reputation models in MANET The ISG group of the UPC proposed a novel service-aware reputation-based routing protocol for video-streaming over MANETs named RDSR-V [96] (Reliable Dynamic Source Routing for Video). RDSR-V is a DSR-based variant that chooses the best path to forward each video-stream applying cross-layering between routing and service layers. Standard DSR always selects the shortest path to forward data. RDSR-V assigns a forwarding reputation value to paths and individual nodes that are being used or have been used to forward video-streams. Then, we use these reputation values to select the best path in order to obtain the best end-to-end Quality of Service (QoS). In our opinion, the high amount of traffic generated by typical video applications can clearly justify the use of a service-specific reputation system. On the other hand, to update the reputation scores, we monitor the video-streaming service feedback instead of using the typical *watchdog* mechanisms based on promiscuous monitoring. Promiscuous monitoring forces nodes to analyze great amounts of data not destined for them. This kind of mechanism can make the system impractical since nodes may waste too much battery in the monitoring process. On the contrary, we propose a novel approach which uses the video-streaming standard signaling packets (RR-RTCP packets). This way of monitoring the video-stream forwarding reputation allows a source node to monitor end-to-end QoS parameters without introducing new overhead. It is important to point out that, differently to the most reputation systems proposed in the literature, like CORE [75] or OCEAN [97], we are not directly monitoring the behavior of individual nodes, but we are monitoring the overall behavior of the transmission path from the source to the destination. This allows us to enhance the end-to-end QoS provided to the final user. Similarly to other reputation systems, RDSR-V introduces cooperation incentives but in our case, these incentives have been especially devised with the final objective of improving the end-to-end QoS of the video-streaming service. Also, performance evaluation results show that RDSR-V clearly outperforms standard DSR and OCEAN for the video-streaming service in presence of non-cooperative nodes.

A basic description of RDSR-V is shown in Figure 6. Essentially, RDSR-V works as follows:

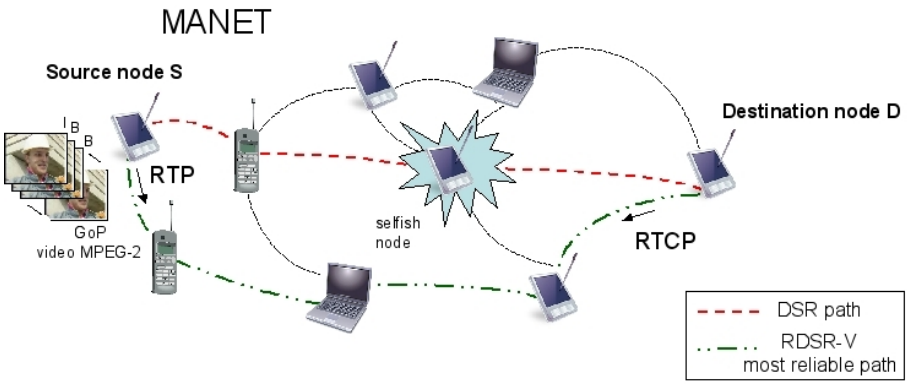


Fig. 6: RDSR-V tackles the damage caused by selfish nodes.

- The source selects the path to forward the video-stream taking into account not only the number of hops but also the reputation of paths.
- During the video transmission, the source uses service feedback information to switch to a different forwarding path if necessary. This could depend on degradation of the current path, on possible node unavailability, and on any other circumstance that could disturb the proper forwarding of the video-stream.
- The source updates the reputation scores of the corresponding path and of the nodes that belong to that path using the service feedback information (RR-RTCP packets).
- The mechanism performs *punishment actions* over selfish nodes as well as *rewarding actions* over collaborative nodes, seeking to enforce network cooperation. In RDSR-V, nodes assign different priorities in its internal queues when forwarding the packets for collaborative, medium-reputation or selfish nodes.
- Silent nodes are treated as medium-reputation nodes. For this reason, they do not obtain the best possible QoS for an intermediate node. This action is an incentive to cooperate.
- Finally, if the reputation of a node decreases until a certain threshold, the source treats it as non-collaborative, and RDSR-V reacts preventing that node from being part of forwarding paths.

As a local (first-hand) reputation system, RDSR-V inherits several characteristics of OCEAN [97]. However, since our proposal is specific for the video service and uses service feedback instead of observation of

the neighbors' behavior, there are important differences between OCEAN and RDSR-V.

4 Proof of concept

Information is one of the most valuable and private possessions of many people. For example, information about our investments, the photographs of our last holidays and the documents in confidence that we manage in our office. Often, personal data are stored in hard disks or removable devices that are both fragile and error prune. Human or computer errors, fire or water damage accidents or stealing of hardware are real threats that will cause loss of personal data. Data are even a more important resource for enterprises, that are willing to pay for expensive and complex mechanisms to ensure their availability. The main objective of a distributed file system in a p2p network is spreading data in many different, separated nodes in order to protect it against local malfunctioning hardware. If data are distributed among a bunch of nodes, the real damage that fire or hardware stealing causes is drastically reduced.

But distribution of data in p2p networks has a main drawback from the point of view of security. Many people will not find acceptable that their data are stored in a computer under control of a complete stranger, since she will be able to access to her sensitive bank accounting information. Not only the private contents of a document something important to protect, even being able collecting the names of those files that a user store in her personal namespace is a violation of her privacy since an attacker can learn important information from them. A system that focuses on providing reliable and private storage of personal data must take into account security as one of the main steps of the system design.

To tackle with this problem, the ISG group of the UPC proposed a secure architecture for the storage of personal data in distributed peer-to-peer networks. The main point of the proposal is to provide a reliable service that clients will use to backup its valuable and sensitive personal data, without fear of losing their data or that attackers are able to learn something from them.

4.1 Cooperative File System

Our proposal aims to add security to the **Cooperative File System** (CFS [98]), that is a distributed file system over a **Distributed Hash Table** (DHT). Figure 7 shows the architecture of CFS.

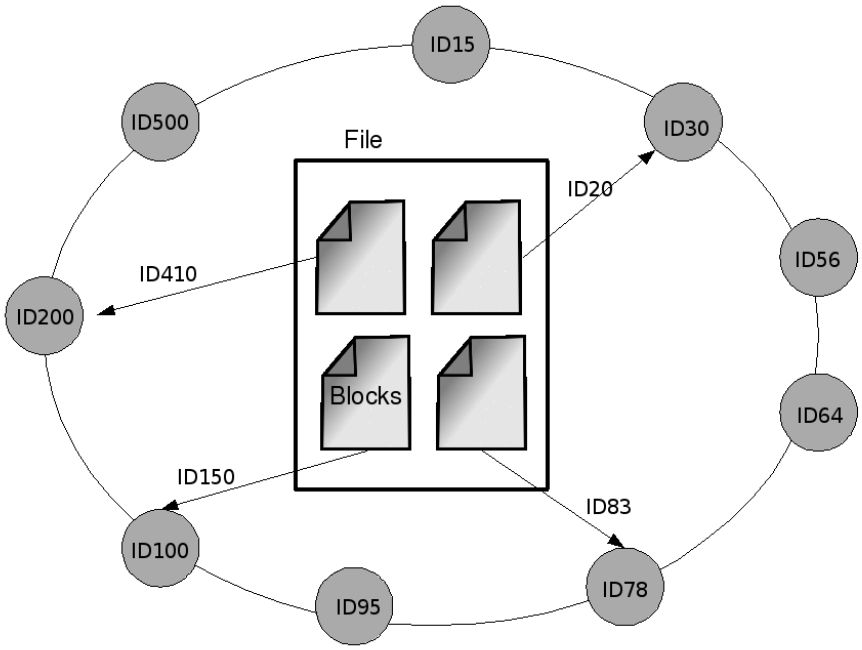


Fig. 7: Cooperative File System.

A DHT is a structured peer-to-peer network where nodes choose a random identifier ID_i. Each piece of data is store under a key that works in the same way that a node identifier. Each node in the network finds and creates a link at least to the node that has the very next $ID_n > ID_i$ in increasing order. Then, the node ID_i is in charge of storing a piece of data that is identified with a key $ID_i \leq ID < ID_n$. The process goes on until that the last node links to the first one and the whole network creates a structure with the shape of a ring. In order to PUT or GET the piece of data that was identified with ID, a node sends clockwise a message in the ring to the node in charge of ID. To improve the routing of messages inside the ring and minimize the number of hops that a message takes to reach its destinations, nodes of the network also link to many other nodes in the ring, according to a certain, preknown algorithm. Current implementation of DHTs include Kademlia [9] or Chord [6], and the main difference between them is the specific algorithm to choose far links.

CFS was implemented over Chord. It gets a file F with a filename f, divides the file in blocks, store the blocks B_i and calculates H_i and ID_f.

$$F = \{B_1 \cup B_2 \cup B_3 ... \cup B_n\} \quad (1)$$

$$H_i = hash(B_i) \quad (2)$$

$$F_h = \{H_1, H_2, H_3, ..., H_i\} \quad (3)$$

$$ID_d = hash(f) \quad (4)$$

Then CFS saves each block B_i in the DHT identified as H_i and stores the list of H_i in a special block saved under ID_f. Files in CFS are persistent and the system warrants that every file will be retrieved, regardless of its unpopularity. This is a main advantage to store personal data and backups over other distributed filesystems that were proposed in the literature. Unfortunately, CFS authors did not consider security in the design of the system.

Security Requirements We considered a scenario where individuals use a P2P network as a backup of their personal files, and as a nearly unlimited disk space for little devices. Their main objective is not an efficient distribution of documents to hundreds of people, as many P2P networks do, but preserving their own and unique documents nearly for ever and the ability of accessing and modifying them from any device that they may own. Furthermore, they are really interested in keeping their documents private and out of the reach of both casual and commercial attackers. Since CFS does not include any mechanism to ensure the privacy of the stored data, we proposed the creation of a Secure Cooperative File System (SCFS).

In our work, the security requirements of SCFS were:

Confidentiality Data should not be readable from other participants of the p2p network that the user than uploaded it. In a distributed filesystem any node stores personal data of users, and the filesystem must supply mechanisms to warrant that the owner of a node is not able to read other users' personal information. In addition, the property of confidentiality presents a real advantage for every owner of a node in the P2P network. If owners cannot access to the content that they store in their nodes, they can deny any knowledge about the data and protect themselves from legal prosecution for storing illicit content.

Privacy Malicious users may collect personal information about habits or interests of users. Even if attackers are not able to access to real

documents, the file name or its metadata may be relevant to learn private information. Commercial research and censorship prune governments may get profit from capturing the names of files accessed by users.

Integrity Since documents are stored in nodes that are out of the control of the data owner, it is not possible to prevent data modifications. In this sense, the filesystem must provide mechanisms to detect modification and restoring of original data, if possible.

Persistence The filesystem must provide mechanisms to prevent lose of data. Files can be lost by means of malicious nodes that remove pieces of data, users that write data in the same place, both intentionally and unintentionally, or network or node fails. Storage systems focus on persistence instead than on publishing of data.

Availability Users are constantly moving and they own many devices with different network capabilities, memory and processing power. In spite of this, they want to access to a consistent disk space from every device regardless of its capabilities. The system should provide mechanisms to allow data to be available from any of the devices of the users, regardless how and where they are connected.

4.2 Prototype description

The Secure Cooperative File System (SCFS, [99, 100]), was implemented in a prototype accessible in [101] under the GPL license. The prototype was coded in Python on a Linux system, it uses a simplified Kademlia as the algorithm for the DHT, blocks have a 2048 bytes and there is a 30% of redundancy in the IDA algorithm. Identifiers of files, nodes and users have 128 bits. AES is used as the encryption algorithm, and the first 128 bits of SHA as a hashing algorithm. The reader is highly advised to download the prototype and perform a run test before proceeding any further.

The prototype runs on any Linux system that includes a Python interpreter. It doesn't need any special library apart from cherrypy, that will be included along with the prototype.

Please, refer to the README of the prototype for updated information.

Command line interface We provide a command line interface to manage the distributed filesystem and run the network nodes. This is the *dfsh.py* script, and it includes well known shell commands such as "rm"

and “cp” in addition to node management commands such as “server” or “configure”. The next text shows the parameters of this script.

```
Usage: dfsh.py [options] COMMAND command-opts

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -c FILE, --conf=FILE  Read configuration from FILE
  -p PWD, --pass=PWD    PWD for the configuration file
  -d LEVEL, --debug-level=LEVEL
                        Debug LEVEL for information
  -q, --quiet           Do not log messages
  -v, --verbose         Log messages
  -l FILE, --logfile=FILE
                        Log messages in FILE
  -S                    Development mode
```

First, the user will need to set up the environment for her node. She will run at least once the *configure* command, that asks for a username, a password and some information about the namespace. Then, the user runs a new local node.

```
./dfsh.py configure
./fssh.py server
```

Currently, we do not provide with any bootstrap process to join the network. If you want to set up a node or connect to an existing p2p network with several nodes on different computers, you may need to manually configure your configuration file. See section 4.2 for further details.

Next, the user is ready to save any local file into the distributed file system. In this example, we will use the root of the filesystem that the user associated to himself during the configuration.

```
./dfsh.py  cp digital2.jpg dfs:///digital.jpg
```

Current implementation has not efficiency as one of its goals. Sending a large file to the filesystem may be a very long process. In order to run the test under a reasonable time, the reader is advised to store in the filesystem only small files, under 100KB in size. In order to get this file, run:

```
./dfsh.py cp dfs:///digital.jpg digital2.jpg
```

The user is able to delete distributed files and create directories in a similar way.

Configuration files Next we describe the configuration files that SCFS uses. These configuration files are associated to a user, namespace or directory, as we will see in section 4.3.

```
[Main]
nick = juanvi
uid = juanvi
root = dfs://juanvi@juanvi/root

[DHT]
datadir = dhtdata
maxbuffer = 4096
descpermetapart = 12
block = 1024

[Ring]
id = 28790746847835588597651173935682614952 5
ip = 341.341.1.1
port = 12345
known = 341.341.1.2:12345, 341.341.1.3:12345
```

The configuration files are organized in different sections:

Main includes the description of the user and the URL of the directory that this configuration file manages. You will find additional information about the various namespaces in section 4.3.

DHT configures some DHT parameters, including the local directory where data will be saved. Currently, we include several implementations of DHT as a playground to check several parameters of the network. This section includes the size of the blocks of the filesystem or the size of the buffer. This parameters will be described in section 4.3.

Ring includes the description of the local node and a reference to remote nodes. For example, the identification of the local node or the local port should be included in this section.



Fig. 8: Login form

This configuration file should be carefully managed by the users. They must be kept privately, and they should be provided to the different interfaces to access the system. The user can manually copy her configuration files to a new device to access to her private files, but she should change the configuration of the **Ring** section.

These configuration files can be encrypted.

The web interface The prototype includes a web interface that the client can use to access to her private files. You can access to this interface running *webserver/webserver.py* and pointing your browser to "http://127.0.0.1:9090". A page like the one in Figure 8 will be presented.

Here, the user can enter her pair username/password or enter a previously created configuration file. A new user can register a new profile and join the distributed system, as well. Once logged in the system, the user is allowed to perform common actions for a filesystem: creation of a new directory, uploading of a file, downloading from the filesystem and deleting a file. From the page shown in figure 9, the user is able to access to her global configuration file and to the local configuration file of her directories.

The web interface is not intended to be run on a remote server, but being a local service inside the computer that the user controls. The reader should take into account that configuration files never should leave the control of the user.



Fig. 9: View of the user's files

The web is only an interface to the filesystem, but you will need at least a node in the network to store the data. We refer to section 4.2 for instructions to run a network of several nodes.

4.3 Secure Cooperative File System

This section details the mechanisms of the Secure Cooperative File System (SCFS, [99, 100]) to grant the requisites of section 4.1. A graphical outline of the whole process is shown in Figure 10.

Assumptions and Definitions Table 1 includes the definition of the main concepts of SCFS. The reader can find a relation of symbols in table 2.

For example, Bob and Alice join to a SCFS network. The both desire to store a file that has "revenues.xls" as filename. Bob has $U_{bob} = bob$ as his identifier, and Alice $U_{alice} = alice$. Bob has $cfs : //bob/root$ as the root directory, and Alice has $cfs : //alice/root/$. In order to access to the personal root directory of Bob, she will need her global key K_d. Since their user identifiers and K_d are different, their namespaces will be different as well, as the next section will show.

Next, we detail each one of the steps of figure 10 to protect the file.

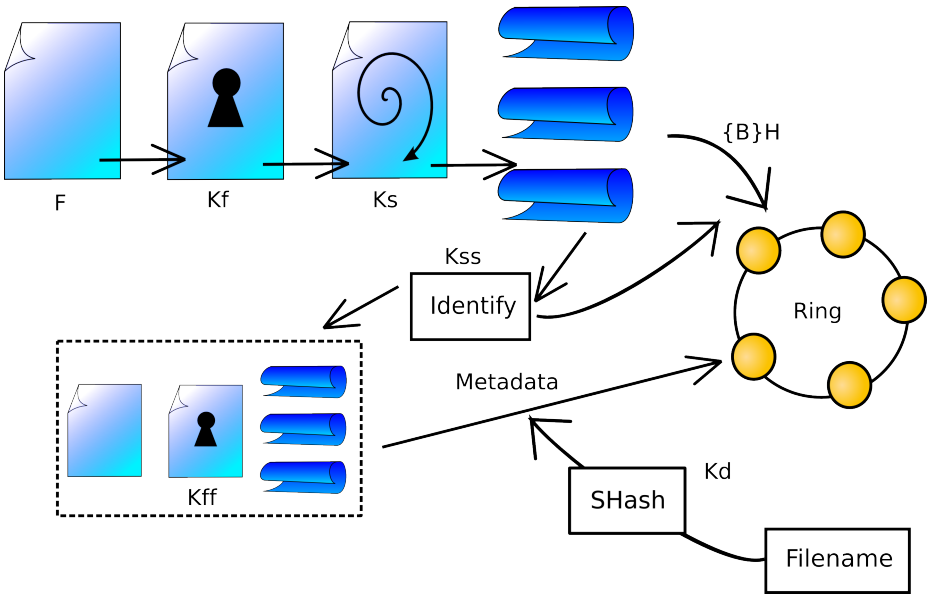


Fig. 10: Description of the process.

Securing the whole file The first step to secure a file is encrypting its contents. Each file is encrypt with a symmetric algorithm with K_f In our prototype we choose AES as the encryption mechanism.

After the encryption process, a shuffling and redundancy creator algorithm takes place. SCFS uses the algorithm IDA described in [102]. This step has a double objective. The first one is preventing that consecutive bytes in the original file were consecutive in the stored blocks. This shuffling prevents some kind of statistical attacks against files with a known structure or common header, such as PDF files. Since users can choose the kernel space of vectors for the IDA algorithm, the distribution of data inside the final blocks can be determined only by the original data owner. Key K_{ss} is used to generate the vector space that the IDA algorithms need. The second objective of the algorithm is creating data redundancy. In order to enhance the availability of the service, the algorithm takes the original file and created m blocks in such a way that it will need $k < m$ blocks to restore the original file. The dimension of B_i is chosen in such a way that it matches the size of a block in the DHT. In this sense, the system is protected against malicious nodes that randomly changes nodes, overloaded nodes than cannot manages all their parallel connections and local network failures.

User	A client of the system. Each user has a unique identifier U. Collision of identifiers of users are not managed in this work
File	An ordered array of data
Directory	An unordered set of files and directories
Filename	A human readable identifier for a file or directory. It is not unique in the system, since two different users can store different files under the same filename
Root Directory	The filename of the directory that holds every file and document of the user. In SCFS it is not mandatory for documents to be included to a directory and that users can use arbitrary, binary identifiers for their documents. In order to provide a user friendly environment, we assume that users have a root directory for all their documents and files.
Block	Each file is divided in blocks that are created with the IDA algorithm. Only k blocks of N are needed to recreate the original file
Metadata Blocks	Special blocks of data that have enough information to restore the whole file
iNode	The first block of metadata that holds a list to the rest of metadata blocks
Config file	It is a file that stores the identification of keys used in each file and the personal information of the user, as her root directory.
Nodes	Each one of the devices that a client uses to join to the network. Each node has a different identifier.

Table 1: Definitions

$$S = \{F\}_{K_f} \tag{5}$$

$$B = \{S\}_{IDA} = \{B_1, B_2, ..., B_i\} \tag{6}$$

$$dim(F) = dim(S) < dim(\bigcup B) \tag{7}$$

Clearly, this algorithm consumes both time, bandwidth and disk space in remote nodes since the dimension of the final data in B are greater than the dimension of the original file F. Users can choose the amount of redundancy to apply to each file, or even if this algorithm runs or not at all over files. Authors expect that clients will use most of the time the default redundancy of 30%.

The encryption and shuffling steps take place locally to the user and the set B is not yet published in the ring.

U	The identifier of each user. U is the same for every node that a user controls.
F	The original file of the user
ID_f	The final identifier of a file
S	The encrypted file of the user
B_i	Each one of the blocks of a file
B	The set of blocks of the file
H	The set of identifiers of blocks
K_d	The key used to secure the filename
K_f	The key used to encrypt the file
K_{ff}	The key used to encrypt the metadata
K_s	The key used to generate a vector space for the IDA algorithm
K_{ss}	The key used generate pseudo random identifiers
K	It is the set of keys used to secure each file. This set contains five conceptual keys associated to a single file. The complete set K can be generated from a master key or stored individually for each file.

Table 2: Symbols used

Securing file blocks After the previous step each block B_i has the same size than the DHT blocks. Since the data in each B_i were encrypted and shuffled, it makes little sense to any casual attacker than sniffs the blocks.

The CFS stores and retrieves blocks of data associated to a identifier. We use the **secure hash of the block contents** to identify a block.

Using hashes as block identifiers has an important drawback. A file of 20MB stored in blocks of 2048B need about 160KB to store the block identifiers. Since SCFS uses special blocks to store file identifiers, this approach needs at least 80 blocks just to store the file structure. Furthermore, since hashes will be protected with a key K_sss, the client must save K_{ss} somewhere. Anyway, hashes have the advantage that they could be used to ensure integrity of data or event prevent unauthorized overwriting, as will be stated in next sections.

The set of block identifiers of a file is H.

Metadata In this moment the user's node has a local buffer with the blocks of the file B, a set H to identify each block and a configuration file, as stayed in section 4.3. Many of these data are stored in a special block that we call iNode, in the same sense that iNodes in the traditional filesystems. iNodes store configuration, identification of the set K and the whole contents of the table of block identifiers H. Metadata are then

managed in the same way that a file object. It is padded, encrypted with K_{ff} and divided in blocks using IDA.

Metadata keep lots of sensitive information. SCFS accepts two different mechanism to protect these special blocks:

- Metadata blocks are not published in SCFS but stored locally in the nodes controlled by the client. In this way, since attackers have no access to the list of blocks of a file, they cannot even gather them to begin with any cryptanalysis. Unfortunately, keeping this data locally means that clients must use always the same node or devise an extern mechanism to distribute these file to other nodes.
- Metadata blocks are published in SCFS as regular blocks. Since they have the same length and entropy as any other block, they cannot be distinguish from regular blocks. Identifiers are assigned in the same way as block identifiers and each block keeps a reference to the next block of the chain. This prevents shuffling metadata as in regular file objects. The first metadata are identified in a special way, described below.

In the case of publishing metadata blocks, the first block is referred with a special identifier. Metadata blocks are then randomly introduced in the local buffer. In this way an eavesdropper cannot discriminate between file blocks and metadata blocks. Our prototype implements both methods.

Securing File Identifications. Every file in SCFS is related to a URI that identifies only one file. Furthermore, since nodes in the network saves part of files, it must be not possible to get the URI out from any number of parts of the file.

Since users store in SDFS their private data with an arbitrary name, the file system must provide with a separate namespace of files for each one of the users. In the other hand, users may want to share their file to any other user of the network, so there must be a common namespace in the whole network.

Our filesystem supports a human readable identification of files. In this sense, a human readable and writable string must be the main interface of the user to the system. The complexity and security involved in mapping that string to a DHT identifier must be hidden to the user by the system.

If an attacker is able to identify the first block of metadata and somehow access, to its contents, she may be able to retrieve the complete metadata and then the complete file. In this sense, securing the identification of the first metadata block is crucial for the security of the whole

system. The next section will study several mechanisms to identify this first block of metadata from the human readable filename.

Public and private identifiers. There are two possible namespaces in SCFS: user's namespace and public namespace. In both namespaces, users must identify their files with a unique filename. Using directory paths as part of the identifier is an easy mechanism to warrant this unicity in the private namespace, but it is a really hard problem in the public namespace that is out of the scope of this research.

It is up to the users to ensure that filenames are unique in the public namespace. The filename is then hashed with SHA and cut down to 132 bits in order to get the file identifier. Every user that knows the filename can get the metadata and then every block of the file. As stated in section 4.3, this does not mean that the attacker has access to the plain data of the file, but she is on the right way.

$$ID_f = hash(filename) \tag{8}$$

Each user has a private key K_d. The filename is hashed as in the public namespace and then encrypted using AES. The hashing ensures a 132 bits input to the encryption, and then maximizing the entropy of the encrypted string. The encryption step warrants that the userspace is unique and secured, since none can identify the file even knowing the author and the filename.

$$ID_f = hash_{K_d}(filename) \tag{9}$$

Publishing the file At this moment, the node has local buffer with a set of blocks B that are identified with H and the iNode of the system identified with ID_f. The node publishes the contents of the local buffer in the DHT under its keys. The reader should notice that since blocks, metadata and the iNode were randomly introduced in the local buffer, an attacker is not able to distinguish between them and then a cryptanalysis is really more difficult.

Reading process The reading process is the inversion of the writing process. From a filename users create a ID_f, maybe using a user key K_d to maintain confidentiality. From ID_f, the user gets the iNode of the file and then the list of metadata block. From the metadata, the user is able

to recreate H. Then, she downloads the blocks, reorders and the decrypt the file.

Typical DHTs warrant than a block can be retrieved for a week. This time may be enlarged with a ping from the user. In this sense, data may be indefinitely stored on the network and the DHT structure warrants that it is accessible nearly forever. Moreover, the IDA algorithm warrants that the file can be restored even if only k blocks are accessible in the network. Furthermore, files can be retrieved from everywhere in the network, so users do not have to contact to the same neighbors each time that they access to the network with different nodes.

It is easy for an attacker to identify all or some of the blocks of a user. He only has to eavesdrop identifiers from the petitions of the user. In this sense, it is easy for an attacker to save junk under the same identifier than valid users. Even fair users may store her own data under the same identifier, and their data will collide with previously stored information.

DHTs let storing several blocks of data under the same identifier. When a user asks for a key, she gets every block that was published with the same identifier. If the iNode includes the hash of the block, as stated as one of the possible options of section 4.3, it is easy to discard random and malicious collisions. Sending the hash of the block to the final node and letting to compute the hash to itself saves space, but augments the bandwidth of the system. The redundancy introduced by the IDA algorithm can help in this phase.

Additional mechanisms Apart from the mechanisms of the previous sections, our implementation of SCFS includes additional mechanisms to enhance the security of the system. In this section we will study these additional mechanisms that takes place in phases other that reading and writing data.

Attackers may join a large number malicious nodes in order to perform Sybil attacks. In this way, the attacker gains a large influence in a region of the Kademlia ring and she may be able to put the whole system down or perform denial of service attacks to some users. In order to prevent this kind of attack, the original creators of CFS proposed that the identifiers of the nodes must depend on the network address of the node. Furthermore, the random identifiers of each block disperse blocks in the whole ring, while the information dispersal algorithm warrants that the file can be retrieved even if a segment of the ring is not available.

Attackers may collect user information by means of sniffing the communications of users in the network. Files are encrypted, shuffled and split

down in blocks as explained in section 4.3, but metadata have a slightly weaker encoding process, especially the first of them. An attacker that gets the first iNode needs to decrypt a single block of data to get the list of every blocks in the system. That first iNode is encrypted using AES, but it contents known information that may simplify the cryptanalysis. An attacker may identify this first iNode because it will be the first block than a user demands. In this way, SCFS asks for several random blocks apart from the iNode.

There is no deletion service in SCFS. A deletion service need to authenticate the owner of the file in order to prevent deletion from unauthorized users. In order to enhance privacy of the users, SCFS does not include any authentication mechanisms. In order to prevent size growing with time, blocks are really deleted in nodes if user do not access to the block after a month. In order to show interest for a block without the need of downloading it and increasing the bandwidth, users must send a "ping" to the data block. Nodes count these pings as an access and will mark the block as no removable for an extra month.

Apart from deletion, both malicious and fair users may overwrite blocks of legitimate users. Since identifiers of blocks are random numbers, collisions are possible. Kademlia lets different blocks under the same key, and when a user demands that key she will receive the whole collection. SCFS takes advantage of this by means of saving the hash of the block in the iNodes. In this sense, when a user demands a key and receives several blocks, she is able to discriminate which one is the valid to create the original file. This mechanisms does not prevent to malicious users to write blocks under the same key, but their blocks won't be used to recreate the original file.

5 Conclusions

This report summarizes the advances carried out within the ARES project to improve the security of p2p networks regarding identity issues.

Prior to deal with identity problems, both research groups made some research trying to improve the security of a generic p2p overlay, in particular the JXTA platform. In this sense, a new security layer has been designed to allow peers which do not support signature to process messaging in a transparent way. Apart from this, extra efforts have been made to secure resource access control and distribution in the JXTA network.

After the security of JXTA was improved, the problem of assigning secure identities to peers has been addressed. A robust scheme to generate node identifiers is a prerequisite for the correct development of p2p structured networks, also for the interoperability of the rest of security services, for instance reputation systems. Both ARES groups have proposed identification system that apply to different scenarios. The first identity system is based on CBIDs (Crypto-based IDs), which avoids the intervention of a TTP, and the second system provides a new protocol to deliver anonymous identifiers by means of two TTPs (using blind signatures). This scheme protects the network against Sybil attacks, and generates nodeIDs with values which accomplish the requirements of uniqueness, stability, and are compatible with the anonymity of their users.

Other extra works related in some way to identity issues were made by both research groups. For instance, we have evaluated the risk of privacy leaks in p2p services as a consequence of the dependency of p2p communications to the current Internet Domain Name resolution System (DNS) protocol. We have also designed and simulated reputation and trust models for the p2p and MANET scenarios.

Finally, a prototype of the Secure Cooperative File System (SCFS) has been presented, a secure architecture for the storage of personal data in distributed peer-to-peer networks.

References

1. The Napster Home Page. http://www.napster.com.
2. Edonkey Home Page. http://www.edonkey2000.com/.
3. Gnutella. http://rfc-gnutella.sourceforge.net/.
4. C.G. Plaxton, R.Rajaraman, and A.W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Theory of Computing Systems*, 1999.
5. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, 2001.

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 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.
7. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*, volume 2218 of *LNCS*, 2001.
8. B.Y. Zhao, J. Kubiatowicz, and A.D. Joseph. Tapestry: An infraestructure for fault-tolerant wide-area location and routing. Technical report, U.C.Berkeley Computer Science Division, 2001.
9. P. Maymounkov and David Mazières. Kademlia: a peer-to-peer information system based on the xor metrid. In *IPTPS*, pages 53–65, 2002.
10. Project Kenai. Project JXTA, 2001. http://jxse.kenai.com.
11. D.S. Wallach. A survey of peer-to-peer security issues. In *Software Security. Theories and Systems*, volume 2609 of *LNCS*. Springer-Verlag, 2003.
12. E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *First International Workshop on Peer-to-Peer Systems*, 2002.
13. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach. Security for structured peer-to-peer overlay networks. In *Symposium on Operating Systems Design and Implementation*, 2002.
14. J.R. Douceur. The sybil attack. In *First International Workshop on Peer-to-Peer Systems*, 2002.
15. S. Marti and H. Garcia-Molina. Identity crisis: Anonymity vs. reputation in p2p systems. In *Third International Conference on Peer-to-Peer Computing*, 2003.
16. J.L. Muñoz M. Rodriguez-Perez, O. Esparza. Surework: a super-peer reputation framework for p2p networks. In *23rd Annual ACM Symposium of Applied Computing (SAC)*, 2008.
17. Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132, New York, NY, USA, 2005. ACM.
18. Roger Dingledine, Nick Mathewson, and Paul Syverson. Reputation in p2p anonymity systems. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.
19. H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta. Limiting sybil attacks in structured p2p networks. In *26th IEEE International conference on Computer communications (INFOCOM)*, 2007.
20. Chuiwei Lu. Detection and defense of identity attacks in p2p network. In *4th International Symposium on Advances in Computation and Intelligence (ISICA)*, pages 500–507, Berlin, 2009.
21. SUN Microsystems. Jxta v2.0 protocols specification, 2007. https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.
22. J. Arnedo-Moreno and J. Herrera-Joancomartí. XML-based security for JXTA core protocols. *International Journal on Autonomic Computing, Special issue on Efficient Techniques for autonomic resources scheduling and management in Grid and P2P systems*, 1(4):447–469, 2010.
23. T. Daradoumis L. Barolli F. Xhafa, R. Fernandez and S. Caballe. Improvement of JXTA protocols for supporting reliable distributed applications in P2P systems. In *Proceedings of the International Conference on Network-Based Information Systems (NBiS)*, pages 345–354, 2007.
24. D. Sax. DNS spoofing (malicious cache poisoning).

25. L. Barolli F. Xhafa J. Arnedo-Moreno, K. Matsuo. Securing a java p2p framework: The JXTA-Overlay case. In *Proceeedings of the 11th International Conference on Integration and Web-based Applications & Services (iiWAS'09)*, pages 158–165, 2009.
26. L. Barolli F. Xhafa J. Arnedo-Moreno, K. Matsuo. A security-aware approach to JXTA-Overlay primitives. In *Proceeedings of the 3rd International Workshop on Advanced Distributed and Parallel Network Applications (ADPNA'09)*, pages 431–436, 2009.
27. Microsoft. MSDN library. cryptography, 2007.
28. CCITT. The directory authentication framework, 1988. Recommendation.
29. J. Arnedo-Moreno and J. Herrera-Joancomartí. JXTA resource access control by means of advertisement encryption. *Future Generation Computer Systems*, 26(1):21–28, 2010.
30. L. Barolli F. Xhafa J. Arnedo-Moreno, K. Matsuo. Persistent interoperable security for JXTA. In *Proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC)*, pages 354–359, 2008.
31. T. Aura. Rfc 3972: Cryptographically generated adresses (cga), 2005. http://www.ietf.org/rfc/rfc3972.txt.
32. A. Shamir. Identity-based cryptosystems and signature schemes. pages 47–53, 1984.
33. M. Girault. Self-certified public keys. pages 490–497, 1991.
34. R. Salz P. Leach, M. Mealling. Rfc 1422: A universally unique identifier (uuid). urn namespace, 2005.
35. B. Schneier. *Applied Cryptography*. John Wiley and Sons, 1996.
36. E. Halepovic R. Deters. The JXTA performance model and evaluation. *Future Generation Computer Systems*, 21(3):377–390, 2005.
37. E.J. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10, 2001.
38. J. Caubet, O. Esparza, J.L. Mu noz, J. Mata-Díaz, and J. Alins. Secure access control and identity management in peer-to-peer overlays. In *Submitted to 9th International Conference on Applied Cryptography and Network Security (ACNS'11)*, LNCS. Springer-Verlag, 2011.
39. M. Barbeau J. Garcia-Alfaro and E. Kranakis. Evaluation of anonymized ons queries. In *Workshop on Security of Autonomous and Spontaneous Networks (SETOP 2008)*, pages 47–60, 2008.
40. S. Castillo-Perez and J. Garcia-Alfaro. Resolucion de consultas anonimas sobre dns. In *X Reunión Espanola sobre Criptología y Seguridad de la Información (RECSI'08)*, 2008.
41. S. Castillo-Perez and J. Garcia-Alfaro. Anonymous resolution of dns queries. In *International Workshop on Information Security (IS'08), International OTM Conference*, volume 5332 of *LNCS*.
42. S. Castillo-Perez and J. Garcia-Alfaro. Evaluation of two privacy-preserving protocols for the dns. In *In* 6th International Conference on Information Technology: New Generations (ITNG 2009), *pp. 411-416.*, 2009.
43. Hori Y. Zhao, F. and K. Sakurai. Analysis of privacy disclosure in dns query. pages 952–957, 2007.
44. Hori Y. Zhao, F. and K. Sakurai. Two-servers pir based dns query scheme with privacy-preserving. pages 299–302, 2007.
45. E. Damiani, D.C. di Vimercanti, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choossing reliable resources in peer-to-peer networks. In *ACM Conference on Computer and Communications Security*, 2002.

46. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents's reputations in p2p systems. In *IEEE Transactions on Knowledge and Data Engineering*, volume 15, 2003.
47. S. Chhabra, E. Damiani, and S. De Capitani di Vimercati. A protocol for reputation management in super-peer networks. In *Fifteenth International Workshop on Database and Expert Systems Applications*, 2004.
48. L. Srour, A. Kayssi, and A. Chehab. Reputation-based algorithm for managing trust in file sharing networks. In *Securecomm and Workshops, 2006*, 2006.
49. S. Song, K. Hwang, R. Zhou, and Y. Kwok. Trusted p2p transactions with fuzzy reputation aggregation. *IEEE Internet Computing*, 9(6), 2005.
50. M. Rodriguez-Perez, O. Esparza, and J.L Munoz. Analysis of peer-to-peer distributed reputation schemes. In *First International Conference on Collaborative Computing*, 2005.
51. C. Y. Liau, X. Zhou, S. Bressan, and K. Tan. Efficient distributed reputation scheme for peer-to-peer systems. In *Second International Conference on Human Society@Internet*, 2003.
52. R. Gupta and A. K. Somani. Reputation management framework and its use as currency in large-scale peer-to-peer networks. In *Fourth International Conference on Peer-to-peer Computing*, 2004.
53. L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 2004.
54. S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Twelfth International Conference on World Wide Web*, 2003.
55. D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical report, Hewlett-Packard, 2002.
56. S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking*, 2002.
57. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *19th International Conference on Data Engineering*, 2003.
58. J.L. Munoz M. Rodriguez-Perez, O. Esparza. Surework: Un sistema de reputación para redes p2p basado en super-peers. In *VI Jornadas de Ingeniería Telemàtica (JITEL'07)*, 2007.
59. K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Tenth International Conference on Information and Knowledge Management*, 2001.
60. L. Mekouar, Y. Iraqi, and R. Boutaba. Detecting malicious peers in a reputation-based peer-to-peer system. In *2nd IEEE Consumer Communications and Networking Conference*, 2005.
61. J. Han and Y. Liu. Dubious feedback: fair or not? In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, 2006.
62. H. Rifa and J. Herrera. A fair and secure cluster formation process for ad hoc networks. *Wireless Personal Communications*, 56(3):625–636, 2010.
63. Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley Professional, 1 2001.
64. Konrad Wrona and Petri Mähönen. Analytical model of cooperation in ad hoc networks. *Telecommunication Systems*, Volume 27:347 – 369, 2004.
65. Marcela Mejia, Néstor Peña, José. L. Muñoz, and Oscar. Esparza. A review of trust modeling in ad hoc networks. *Internet Research*, volume 19(1):88–104, 2009.

66. Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.
67. G. F. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas. Cooperation enforcement schemes for manets: a survey: Research articles. *Wireless Communications and Mobile Computing*, 6(3):319–332, 2006.
68. Levente Buttyán and Jean-Pierre Hubaux. Enforcing service availability in mobile ad-hoc wans. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 87–96, Piscataway, NJ, USA, 2000. IEEE Press.
69. Sheng Zhong Yale and Sheng Zhong. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *in Proceedings of IEEE INFOCOM*, pages 1987–1997, 2002.
70. Hamed Janzadeh, Kaveh Fayazbakhsh, Mehdi Dehghan, and Mehran S. Fallah. A secure credit-based cooperation stimulating mechanism for manets using hash chains. *Future Gener. Comput. Syst.*, 25(8):926–934, 2009.
71. S. Bansal and M. Baker. Observation-based cooperation enforcement in ad hoc networks. Technical report, 2003.
72. Luzi Anderegg and Stephan Eidenbenz. Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 245–259, New York, NY, USA, 2003. ACM.
73. Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the confidant protocol. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 226–236, New York, NY, USA, 2002. ACM Press.
74. Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *In Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.
75. P. Michiardi and R. Molva. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile Ad Hoc networks. *6th IFIP Communications and Multimedia Security Conference*, page 107–121, 2002.
76. Haidar Safa, Hassan Artail, and Diana Tabet. A cluster-based trust-aware routing protocol for mobile ad hoc networks. *Wirel. Netw.*, 16(4):969–984, 2010.
77. Yan Lindsay Sun, Wei Yu, Zhu Han, and K.J.R. Liu. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):305–317, Feb. 2006.
78. Rob Sherwood, Seungjoon Lee, and Bobby Bhattacharjee. Cooperative peer groups in nice. *Computer Networks*, 50(4):523–544, 2006.
79. Martin Osborne. *an Introduction to Game Theory*. OXFORD University Press, 2004.
80. Walid Saad, Zhu Han, Mérouane Debbah, and Are Hjørungnes. A distributed coalition formation framework for fair user cooperation in wireless networks. *Transactions on Wireless Communications*, 8(9):4580–4593, 2009.
81. M. Felegyhazi, J.-P. Hubaux, and L. Buttyan. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(5):463–476, May 2006.
82. Marcin Seredynski and Pascal Bouvry. Evolutionary game theoretical analysis of reputation-based packet forwarding in civilian mobile ad hoc networks. In *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

83. Fabio Milan, Juan José Jaramillo, and R. Srikant. Achieving cooperation in multihop wireless networks of selfish nodes. In *GameNets '06: Proceeding from the 2006 workshop on Game theory for communications and networks*, page 3, New York, NY, USA, 2006. ACM.
84. K. Komathy and P. Narayanasamy. Trust-based evolutionary game model assisting aodv routing against selfishness. *Journal of Network and Computer Applications*, 31(4):446–471, 2008.
85. Kun Wang, Meng Wu, Chao Ding, and Weifeng Lu. Game-based modeling of node cooperation in ad hoc networks. In *19th Annual Wireless and Optical Communications Conference (WOCC)*, 2010.
86. Zhu Ji, Wei Yu, and K.J. Ray Liu. A belief evaluation framework in autonomous manets under noisy and imperfect observation: Vulnerability analysis and cooperation enforcement. *IEEE Transactions on Mobile Computing*, 9:1242–1254, 2010.
87. Marcela Mejia, Néstor Peña, José. L. Muñoz, Oscar. Esparza, and Marco-Aurelio Alzate. A game theoretic trust model for on-line distributed evolution of cooperation in manets. *Journal of Network and Computer Applications*, volume 34(1):39–51, 2011.
88. Marcela Mejia. *Evolución Genética de Estrategias para Modelos de Confianza en Redes Móviles Ad Hoc Basados en Teoría de Juegos*. PhD thesis, Universitat Politècnica de Catalunya (UPC, Spain) and Universidad de los Andes (Colombia), 2011.
89. M. Seredynski, P. Bouvry, and M.A. Klopotek. Modelling the evolution of cooperative behavior in ad hoc networks using a game based model. *IEEE Symposium on Computational Intelligence and Games CIG.*, pages 96–103, April 2007.
90. Jeremy Dale and Simon Park. *Molecular Genetics of Bacteria*. Wiley, 4 edition, 3 2004.
91. I. W. Marshall and C. Roadknight. Adaptive management of an active service network. *BT Technology Journal*, 18(4):78–84, 2000.
92. E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms (Operations Research/-Computer Science Interfaces Series)*. Springer, 1 edition, 6 2008.
93. Enrique Alba and José M. Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–52, 1999.
94. Erick Cantupaz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Réseaux et Systémes Répartis*, 10(2):141–171, 1998.
95. M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, pages 88–92, 1999.
96. Muñoz, J. L. and Esparza, O and Aguilar, M and Carrascal, V and Forné, J. Rdsr-v. reliable dynamic source routing for video-streaming over mobile ad hoc networks. *Computer Networks*, 54(1):79–96, 2010.
97. S. Bansal and M. Baker. Observation-based Cooperation Enforcement in Ad Hoc Networks (OCEAN), arxiv:cs.ni/0307012 v2. Technical report, Stanford University, 2003.
98. Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with cfs. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM Press.
99. Juan Vera del Campo, J. Hernández-Serrano, and J. Pegueroles. Scfs: Towards design and implementation of a secure distributed filesystem. International Con-

ference on Security and Cryptography (SECRYPT), International Conference on Security and Cryptography (SECRYPT), 07/2008 2008.
100. Juan Vera del Campo, J. Hernández-Serrano, and J. Pegueroles. Análisis de seguridad de un sistema de archivos distribuido. X Reunión Espanola sobre Criptología y Seguridad de la Información (RECSI'08), 09/2008 2008.
101. Scfs: lewis.upc.es/svn/dfs, 2008.
102. Michael O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335 – 348, April 1989.