UNIVERSIDAD CARLOS III DE MADRID

Ph.D. THESIS

Content Authentication and Access Control in Pure Peer-to-Peer Networks

Author:
Esther Palomar González

Supervisors:
Dr. D. Juan M. Estévez Tapiador
Dr. D. Arturo Ribagorda Garnacho

Computer Science Department

Leganés, April 2008

UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Autenticación de Contenidos y Control de Acceso en Redes Peer-to-Peer Puras

Autor:
Esther Palomar González

Directores:
Dr. D. Juan M. Estévez Tapiador
Dr. D. Arturo Ribagorda Garnacho

Departamento de Informática

Leganés, Abril 2008

TESIS DOCTORAL

CONTENT AUTHENTICATION AND ACCESS CONTROL IN PURE PEER-TO-PEER NETWORKS

Autor: Esther Palomar González

Directores: Dr. D. Juan M. Estévez Tapiador
Dr. D. Arturo Ribagorda Garnacho

Firma del Tribunal Calificador:

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Calificación:

Leganés, de de 2008

A mi madre.

Acknowledgements

I would like to acknowledge the help of many people who get this done.

First, I would like to thank my Ph.D. Supervisors. I would like to express my gratitude to Juan, for his good ideas and numerous comments on chapter drafts. I am extremely grateful to Arturo, who has assisted me in numerous ways.

I would also like to thank all the rest of my colleagues. Benjamín, for all those lunches and fresh discussions. Anabel, for encouraging me not to lose heart, and for the practical help with different problems. Thanks to Julio, who selflessly gave me this gratifying opportunity. Almudena, Agustín, Chema, and Jorge, for all the emotional support and interest. Special thanks to Peris, for the music, for understanding... *Y hasta aquí puedo leer, verdad, Peris!*

Finally, I wish to thank my parents for everything. In particular, the impatience shown by Mama throughout these years encourages me to meet the deadlines and reach my objectives. *Gracias mama!* Papa, San, Carlos (and Conan), on whose constant encouragement and love I have relied. I cannot leave out my friends: my best friends from the elementary school, Chisi and Fer, Emily, Carlota, Marciano, Alex..., for believing in me and in my possibilities.

Abstract

The study and analysis of the state-of-the-art on security in Peer-to-Peer (P2P) networks gives us many important insights regarding the lack of practical security mechanisms in such fully decentralized and highly dynamic networks. The major problems range from the absence of content authentication mechanisms, which address and assure the authenticity and integrity of the resources shared by networking nodes, to access control proposals, which provide authorization services. In particular, the combination of both, authentication and access control, within well-known P2P file sharing systems may involve several advances in the content replication and distribution processes.

The aim of this thesis is to define, develop and evaluate a secure P2P content distribution scheme for file sharing scenarios. The proposal will be based on the use of digital certificates, similar to those used in the provision of public key authenticity. To carry out this proposal in such an environment, which does not count on a hierarchy of certification authorities, we will explore the application of non-conventional techniques, such as Byzantine agreement protocols and schemes based on "proof-of-work."

We then propose a content authentication protocol for pure P2P file sharing systems. Under certain restrictions, our scheme provides guarantees that a content is authentic, i.e. it has not been altered, even if it is a replica of the original and the source has lost control over it. Moreover, we extend our initial work by showing how digital certificates can be modified to provide authorization capabilities for self-organizing peers.

The entire scheme is first theoretically analyzed, and also implemented in `C` and `Java` in order to evaluate its performance.

This document is presented as Ph.D. Thesis within the 2007–08 Ph.D. in Computer Science Program at Carlos III University of Madrid.

Resumen

Esta tesis doctoral se enmarca dentro del área de investigación de la seguridad en entornos Peer-to-Peer (P2P) totalmente descentralizados (también denominados puros.) En particular, el objetivo principal de esta tesis doctoral es definir, analizar e implementar un esquema para la distribución segura de los contenidos compartidos.

En este trabajo de tesis se han realizado importantes avances e innovadoras aportaciones enfocadas a garantizar que el contenido compartido es auténtico; es decir, que no ha sido alterado, incluso tratándose de una réplica del original. Además, se propone un mecanismo de control de acceso orientado a proporcionar servicios de autorización en un entorno que no cuenta con una jerarquía de autoridades de certificación.

A continuación, se resume la metodología seguida, las principales aportaciones de esta tesis y, finalmente, se muestran las conclusiones más importantes.

0.1 Metodología

Analizando exhaustivamente el estado de la cuestión relacionado con los servicios de seguridad en las redes P2P, se pueden detectar diversas carencias en las propuestas existentes, tales como la falta de esquemas que aseguren la autenticidad e integridad de los contenidos compartidos por los nodos de red. Especialmente, los esquemas de control de acceso presentados hasta la fecha deben ser cuidadosamente adaptados a este tipo de redes totalmente descentralizadas y altamente dinámicas.

Tras completar el análisis anterior, el objetivo principal de esta tesis doctoral es definir, desarrollar y analizar un modelo para la distribución segura de contenidos en sistemas P2P de intercambio de contenidos. La solución prop-

uesta se basa en el empleo de certificados digitales, análogos a los utilizados para garantizar la autenticidad de una clave pública (estándar X.509.) Para llevar a cabo esta tarea en un entorno que no cuenta con una jerarquía de autoridades de certificación, se ha estudiado el uso de técnicas no convencionales, tales como los protocolos de acuerdo bizantino o las pruebas de esfuerzo.

La propuesta inicial, que proporciona autenticación de contenidos P2P, es refinada sucesivamente mediante su análisis o verificación hasta lograr los objetivos de seguridad y eficiencia deseados. Como resultado de esta evaluación, extendemos el esquema hacia un modelo de control de acceso. Por último, se desarrolla un prototipo software que implemente las propuestas de esta tesis.

0.2 Principales Aportaciones

Las redes P2P suponen un nuevo paradigma dentro de las tecnologías de la información con nuevos retos desde el punto de vista de la Seguridad. En un sistema P2P, todos los nodos desempeñan el mismo rol, pudiendo ejercer tanto de servidores como de clientes simultáneamente. Por otro lado, debido a la expansión de los dispositivos móviles y de los entornos ubicuos, están apareciendo nuevas aplicaciones P2P en entornos ad hoc. Estas redes presentan ciertas peculiaridades en cuanto a su arquitectura, y se caracterizan principalmente por su descentralización y capacidad de autoorganización.

A continuación, enumeramos las principales motivaciones de la presente tesis doctoral:

- El paradigma P2P constituye un entorno de cómputo y comunicación distribuido en el que cualquier nodo puede completar transacciones con, a priori, cualquier otro nodo. La investigación en este campo ha sido intensa, habiendo sido propuestas hasta la fecha diversas arquitecturas y topologías basadas especialmente en la colaboración honesta entre una comunidad heterogénea de participantes. Sin embargo, las características inherentemente distintas de este modelo con respecto al ya clásico paradigma Cliente-Servidor hacen que muchas de las soluciones de seguridad existentes no sean aplicables o, en todo caso, necesiten ser cuidadosamente adaptadas. En esta tesis se han revisado los aspectos más representativos concernientes a la seguridad de las redes P2P, incluyendo un

análisis comparativo entre las arquitecturas más relevantes propuestas hasta la fecha.

- Entre las numerosas ventajas proporcionadas por los sistemas P2P, quizás la más interesante sea la capacidad de ofrecer réplicas del mismo contenido en diferentes localizaciones. Nos referimos a los sistemas intercambio de contenidos tan populares hoy en día entre extensas comunidades de nodos. Estos sistemas permiten acceder a los contenidos incluso cuando existen nodos desconectados. Sin embargo, el alto grado de redundancia conlleva la necesidad de disponer de ciertos mecanismos de seguridad a fin de evitar ataques basados en la modificación no autorizada de los contenidos compartidos.

- Existen muchas razones por las que es necesario proporcionar autenticación del contenido, cuyo objetivo sea evitar que nodos con comportamiento malicioso puedan crear desconfianza en la comunidad. Una vez un contenido es replicado en la red, el autor pierde el control sobre el mismo, pudiendo aparecer nodos deshonestos que alteren el contenido con diversos propósitos: reclamar la autoría del contenido publicado, insertar código malicioso (virus y *spyware*) en contenidos con mucha demanda, u ofrecer falsos contenidos. Estos problemas demandan mecanismos de seguridad que ofrezcan servicios de integridad y autenticidad, tanto de los nodos como de los contenidos compartidos.

- Por otro lado, debido a la inexistencia de terceros de confianza (o TTPs del inglés, *Trusted Third Parties*) en este tipo de entornos descentralizados, las soluciones tradicionales, que descansan en la utilización de autoridades de certificación (AC), son difícilmente adaptables. De hecho, esto implica la necesidad de buscar soluciones alternativas, ya que no es realista pensar en la existencia de una AC en este tipo de entornos. Nuestro modelo depende de la colaboración entre un grupo de nodos, quienes conjuntamente desempeñan el papel de una AC. Sin embargo, existen graves problemas de complejidad y escalabilidad, especialmente si queremos contar con infraestructuras de clave pública.

  En esta tesis, se propone, en primer lugar, un protocolo para dotar a una red P2P pura de autenticación del contenido, basado en el uso de certificados digitales y de firma digital. Brevemente, la idea principal

es que a todo contenido publicado en la red le acompañe un certificado firmado por un grupo de nodos honestos o confiables. Este conjunto de firmantes desempeñan conjuntamente el papel de una AC. Para ello, se dispondrá del protocolo de autenticación de claves públicas para entornos totalmente descentralizados presentado por Pathak e Iftode [97], que también implementamos y analizamos en el presente trabajo.

- El protocolo anterior es posteriormente extendido, mostrando cómo los certificados digitales pueden ser modificados con el fin de proporcionar capacidades de autorización a los nodos en un entorno auto-organizativo. Se propone, por tanto, un mecanismo de control de acceso que permita a los propietarios/distribuidores de contenidos controlar los permisos de acceso a sus contenidos dependiendo de la política de seguridad local establecida por dichos propietarios.

- Asociados al concepto de control de acceso se encuentran los servicios de revocación y de delegación de privilegios. En esta tesis se propone, por un lado, un subprotocolo de delegación orientado a controlar la replicación de los contenidos; y, por otro lado, un subprotocolo de revocación que gestione la validez de los privilegios de cada nodo en el tiempo.

- Se ha realizado un análisis de la eficiencia y escalabilidad del esquema empleando umbrales de confianza y técnicas de pruebas de esfuerzo (en inglés, *proof-of-work*.) Asimismo, el esquema se ha evaluado desde el punto de vista de la seguridad, proponiendo numerosos escenarios de ataque posible y verificando informalmente su seguridad en cada caso.

- El esquema completo se analiza, en primer lugar, teóricamente, y posteriormente es implementado en `C` y `Java` con el objetivo de evaluar su rendimiento y validar su correctitud.

Los objetivos principales de esta tesis son, por tanto, primero formalizar los servicios de seguridad básicos necesarios en sistemas P2P de intercambio de contenidos; y, segundo, desarrollar y diseñar un esquema de seguridad de forma que se solucionen los problemas ilustrados en los párrafos anteriores. Estos objetivos se pueden desglosar en los siguientes:

- **Estado de la cuestión**, en el que se explora y formaliza la necesidad de servicios de seguridad básicos, como son la autenticación de contenidos y el control de acceso.

- **Diseño** de un esquema de distribución segura de contenidos para redes totalmente descentralizadas. El esquema presentado se estructura como sigue:

  - Protocolo de autenticación de contenido.
  - Protocolo de control de acceso.
  - Servicios avanzados de autorización.

- **Implementación** de los protocolos anteriores, incluyendo el de autenticación de clave pública basado en acuerdo bizantino para redes ad hoc.

- **Análisis y evaluación** del esquema de intercambio seguro de contenidos obtenido, respecto a los siquientes términos:

  - **Análisis de la eficiencia**. Generalmente, la provisión de mecanismos de seguridad ha ido acompañada de una significativa sobrecarga computacional y de comunicaciones, sobre todo en entornos distribuidos. Los resultados obtenidos en esta tesis has sido cuantificados experimentamente atendiendo a tiempo computacional y de mensajes intercambiados en los protocolos.
  - **Análisis de la seguridad**. El esquema completo garantiza los requisitos fundamentales de seguridad, habiendo sido analizado a través de los principales escenarios posibles.

0.3 Conclusiones

De las novedosas aportaciones presentadas en esta tesis, y de las características del esquema propuesto, es posible extraer varias conclusiones:

- El análisis que se ha realizado acerca del estado de la cuestión de las propuestas relacionadas con los esquemas de seguridad en redes P2P y ad hoc ha mostrado múltiples aspectos aún sin una solución ampliamente

aceptada. Se podría afirmar que tales carencias son debidas fundamentalmente a dos hechos concretos: la hetereogeneidad de tales entornos, y por tanto de las arquitecturas propuestas, y en segundo lugar la falta de una perspectiva global a la hora de afrontar retos de seguridad. El trabajo de tesis ha considerado estos dos factores a la hora de proporcionar una solución completa al problema de la replicación segura de los contenidos compartidos por los sistemas comunes de intercambio de ficheros en Internet y en entornos colaborativos.

- La suma de los protocolos presentados permite, en su conjunto, una replicación segura de los contenidos compartidos entre los nodos de una comunidad P2P, los cuales son capaces de detectar posibles alteraciones no autorizadas sobre ellos. Bajo ciertas suposiciones, nuestro esquema garantiza que el contenido compartido es auténtico, es decir, que no ha sido alterado, incluso tratándose de una réplica del original y, por tanto, el propietario ya no tiene control sobre él.

- El modelo presentado está basado en el uso de certificados digitales y no requiere los servicios de un tercero de confianza, entidad impracticable en entornos P2P descentralizados. Las propuestas realizadas han sido implementadas y posteriormente validadas en entornos de aplicación concretos con diversos requisitos.

- Tras el estudio de las propuestas más representativas en lo relativo a modelos de autorización y control de acceso en sistemas distribuidos, se presenta un mecanismo de autorización apto para entornos que no cuentan con puntos centrales de decisión. La gestión de los privilegios en estas cirscunstancias es un reto muy considerado por la comunidad investigadora.

En lo que respecta a la extensión de las soluciones y aportaciones presentadas en el presente trabajo de tesis, algunas líneas de investigación futuras son las siguientes:

- Estudiar exhaustivamente la aplicación de mecanismos alternativos, tales como la criptografía umbral y los esquemas multifirma, para proporcionar soluciones al mismo problema. Éstas técnicas han sido aplicadas en algunos trabajos dentro del area de las redes ad hoc para diseñar,

por ejemplo, modelos de encaminamiento, mecanismos de incentivos y de control de admisión, entre otros.

- Extender la arquitectura a entornos móviles y/o MANETs, sin olvidar aplicaciones íntimamente relacionadas como las redes GRID. Algunas de las propuestas de seguridad presentadas por la comunidad investigadora constituyen, en algunos casos, mecanismos ya muy explotados (por ejemplo, sistemas de confianza y anonimato) y cuya aplicabilidad está en entredicho cuando la movilidad y la cooperación son esenciales.

- Integrar y analizar el establecimiento de una política de autorización global para todos los miembros de la comunidad. Evaluar los resultados que se pueden obtener en entornos corporativos.

- Otra línea de investigación ya iniciada es el análisis de la idoneidad de nuestra propuesta a través del uso de Teoria de Juegos. Este enfoque nos proporciona una visión del comportamiento del protocolo para diferentes tipos de comunidad, según la naturaleza cooperativa de los nodos.

En resumen, la posible extensión de los servicios de seguridad presentados como resultado de esta tesis representa un punto de partida de enfoques y trabajos posteriores que podrían formar parte del uso futuro de tales sistemas de intercambio de contenidos.

Contents

List of Figures

List of Tables

Symbols and Terminology

A, B, ...	Specific nodes will be occasionally designated by capital letters.
CA	Certification Authority.
C_m	Content certificate associated with m.
$C_{n_i}^{n_j}$	Authorization certificate issued by n_j to n_i.
CRL	Certificate Revocation List.
DoS	Denial of Service.
$E_{K_{n_i}}(x)$	Asymmetric encryption of message x using K_{n_i} as key.
$enc(x, K_S)$	Symmetric encryption of message x using a secret key K_S.
$h(x)$	A cryptographic hash function applied to x.
K_{n_i}	Public key of node n_i.
$K_{n_i}^{-1}$	Private key of node n_i.
K_S	A session key used with a symmetric cipher.
L_i	A security label/clearance.
m	A content.
MANET	Mobile and Ad Hoc Network.
N	Number of network nodes.
n_i	Node identification.
OLS	An ordered list of signers.
P2P	Peer-to-Peer.
PGP	Pretty Good Privacy.
PKI	Public Key Infrastructure.
PMI	Privilege Management Infrastructure.
$s_i(x)$	n_i's signature over x, i.e.: $s_i(x) = E_{K_i^{-1}}(h(x))$.
S_{n_i}	Local data base which keeps track of content certificates previously signed by n_i.
Ss_{n_i}	Local register containing the subscribers of n_i.
TCP	Transmission Control Protocol.
TLS	Transport Layer Security.
TMS	Trust Management System.
T_{n_i}	Local register containing the trusted nodes of n_i.
ts_i	A timestamp.
TTP	Trusted Third Party.

Chapter 1

Introduction

Advances in distributed systems with increasingly growing capabilities for efficient file transport (both wired and mobile), and their immediate consequence, i.e. the ability to rapidly replicate a content over a network, have made sharing of electronic files become a revolution in business and domestic environments. The extreme decentralization, dynamism, and self-organization of a number of emerging environments, including teamwork, pure Peer-to-Peer (P2P), and mobile ad hoc networks (MANET), where nodes are involved in the process of sharing and collaboration without relying on central authorities, enforce cooperation to play an essential role in the overall network functioning. Particularly, ad hoc networks rely upon the cooperation among individual nodes to carry out essential tasks such as packet forwarding. P2P file sharing systems face a similar situation.

File sharing has become a common practice for Internet users to obtain, for example, software updates from public sites. However, such a practice still provokes mistrust. File corruption may occur easily through dishonest and malicious actions or even by mistake. Similarly, an impostor could masquerade himself as the originator of a certain file, publishing a corrupted version of the file. In fact, users of currently deployed file sharing systems are unable to verify that files they retrieve are uncorrupted, or whether the content has been truly created by the presumed owner.

Providing efficient security services in P2P and ad hoc networks is an active research area which prompts many challenges [130]. Researchers have to adapt common cryptographic techniques, e.g. threshold and usual public key cryptography, to highly dynamic environments to ensure that even when some

nodes are unavailable, others can still perform the task through the coalition of cooperating parties [27]. This collaboration-based nature of the communication layers imposes a number of challenges in the provision of other services, especially concerning security. In fact, most of the difficulties found to apply classic security solutions are just related to the inherent lack of central authorities, such as public key infrastructures (PKI), and new proposals have to deal with avoiding such a centralization by exploring alternative paradigms which, in turn, require cooperation among peers.

For instance, many of the network security services offered today rely in public key cryptography. However, one of the most important issues when dealing with public keys is ensuring their authenticity. In distributed environments, the classic solution relies on the existence of a PKI. A hierarchy of Certification Authorities (CA) can assure whether a public key belongs to someone or not, as well as some additional services (e.g. if the expiration date has passed.) Nevertheless, it is not realistic to assume that trusted third parties (TTP) can be deployed in most pure P2P networks, especially in the case of a mobile ad-hoc network, where the lack of fixed infrastructure makes it particularly difficult [91].

Furthermore, among the main security properties demanded on a system, authentication has been identified as a critical issue in self-organized environments [90]. Several approaches have addressed the provision of fault tolerant authentication, most in the way of protocols relying on threshold cryptography using key sharing and agreement techniques [41]. The key idea is to use a quorum of participants to create a digitally signed public key certificate. On the other hand, resource authentication become critical since a content is replicated through different locations, and therefore the originator loses control over it.

1.1 Objectives and Scope

This thesis studies and analyzes the above characteristics for the provision of security services, paying special attention to authentication and access control techniques suitable for self-organized networks. We propose a content authentication scheme whose main objective is to maintain content integrity, ensuring its authenticity and avoiding non-authorized alterations. Moreover, since it is

proved that sharing can be encouraged by imposing a cost on the downloads, the scheme is extended in this regard by means of providing access control services. We next briefly introduce the motivations of this thesis.

1.1.1 Motivation

In a P2P network, peers communicate directly with each other to exchange information. One particular example of this information exchange, that has been rather successful and has attracted considerable attention in the last years, is file sharing. This kind of systems are typically made up of millions of dynamic peers involved in the process of sharing and collaboration without relying in central authorities. P2P systems are characterized by being extremely decentralized and self-organized. These properties are essential in collaborative and ad-hoc environments, in which dynamic and transient population prevails.

The popularity and inherent features of these systems have motivated new research lines in the application of distributed P2P computing. New problems have also been posed, such as scalability, robustness and fault tolerance, organization and coordination, adaptability, distributed storage, location and retrieval, reputation, and security. In particular, security advances have focused on anonymity, access control, integrity, and availability. New areas are being explored, such as fairness and authentication [37]. Additionally, a significant part of the research on security in P2P systems intends to mitigate attacks against four main system properties: availability, authenticity, access control, and anonymity. Recent work primarily focuses on addressing attacks against availability and authenticity [39]. For instance, some results already exist on the provision of both security properties in traditional Gnutella-like systems. Different authors have also studied how to use a P2P network to prevent Denial of Service (DoS) attacks on the Internet [67, 81]. On the other hand, works such as [32] study how to use P2P networks to provide user anonymity. Furthermore, current architectures for P2P networks are plagued with open (and apparently difficult to solve) issues in digital rights management and access control (e.g., [55] outlines some of the problems in this area.) The foregoing problems in this field are presented and an advancement is provided with this thesis for a method to securely distribute electronic files in a fully distributed and self-organized P2P network.

On the other hand, one of the main advantages of P2P systems is their

capability to offer replicas of the same content at various locations. Faced with different locations of the same content, an application can grant priority to that which offers a less expensive path (e.g. in terms of bandwidth.) To some extent, replication also guarantees some sort of fault tolerance, since information can be available even if some parts of the network are temporarily disconnected. However, this high degree of redundancy implies that it is necessary to apply some security mechanisms in order to avoid attacks based on non-authorized content modification, i.e. to detect whether a content has been altered, and also to verify whether it was actually sent by the person/entity claimed to be the sender.

Early approaches to providing integrity of electronic files were based on assuring the communication environment through verifying the transmissions, i.e. ensuring the provider and the requester, the source and the original content. Such mechanisms apply symmetric key cryptography for message transmissions by means of secure channels using a shared secret, e.g. applying secret key encryption algorithms (block ciphers and stream ciphers.) Encryption of messages guarantees against unauthorized access, and serves for "distinguishing" the other party between the defined, limited universe of users. Despite its speed and low computational requirements, the key management is difficult to achieve reliably and securely.

An advance in the question was the use of asymmetric (or public key) algorithms in order to distribute symmetric keys at the start of a session. In practice, public key cryptography is commonly used to bind the public keys to a user name (in the form of digital certificates – X.509 ITU-T standard for PKI), and can be used to implement digital signature schemes, i.e. to sign a message by creating a hash of the message and then encrypting the hash with a private key. If a message is digitally signed, any change in the message will invalidate the signature. As a result, the sender identity can also be permanently tied to the content of the message being signed. The verification of the signature depends on being in the possession of the corresponding public key. After decrypting the hash, anyone can compare the hash from the signature to that generated from the message. Nevertheless, because asymmetric keys have to be distributed authentically, a public directory is required.

Additionally, another progress is the process of securely keeping track of the creation and modification time of a file, fixing the time and content of a

document. A central entity, a timestamping authority (TSA), is also needed to ensure data integrity against a reliable time source; such an information is in turn digitally signed with the private key of the TSA. Thus, a trusted party (typically a CA) facilitates the interactions between two parties who both trust her. This contrasts with web-of-trust models in which there is not reliance exclusively on a centralized CA (or a hierarchy of such.)

Finally, the vast majority of these schemes to do message integrity checks and digital signatures use cryptographic hash functions. The hash value, or digest, of a file is a concise representation of that file as a sort of digital fingerprint of it. Similarly, a cryptographic message authentication code (MAC) value protects both message's integrity as well as its authenticity using as input a secret key, as is the case with symmetric encryption.

The foregoing classic approaches have some limitations when applied to ensure that a file a user is downloading is the original file from the real author. In fact, these deficiencies in current systems are a burden on the secure P2P content distribution, thereby limiting the use of these networks for sharing files in a manner on which users can rely on [102].

1.1.2 Summary of the Proposal: P2P Content Authentication and Access Control based on Byzantine Agreement

In this section, we provide a brief summary of the contributions of this thesis, as follows:

(a) The study and analysis of the state of the art on security in P2P networks gives us lacking of content authentication and authorization proposals. We present a comparative analysis of existing solutions, some of which will be addressed in this thesis. Concretely, we especially elaborate on the concept of cooperation-based security services.

(b) A recent work addresses the issues related to this problem in P2P systems [97]. Authors introduce a scheme based on Byzantine agreement for authenticating public keys. The proposed mechanism is autonomous and does not require the existence of a TTP in charge of issuing and verifying certificates to ensure key authenticity. The key point is that

the scheme works correctly if the number of honest peers in the network is above a certain threshold. We implement this scheme, and analyze it as well, aimed at building high level security services in an upper layer.

(c) We propose a content authentication protocol for pure P2P file sharing systems. Under certain restrictions, our scheme provides guarantees that a content is authentic, i.e. it has not been altered, even if it is a replica of the original and the source has lost control over it. Our proposal employs a combination of three main cryptographic concepts which are well known. These are: hash functions, digital signatures and signed certificates. Moreover, our scheme relies on a set of peers playing the role of a CA, for it is unrealistic to assume that appropriate TTPs can be deployed in such environments. An exhaustive report on the efficiency is presented. We also discuss some of its security properties through several attack scenarios. For this, we need an implementation and implantation of the obtained file sharing system.

(d) Many inherent characteristics of P2P environments introduce new requirements for access control, such as different applications running on each node, control decentralization, and node mobility and dynamics, to name a few. In general, the extremely decentralized nature of a pure P2P network makes impossible to apply solutions that rely on some kind of fixed infrastructure, such as on-line TTPs. We extend our initial work by showing how digital certificates can be modified to provide authorization capabilities for self-organizing peers.

(e) Finally, since classic access control schemes must be adapted to such fully decentralized and highly dynamic networks, we also discuss on the provision of authorization services, especially delegation and revocation.

Summarizing, the main subject of this thesis will be the design of a secure content distribution protocol for pure P2P networks. For this, we rely on some results derived from well-studied problems such as those of reaching consensus in presence of traitors. Content authentication confirms non-alteration and source identification of the content, implemented through a digital content certificate. Once authentication is provided, access control schemes could be designed.

1.1.3 Validating Results

This thesis presents several results obtained from the empirical analysis of the specification of our proposal, and from the development and evaluation of the entire scheme as well. Our motivated scenarios are the implementation of our protocol in future P2P networks, particulary in collaborative environments and file sharing applications in order to make them more reliable and secure. In fact, the implementation of the scheme shows that the overhead of the entire process is low enough, to some extent, and therefore its real application is possible in current P2P networks.

1.2 Organization of this Document

The remainder of this document is structured as follows.

A brief overview of the state of the art on security in P2P and MANETs is presented in Chapter 2. First, we motivate the need for basic security solutions for pure P2P and ad hoc networks, where cooperation among nodes is required. Finally, we overview main security approaches proposed so far by the research community, and discuss the most interesting properties they provide.

In Chapter 3, we present our approach aimed at assuring content authentication. An exhaustive description of all the protocol components and phases is provided, together with an efficiency and security analysis.

Moreover, this proposal is extended in order to provide access control in Chapter 4. Along with content authentication, our Join subprotocol and Content Access subprotocol guarantee a secure content distribution in a fully decentralized P2P network. We also address the provision of authorization services, i.e. delegation and revocation of privileges. The entire proposal is further analyzed in terms of the computational and communication overheads introduced by the security scheme.

Chapter 5 shows a software implementation and testing in order to evaluate and analyze the functional requirements of our proposal. Finally, Chapter 6 sets some conclusions and research directions.

In Appendix A, we briefly describe the implantation of the public key authentication protocol and its validation. Furthermore, Appendix B is devoted to examine the availability of computational puzzles and the additional cost spent in solving cryptographic puzzles through numerous simulations.

Chapter 2

Security in P2P Networks: State-of-the-Art

Research on fully decentralized and self-organized networks, such as P2P or MANETs, is receiving special attention at present due to the emerging range of new applications and their real deployment. Note that if a centralized systems is generally difficult to handle, a distributed one often poses much more difficulties. However, the inherent differences between the P2P model and the classic client-server paradigm cause that many solutions developed for the latter are not applicable or, in the best case, have to be carefully adapted. Concretely, the provision of security services introduces lots of research challenges. It is particulary challenging because of the unique characteristics of such environments, especially mobility and heterogeneity, dynamic node joins and leaves, and routing issues.

Generally, the inherent lack of fixed infrastructures and a high transient topology of the physical network, formed and maintained independently by the peers, can cause significant problems. In fact, most of the difficulties found to apply classic security requirements (e.g. authentication and authorization services) in a P2P system are just related to the absence of a TTP functionality, such as the establishment of a PKI. Moreover, this makes difficult the application of many common cryptographic primitives such as digital signatures. New proposals have to deal with avoiding such a centralization by exploring alternative paradigms which, in turn, require cooperation among peers. This fact enforces the overall cooperation to play an essential role. Furthermore, researchers have to adapt common cryptographic techniques, e.g.

threshold and public key cryptography, to highly dynamic environments to ensure that even when some nodes are unavailable, others can still perform the task through the coalition of cooperating parties. Nearly all the solutions proposed so far are one way or another based on the idea of replacing the whole PKI by a collaboration scheme. Put simply, a subgroup of peers must cooperate to perform tasks such as generating (or verifying) a digital signature, o negotiating a group key for a secure communication. In particular, ad hoc networks rely upon the cooperation among individual nodes to carry out essential tasks such as packet forwarding. P2P file sharing systems face a similar situation. This collaboration-based nature of the communication layers guarantees fault-tolerance though, however, imposes a number of drawbacks as well.

However, collaboration-based applications have to deal with the social and rational dilemma, which motivates a possible tendency of entities towards self-interested behavior [25]. This way, a fundamental feature of P2P networks is the *honest* collaboration among an heterogeneous community of participants. For example, a node may decide not to cooperate to save its resources while still using the network. After `Napster` success –the first P2P file sharing application massively used–, advances in this area have been intense, with the proposal of many new architectures and applications for content and computing sharing, and collaborative working environments. Most research efforts have modeled and quantified the incentives and disincentives for cooperation in P2P networks [80]. On the other hand, anonymity and a highly transient population incur additional complexity for peers to determine others' identities, making difficult (or very inefficient) some tasks such as authentication and accounting.

In this chapter, we present a survey on security issues in P2P networks. Concretely, Section 2.1 leads to the description of the P2P system and adversary model, especially elaborating on the concept of cooperation-based security services, such as authentication and access control. We discuss in Section 2.2 most representative cryptographic primitives used to commonly provide security in a fully distributed manner. Moreover, Section 2.3 summarizes main

[11]Several works have applied rational models, such as the Prisoners' Dilemma, in order to empirically observe a networking social pattern. We will further elaborate on this topic in next chapter.

security proposals presented so far, some of which will be addressed in this thesis. We also identify several emerging trends in this research area (see Section 2.4), and provide a comparative analysis of existing solutions in Section 2.5. Finally, we conclude by identified some open research issues in Section 2.6.

2.1 Pure P2P Networks

P2P is often described as a type of distributed computing paradigm where nodes communicate directly with each other to exchange information. P2P applications allow users to communicate synchronously, supporting tasks such as instant messaging, working on shared documents or sharing files, among many others. As a result, the P2P paradigm provides users with the capability of integrating their platforms within a distributed environment with a broad range of possibilities.

A P2P network has neither clients nor servers; each individual node could act simultaneously as a client and as a server for the rest of the nodes in the network. Within this paradigm, any node can initiate or complete a transaction, and it can also play an active role in the routing operations. In general, nodes will be users' personal computers, instead of typical elements of the network infrastructure, but they can present heterogeneous characteristics regarding the local configuration, processing power, connection bandwidth, storage capacity, etc.

Fully decentralization often enables the possibility of replicating contents. Replication has important properties, such as the possibility of accessing contents even when some nodes are disconnected. Well-known P2P file sharing systems have evolved towards an increasing decentralization. However, P2P networks are useful not only for relatively simple file sharing systems, in which the main goal is directly exchanging contents with others. Large P2P distribution networks will be more robust against attacks and range to more sophisticated structures which self-organize into ad hoc network topologies with the purpose of sharing resources such as content and CPU cycles (GRID), of maintaining secure and efficient storage, indexing, searching, updating, and retrieving data. In fact, a number of factors, such as the increasing popularity of wireless networks, the opportunities offered by 3G services, and the rapid

proliferation of mobile devices, have stimulated a general trend towards extending P2P characteristics to wireless environments. As a result, the P2P paradigm has begun to migrate to pervasive computing scenarios. So far, the most straightforward approach consists in mounting a mobile P2P (M-P2P) system over a MANET. Nevertheless, nodes in a MANET are constrained by a limited amount of energy, storage, bandwidth and computational power. These factors, among others, limit the type of security measures that can be deployed.

In the following sections, a brief overview of the P2P system model is presented. Firstly, we motivate the need for basic security solutions for P2P and ad hoc networks, and also elaborate on the influence of collaboration-based security services which are in the spotlight.

2.1.1 System Model

The study of security issues in P2P networks becomes more difficult due to the diversity and heterogeneity of existing P2P architectures. With the aim of providing a general specification, we have identified three elements common to every P2P system:

1. **The user community (nodes.)** The user community is characterized by a high node transience, the total ignorance of the node's intentions, and the lack of a centralized authority. These issues have been tackled by different models, protocols and systems which mostly stress cooperation playing an essential role in the network performance.

   The evolution of cooperation is the creation of social profiles of P2P virtual communities [105, 48], which are addressed by recent investigations focused on the establishment of incentives that motivate users to behave well. Typically, at the heart of these proposals operate traditional reputation schemes as polling-based algorithms.

   A solution to encourage resource sharing is to force each peer to contribute before being served. This collaboration is evaluated for computing a user participation level, rewarding the most collaborative peers with, for example, high priorities for their queries or by decreasing the transmission delays of their desired services. The basic assumption of

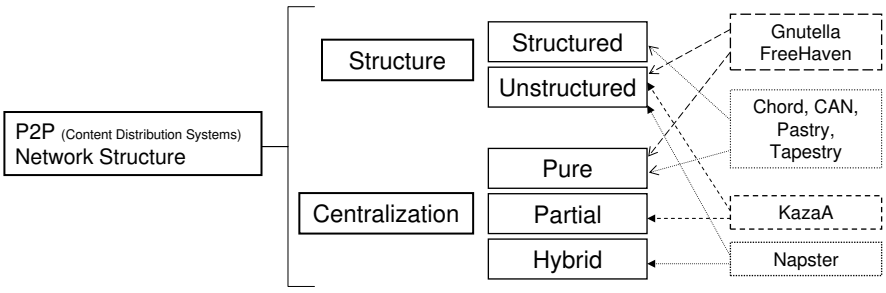


Figure 2.1: Overlay classification.

these models is that peers engage in bilateral evaluations and interaction feedback are done on an agreed scale [115].

2. **The overlay architecture**, which defines the logical structure of the network over the underlying communication layer(s.) Essentially, the overlay network manages the aspects related to node location and message routing.

   Overlay networks can be classified in terms of their degree of centralization and structure. There are three categories concerning the former:

   - **Purely Decentralized Architectures**: There is no central coordination point of the distribution activities. Nodes are referred as *servents* due to their dual nature (SERVers+cliENTS.)
   - **Partially Centralized Architectures**: Special roles are assumed for some nodes called "supernodes", which carry out special tasks mainly aimed at improving the performance of network routing.
   - **Hybrid Decentralized Architectures**: A central server provides the interaction among the nodes, since indexes which support data searches and node identification are centralized, but the data is distributed.

   On the other hand, P2P networks are categorized in terms of their structure as **unstructured** or **structured**. The first category of overlay models was popularized by `Napster` [4], which showed some scalability limits, but reduced the network dependence to a small number of highly connected, easy to attack peers. Peers join the network without any prior knowledge of the topology. Searching mechanisms include brute

force methods, such as flooding the network with propagating queries to locate highly replicated contents. Other well-known unstructured systems are `Gnutella` [3] and `FastTrack/KaZaA` [2].

On the contrary, structured P2P networks provide a mapping between content and location in the form of a distributed routing table. Queries can be efficiently routed to the node having the desired content, and data items can be discovered using the given keys. The overlay network assigns keys to data items and organizes its peers into a graph that maps each data key to a peer. Maintenance of this graph is not easy, especially due to the high transience of nodes.

Figure 2.1 sketches both categories and shows some representative P2P systems for each of them.

3. **The information (content)** stored at nodes and accessible through the services offered by the network. Replication is a common approach to improve performance when distributed systems need to scale in number of users and objects in the system, and/or geographical area. The most commonly used file replication strategy in P2P systems simply makes replicas of objects on the requesting peer, upon a successful query/reply.

2.1.1.1 P2P in Mobile Scenarios

P2P systems for mobile and ad-hoc networks introduce a number of new issues related to naming, discovery, communication and security. In particular, as many mobile devices cannot still store many large files and the network infrastructure have remarkably low capabilities, M-P2P services will differ notably –at least by now– from current P2P applications. Smaller contents, such as games, video clips, photographs, and news, to name a few, are more likely to be distributed. Indeed, there are some future collaboration-based applications in which peers may share data with each other using mobile devices in a P2P fashion. Some works present middleware implementations in Java and SOAP to provide M-P2P services for various wireless/mobile technologies, including WLAN, UMTS, GPRS and GSM [20]. However, the highly dynamic, decentralized and self-organizing nature of MANETs does not fit well with many approaches developed in the P2P world.

Research on MANET technology has addressed the efficiency of data search

and routing, a major problems since the topology continuously and unpredictably changes due to frequent joins and leaves, and since the network performs physical broadcast. Basically, the efficiency of routes often depends on the honest collaboration among nodes, who may serve also as routers, and generally the trustworthiness of routes is kept in order to exchange them among different peers. Obviously, this raises a scalability problem. Some recent works and surveys on MANETs study these fundamental problems (e.g. [43, 70].) On the other hand, it is also necessary to identify vulnerabilities and threats, establish the essential security requirements, and define appropriate mechanisms for M-P2P applications. Moreover, most interactions and operational tasks carried out in a MANET (e.g. forwarding, recommending/accusing, and, according to the service provided, participating) are mainly built around trust, and have to necessarily rely on the cooperation of all the participants. Nevertheless, this fact also incurs a problem; malicious nodes and *whitewashers* may impersonate others and use the spoofed identities to launch false accusations and purify the bad reputation accumulated under their previous identity.

Regarding adversary models, collusion is one of the most dangerous threats, since dishonest peers can act together in efforts to impact honest's decisions. For instance, *Sybil* attack may involve a stream of colluding recommenders boosting the trust of one badly behaved principal. This should be limited by using identity certification. Thus, peers need to be sure that the other party (especially the source, or the recommender) is really who she claims to be. On the contrary, anonymity has been an important consideration in the earlier P2P designs, and especially in mobile environments where anonymity incurs an unacceptable overhead of flooding and uncertainty about the credibility of the potential participant. Recent solutions consider the existence of a self-organized PKI to ensure a correct identification, such as that proposed in [27].

2.1.2 Security Goals

Basically, P2P security schemes proposed so far show two main characteristics: they are inefficient and, in the absence of central authorities, require nodes to cooperate. In fact, their inefficient performance is currently an obstacle to the acceptance and usage of several cryptographic solutions. Particulary, it does not seem easy to deploy security solutions for mobile architectures due, among other reasons, to the inherent limitations of the peers' devices and their po-

tentially very sporadic interaction with other peers. Under these assumptions, early works have tried to establish a decentralized trust management system.

Furthermore, despite the advances in P2P technology, security-related issues have remained systematically unaddressed or, at best, handled without a global perspective. Classic approaches have concentrated on specific points, such as mitigating attacks against three main system properties: availability, integrity, and anonymity.

Availability is measured by how often object requests are successfully served and, in particular, mapping two factors: the number of peers (average node availability) and the number of object replicas (replica storage size.) An important problem is how to deal with an overestimated number of copies that could cause serious security conflicts, like DoS, self replication (false information distribution), man-in-the-middle and pseudo-spoofing attacks. Some algorithms increase the availability of all shared files toward a common target availability while allowing peers to act completely autonomously using only a small amount of loosely synchronized global data. In `Gnutella`, a decentralized P2P infrastructure has been implemented to hold self-replication (distribution of false information), man-in-the-middle, pseudospoofing, ID stealth and shilling attacks.

It is also necessary to balance the total network anonymity (see references [83, 116]) and the need of preventing network abuse, to assure content's high quality. Common components in such algorithms are: the servent, generally identified by a public key digest, reputation, the resource (i.e. a content digest) reputation, and a simple binary algorithm for voting. In the same way, `Freenet`, like `Chord`, does not assign responsibility for data to specific peers, and its lookups take the form of searches for cached copies. But, in `Freenet`, files are identified by content-hash keys, which gives every file a pseudo-unique data file key, and by secured signed-subspace keys to ensure that only one content owner can write to a file and anyone can read it. Moreover, the integrity of information in a P2P system may be attacked through the introduction of degraded-quality content or by misrepresenting the identity of the content (e.g. falsely labeling.)

On the other hand, emerging security problems are being analyzed, such as *attrition* attacks, which perform a especial type of network flooding (see e.g. `LOCKSS`, *Lots Of Copies Keep Staff Safe*, system [81].) Furthermore, when a

malicious node claims that his storage is full or that he contains a particular file, an attack against a *fair* sharing is running, and its detection is being investigated.

Moreover, research efforts have also focused on the study of DoS attacks and the abuse of multiple identities (*Sybil* attack) [44]. Other problems have been recently identified, such as those associated to the transience of peers (*churn*) or how to combat the selfish behavior exhibited by nodes that do not share their resources (*free-riding*) [51].

2.1.2.1 Security Requirements

Authentication and access control are fundamental to secure the system from unauthorized actions. In this regard, several works have already shown how to provide authentication, confidentiality, integrity and non-repudiation services in ad-hoc domains, based on identity, reputation and trust, proximity, and also public key cryptography [61]. Traditional cryptographic primitives can be actually used, perhaps with some restrictions, in mobile architectures. However, the low capabilities of wireless nodes have reinforced the use of trust-based solutions rather than the inclusion of regular cryptographic schemes. The main concern with these approaches is that, in most of them, nodes are trusted by default, and therefore they are susceptible to attacks. Concretely, a correct node identification is critical, and the lack of control on it could yield to vulnerabilities in the main processes. Node identification (and its relationship with anonymity) is an intense research area due to the potential risk of performing traffic analysis attacks and the traceability of communications among nodes. An example is the problem during *churn*, which involves a large number of potentially malicious peers in the P2P system to certify the peers identities. The simplest (but unrealistic) scheme for assigning an ID to each node is to have a centralized authority providing cryptographic certificates, which is only consulted when new nodes join.

The existence of an underlying mechanism for providing keys is also a problematic issue. Approaches based on the creation and distribution of a common key or on the inference of a strong key from a weak shared password, have some problems with scalability and mobility [43]. Thus, an encrypted channel can be created if nodes share (or negotiate) a key. If the common key can be stored and shared by devices, the problem would have an easy

solution. Otherwise, a pair-wise shared key has to be established on the fly, without requiring the use of any on-line key distribution center. Current solutions envisage probabilistic key sharing and threshold secret sharing. Other solutions suggest the combination of centralization and key agreement techniques [127]. In fact, the key agreement protocol is only executed between a subset of nodes, which play a connecting role within the community. Then, the main idea is to cluster nodes in service-oriented communities, generally according to their physical position [123]. Apart from this, problems such as *Sybil* and *pseudo-spoofing* attacks –extensively studied in the context of P2P networks– appear whenever authentication protocols use opaque identifiers in favor of anonymity [135]. For this reason, most mobile solutions require the existence of some kind of control, such as a CA or a key sharing mechanism.

The establishment of a key management service using a hierarchy of CAs seems unsuitable for the moment, since a naive delegation and replication of the CA's responsibilities makes the service more vulnerable. To solve this problem, some works suggest the use of trust as a principal building block to address public key management. Particularly, schemes similar to Pretty Good Privacy (PGP [136]) are fully distributed and self-organized [27]. Nodes can generate their keys, and their distribution can be done without relying on external directories, such as in *Friend-to-Friend* protocols (F2F.) Basically, this kind of approaches binds an e-mail address to a public key, that is sent to a user by e-mail, and then a a ring of trust is created. Key authentication depends on the execution of an agreed checking on that key's fingerprint. This authentication does not exist in the majority of P2P systems, though some studies show that it may be possible to use some form of cryptographic puzzles, which avoid attackers with large computational resources to get a huge range of node IDs.

Nevertheless, this approach presents some inconveniences, e.g. those derived from high transient communities with a high number of sporadic participants. On the other hand, proposals based on threshold cryptography present a completely different approach –distributing the CA's functionality over selected nodes [134]. Briefly, this scheme is secure if the adversary cannot compromise more than k out of the n members in any period of time. These schemes have also a number of drawbacks, mainly related to their communication overhead (any client needs to contact at least k different nodes to get

a certificate.) Furthermore, the work presented in [126] relies on threshold signatures to protect both routing and data forwarding.

On the contrary, trust negotiation solves the problems associated with classical authentication and authorization schemes by allowing individuals to safely agree participation admissions for resources and services. Other approaches use a similar idea but using mobile agents [75].

2.1.3 Challenges on Cooperation

Besides the proposals and constraints mentioned above (mostly provided by ad hoc computing), cooperation-based services have been successfully addressed in order to create opportunities for new approaches which involve interaction among nodes, and to support different constraints such as those listed below:

1. Support for decentralization, fault-tolerant and scalability.

   Since centralized control on ad hoc networks has very poor performance for their single point of failure, research advances have had either to adapt classic schemes or to explore novel mechanisms, most in the way of using informal collaborations to solve complex problems.

2. Support for trust-related tasks.

   Almost all reputation systems have been designed for motivating peers to positively contribute to the network, just like for punishing adversaries who try to disrupt the system. In fact, users of a current P2P systems actually appreciate the notion of reputation as an incentive in view of future interactions [30]. There has also been a trend in the research community towards the use of trust models to address some security concerns, particularly for ad hoc and self-organized environments. However, trust-based solutions generally rely on the exchange of reputation feedback (submit ratings on performance of their mutual transaction) among community nodes in order to globally/locally evaluate a certain node's trust value. These interactions with many other participants necessary require collaboration.

3. Support for incentive and fairness models.

   Both concepts begin to be taken into account to control aggressive behavior ("antisocial") between peer interactions, and also both look to

enforce fair resource sharing in which, at best, users are willing to cooperate even without being explicitly rewarded by the system for their contributions. Some studies have pointed out the benefits of employing incentive-based mechanisms as a means to foster cooperation among peers. Even though these incentives rarely can substitute the strength provided by a cryptographic primitive, they can be still very useful as a trade-off for non-critical applications. As an example, node participation in a network has been recently addressed in [38] by adopting a game theoretical approach. From the results presented, the authors conclude that even if nodes perceive a cost in sharing their resources, this may induce node participation.

2.2 Public Key Primitives for P2P and MANETs

For readability and completeness, we first discuss some of the most representative cryptographic primitives to provide security through cooperation.

The provision of security services in a fully distributed system (e.g. a pure P2P network) becomes quite difficult if public key cryptography cannot be used –at least, in the way it is employed in classic distributed systems. Nearly all the solutions proposed so far are one way or another based on the idea of replacing the whole PKI by a collaboration scheme. Put simply, a subgroup of peers must cooperate to perform tasks such as generating or verifying a digital signature. Threshold cryptography has been repeatedly pointed out as an appropriate technique to achieve these purposes, even though its computational cost can disable its application for restricted devices (e.g. mobile scenarios) [33]. We will overview basic operational characteristics of this technique among others.

2.2.1 Overview of Public Key Cryptography

In a public key (or asymmetric) cryptosystem, users have a pair of cryptographic keys, a public key and a private key, related mathematically, but the private key cannot be practically derived from the public key. The private key is kept secret, while the public key may be widely distributed. Encryption with public key cryptography has some interesting properties. A a message

encrypted with a recipient's public key cannot be decrypted by anyone except the recipient possessing the corresponding private key. This is used to ensure confidentiality. Conversely, in a secret key or symmetric cryptosystem both entities must agree to use a single secret key for both encryption and decryption. On the other hand, in order to ensure authenticity and integrity, a message can be digitally signed with a sender's private key. The digital signature can be verified by anyone who has access to the signer's public key, thereby proving that the sender signed it and that the content has not been modified [113].

The critical problem in public key cryptography is how to become convinced that a certain public key is authentic, and has not been tampered with or replaced by a malicious third party. To face this problem is commonly applied a PKI, in which one or more (configured in hierarchy) third parties (CAs) certify the ownership of key pairs by means of binding public keys with respective user identities (along with several other attributes) in public key certificates.

However, while TTPs are acceptable in the client - server computing paradigm, they are not suitable to P2P models for a number of reasons. The viability of a PKI between P2P communicating parties is limited by practical problems such as uncertain certificate revocation, CA conditions for certificate issuance and reliance, variability of regulations and evidentiary laws by jurisdiction, and trust. In fact, another approach is the use of trust metrics, also known as "web of trust", used by PGP. We further elaborate on this primitive below.

Definition 2.2.1 (Public-Key Cryptosystem [89])**.** Generally, a public-key encryption scheme, (G, E, D), consist in three main phases:

1. G is a key generator that on input n, the security parameter, outputs a pair (e, d) where e is the public-key written in a public file and d is the private key.

2. E is an encryption mechanism that given a message m and the public key e produces a ciphertext c.

3. Finally, D is the decryption mechanism that on input e the public key and d the private key and a ciphertext e produces a plaintext message m.

As a practical application, let B be a receiver who publishes his public encryption key e_B (say, in some public file), while keeping secret the private

decryption key d_B. Then, whoever wants to send a plaintext message m secretly to B, will pick e_B from the public file, encrypts the plaintext and sends him the resulting ciphertext $E_{e_B}(m)$. This way, only B can decrypt the message by applying the decryption key $D_{d_B}(E_{e_B}(m)) = M$.

Specifically, if a public key can be securely associated to a party, a typical scenario is to ensure the integrity of a content generated by her, as follows. First, the source:

1. Computes a hash value from the content.
2. Encrypts the hash value with her private key, obtaining a digital signature.
3. The signature is enclosed together with the digital certificate which contains the user's public key.
4. A CA validates the sender's digital signature.

Then, the receiver:

1. Computes a hash value from the received content.
2. Decrypts the digital signature enclosed by using the public key certificate, thus generating a second hash value.
3. Compares both hash values to confirm the non-alteration of the content.

As shown, distributed online CA-based systems count on TTPs to generate public keys and issue digital certificates. Research works on ad hoc computing have had to adapt such schemes to such scenarios in absence of CAs [126]. For further details on introduction to cryptography, the reader is referred to [104].

2.2.2 Threshold Cryptography

The goal of secret sharing schemes and threshold cryptography is to distribute the provision of basic security services/tasks (e.g. the authority to sign a file), in fully distributed environments without the existence of any fixed infrastructure, and in presence of malicious nodes [41]. Such techniques offer better fault tolerance and resilience with respect to crashes of some of the system

components since even if some nodes are unavailable, others can still perform the task. Indeed, threshold cryptography exploits cooperation in a way that a coalition of cooperating parties can jointly perform a critical action.

Definition 2.2.2 ((t, n)–Threshold Signature Scheme [33])**.** At any time, a node i may request a signature of message m from other parties in the community, by sending her identity, parameters t, n and the message m to be signed; at this point those parties need to generate a t out of n threshold signature for message m. The scheme is formally structured in the following two main phases: a key-generation phase and a signature phase. Let $(\mathcal{P}_{kg}, \mathcal{P}_{sgn})$ be an execution of threshold signature scheme, where:

1. A randomized distributed threshold key generation protocol, denoted as $\mathcal{P}_{kg}$, is run by the players $P_1 \dots P_n$. The output of this phase is a string sk_i, pk_i for each player, who use as input the common parameters and a different random string. Each party will output its signature share.

2. The execution of t-out-of-n signature protocol, denoted as $\mathcal{P}_{sgn}$, produces a pair (sig, out) for the node P_i, where sig is supposed to be a (t, n)-threshold signature of m, and $out \in \{yes, no\}$ denotes whether sig is a valid signature of m. P_i uses as input the common parameters, t (not necessarily predetermined) parties from the community and a message m of her choice, while each party P_i uses as inputs the common parameters and the corresponding sk_i.

In a MANET, the scheme should satisfy the following properties: correctness, unforgeability and robustness.

Concretely, n parties share the ability of performing a cryptographic operation (e.g. creating a digital signature), and any subset of at least t out of n parties can jointly perform the operation. On the other hand, any $t - 1$ (or less) parties cannot perform the operation. Moreover, any $t < \frac{n}{2}$ parties cannot prevent the remaining honest parties to perform the task.

We can classify threshold schemes in two types: those that need a trusted centralized authority or dealer, and those that do not required this kind of TTP. Verifiable secret sharing and common RSA or DSS threshold signature schemes belong to the former, while others provide joint secret sharing without a dealer. With the aim of not violating the nature of the P2P and ad hoc network, recent approaches let peers themselves to locally handle main procedures without any dealer [90].

The establishment of the threshold t is usually a main matter. With a fixed threshold policy, an adversary may try to manage and permanently compromise t group members in order to expose the secret. In such cases, the reduction of value t involves associated problems such as when a large number of members leave the group, resulting in a new group size, probably less than t. Thus, threshold schemes need to securely and reliably determine the number of current participants on each task.

Several approaches have addressed the provision of fault tolerant authentication, most in the way of protocols relying on threshold cryptography that use key sharing and agreement techniques. Practically they all share a common idea: to use a quorum of participants to create, for example, a digitally signed public key certificate. Particularly, the work presented in [79] provides an ubiquitous authentication service by following a certificate-based approach. In this case, no single node has the power of providing full certification services (e.g. certificate issuing, renewal and revocation), since none of them holds the complete certificate signing key. Instead, multiple nodes collaborate in a network locality, by means of establishing a temporary trust relationship via a localized trust model. In general, a node is trusted if any fraction k (a system-wide parameter) of trusted nodes (typically the neighboring nodes of the entity) claims so. The required k sets the global acceptance criteria since a locally trusted entity is globally accepted anywhere. The latter incurs a problematic requirement concerning to node identification reliability.

2.2.3 Byzantine Agreement

The Byzantine Generals Problem basically consists in deciding a common battle plan; a group of distributed Byzantine generals, camped around an enemy city, must agree upon "attack" or "retreat" the enemy. Each base communicates among each others sending conflict information, with the vulnerability of traitors and enemies who try to prevent the loyal generals from reaching a plan.

Definition 2.2.3 (Byzantine Generals Problem [71])**.** Using oral messages, this problem is solvable if and only if more than $\frac{2}{3}$ of the generals are loyal, or what it is the same, no solution with fewer than $3m+1$ generals can cope with m traitors. With signed messages, the problem is solvable for any number of

general and possible traitors.

Consequently, if all the participants are honest, they will reach consensus on validity.

Regarding computation, we can model two types of processes/participants: honest and faulty. Obviously, a participant is faulty if it does not follow, accidentally or maliciously, the specified algorithm. There is no solution if the upper bound on the number of faults exceeds one third of the nodes. Note that if the actual number of faulty processes is larger than the upper bound, then the algorithm may fail to reach Byzantine Agreement without alerting any correct process to that fact. Therefore, the immediate sender of any message must be identified, and also there must be a low threshold which prevents potential collusion of faulty nodes from introducing fake information.

Several works have applied that idea in order to provide Byzantine Fault Tolerance [28], while others apply the scheme to support high level security services based on consensus. Pathak and Iftode [97] apply the same idea in order to provide public key authentication in ad hoc networks. This work postulates that a correct authentication depends on an honest majority of a particular subgroup of the peers' community, labeled "trusted group". However, in P2P systems an authenticated peer could create multiple fake identities and act maliciously in the future (Sybil attack [44].) For this reason, the classification of the rest of the community maintained by each node (see Fig. 2.2) has to be proactive and should be periodically flushed. A periodic pruning of the trusted group will ensure honest majority. Thus, honest members from trusted groups are used to provide a functionality similar to that of the PKI through a consensus procedure.

The authentication protocol consists in the four phases that are briefly discussed below. In Fig. 2.3, a node A belongs to "others" group. Node B authenticates A using its trusted peers, and one of them turns malicious (black node) that tries to prevent authentication of A, C... Interested readers can find further details in [97].

1. **Admission request**. The protocol begins when B (Bob) run into a newly discovered peer A (Alice), which claims to be the owner of an unauthenticated public key K_A. Then, B asks to a subgroup of his trusted group for helping him in verifying the authenticity of K_A. Finally, B sends K_A to those trusted peers that agree.

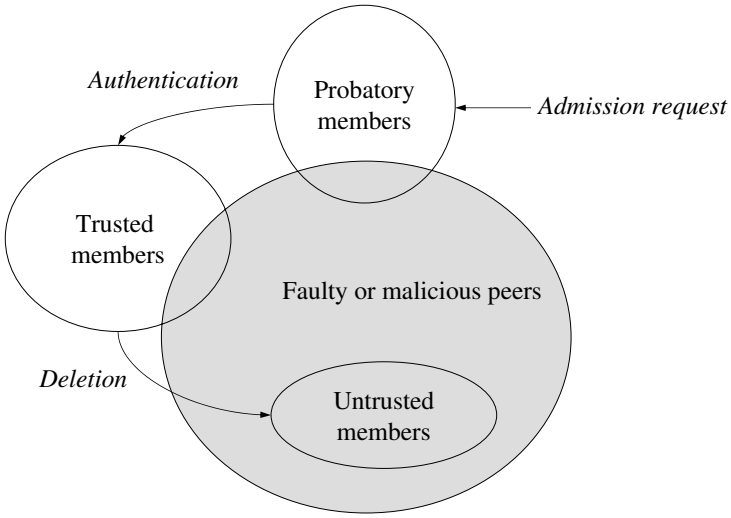


Figure 2.2: Community structure according to the authentication state of each node.

2. **Challenge response**. Each notified peer challenges Alice by sending a random nonce encrypted with Alice's supposed public key. Alice is able to return each received nonce if and only if she holds the corresponding private key, K_A^{-1}. Each challenger checks if the received response is correct, thus obtaining a proof of possession of K_A.

3. **Distributed authentication**. Each peer helping Alice sends her proof of possession to Bob. If all peers are honest, then there will be a consensus, so Bob gets the authentication result: K_A belongs to Alice or not. However, some of the peers summoned by Bob could be malicious or faulty, which may result in Bob receiving different opinions about the authenticity of K_A. In this case, Bob must initiate the Byzantine agreement phase.

4. **Byzantine agreement**. First, Bob verifies if Alice is malicious by sending to her a proof request message. Alice must respond with all challenge messages received, and the respective responses sent by her. If A is honest, she can provide a correct response and also demonstrate her good behavior by sending to B the challenges she received and the corresponding responses. If A cannot be proved to be malicious, then some of the

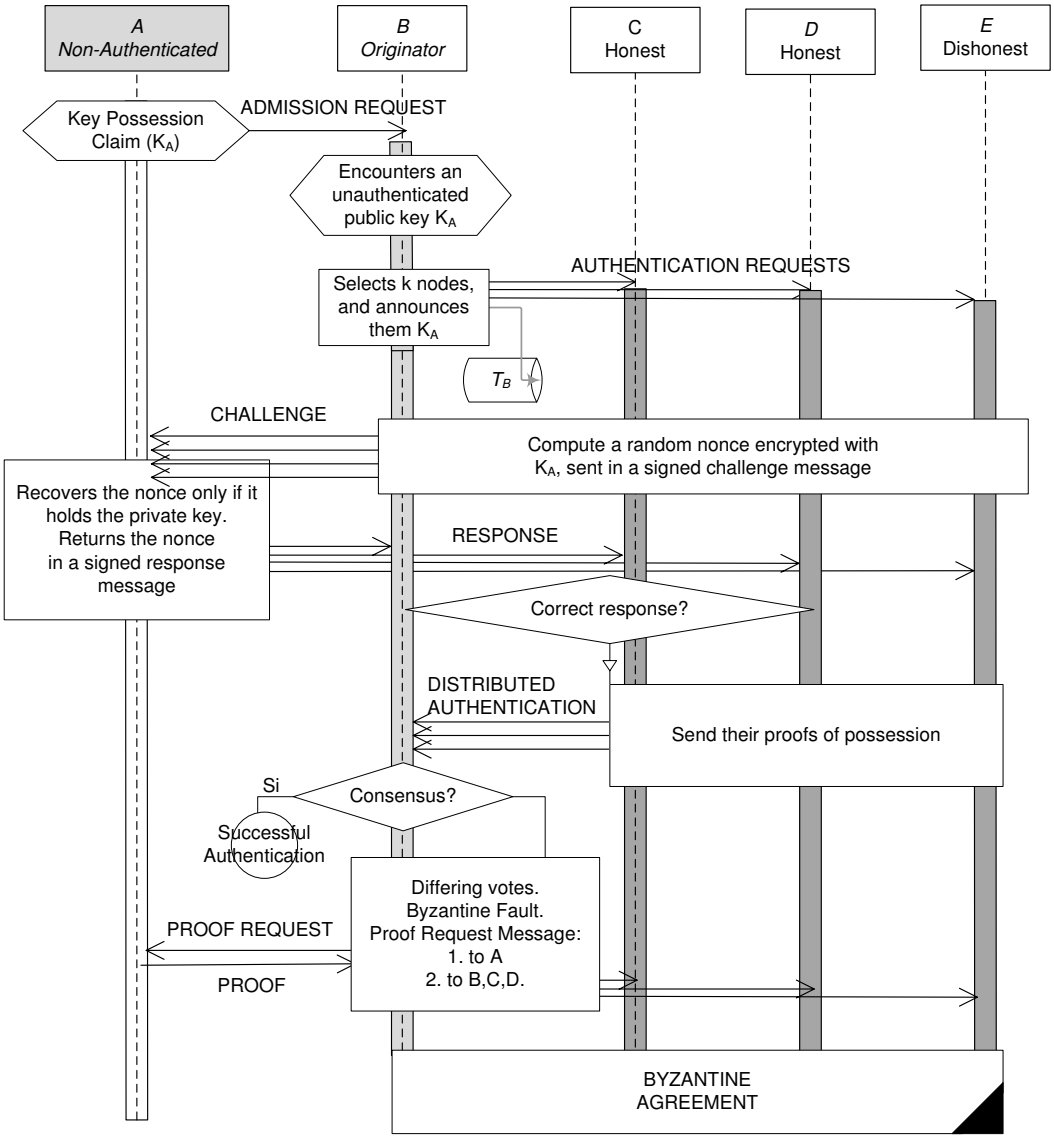


Figure 2.3: Phases of the public key authentication protocol.

peers must be. At this stage, B announces a Byzantine fault to the group. Each group member sends an agreement message to others. At the end of this phase, the honest peers will be able to recognize malicious peers causing the split in authentication votes.

Successful authentication moves a peer to B's trusted group, while encountered malicious peers are moved to the untrusted group. Peers can be also deleted from trusted groups due to inactivity and periodic pruning of the

group.

The fundamental limit of this scheme is the following. Assume that N is the number of peers in the community, t the number of malicious or faulty peers, and ϕ a fraction of N, denoting that ϕN peers may not be reached during the protocol execution and another ϕN peers exhibit faulty behavior because the path between the source and them suffers from a man-in-the-middle attack. It can be shown that the community has honest majority if $t < \frac{1-6\phi}{3}N$. As the value of ϕ does not have influence on a random selection, we can consider that a group has honest majority with $3t + 1$ peers [97].

Summarizing, the previous protocol provides us with public key cryptography without relying on CAs in MANETs. This can be viewed as an essential building block upon which more complex security schemes can be developed. As an example, authors mention its application within an e-mail authentication system named `SAM` (Self Authenticating Mail.)

2.2.4 Web-of-Trust, and Reputation Metrics

The idea of creating trustworthy communities has attracted considerable attention in the last years [25, 95]. Since it is not realistic to assume such a reliance exclusively on a a hierarchy of CAs in a P2P network, and especially if it is the case of a MANET, some works use web-of-trust models (like PGP) to provide, among others, authentication mechanisms [93].

This technique has focused largely on public key authentication, digital signatures, and certificates, and creates a decentralized fault-tolerant web of confidence for all public keys. It does not require the existence of a single point of trust, but allows the use of email digital signatures for self-publication of public key information. The key idea is simple; trust decisions are in the hands of individual users. This way, it is relatively easy to establish one's own trust community. For example, the model proposed in [34] manages utility functions for individual users as a function of the quality of service. These utility functions are mainly based on the amount of shared contents and their quality for estimating the node's aptitude. The drawback is that these kind of mechanisms are easily disrupted by the actions of dishonest nodes. Each peer has a credit value during some time interval and sends it when peers interested on him require this information. The interested peer compares his value with the value of other peers interested on the same peer. In order to perform an

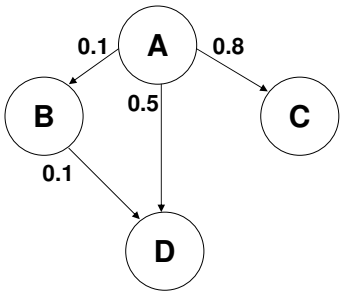


Figure 2.4: Example of asymmetry of trust evaluation.

integrity check based on the information provided by the interested peers, we need to let each peer to disseminate his information to the interested peers during operation.

Nevertheless, how one authenticates others' public key? The response is a digest value (hash result or fingerprint.) A key is validated by verifying the key's fingerprint with the key's owner. By signing the key one certifies it as a valid key [16].

A few security protocols proposed for P2P environments so far are based on cryptographic schemes as diverse as threshold cryptography, distributed consensus or Byzantine agreement. Nevertheless, most proposals require an underlying trust model. In fact, trust and reputation models have been recently suggested as an effective security mechanism for open environments, showing that rating nodes is an effective approach in distributed environments to improve security, to support decision-making, and to promote node collaboration [22].

On the other hand, the trust metric implemented in PGP is simple and can lead to counter intuitive decision being made [49]. Concerning trust metrics, different trust models have been developed and their properties have been extensively studied. Because of the dynamic nature of MANETs, trust computation/evaluation may be uncertain and incomplete. Some works give intuitive requirements that any trust algorithm should have under that framework as well. Basically, users need not have direct experience with every other user in the mobile network in order to compute an opinion about them [120]. Instead, they can base their opinion on the indirect recommendations provided by intermediate nodes (and probably neighbors.) Obviously, trust schemes

in MANETs rely on the cooperation among nodes (or on secure automated agents), and also be robust in the presence of dishonest actions. As many services in MANETs, such a problem is addressed as a generalized shortest path problem on a weighted directed graph. As an example, let the graph plotted in Fig. 2.4 be a simple trust network (temporary or not) in which directed edges contain explicitly trust values specified by the evaluator. Node A trusts node B as 0.1 ($\in [0, 1]$), node C as 0.8, and node D as 0.5, while B trusts D as 0.1. In turn, this does not mean that, for example, B trusts A as the same value; arrows represent the direction. C knows no one except for A, so a trust metric can be used to infer the trust values between two nodes who are not directly connected, e.g. how much C could trust D. Consequently, there are two different interaction paths from C to D: crossing B or not. A simple metric may be the sum of the trust values obtained directly from own experiences (D has none) together with the recommendation of others, with the aim of choosing the maximum. In the example, the shortest path is also the most trustworthy, by chance, and usually the recommended node will store the recommended trust value (and the recommender) to compute his own trust strategy. These topics are out of the scope of the present chapter, and therefore we do not provide more details either. Note that we have already mentioned some drawbacks, e.g. the lack of robustness, of adopting security models which rely, to what extent, on trust in mobile environments.

In similar way, reputation systems counter corrupt content attacks by enabling users to rate the validity of content and those who provide it. To ensure that all copies of the same content share the same reputation, content may be identified by its hash. This enables reputations to scale far beyond trust in the user and allows widely duplicated corrupt files to be recalled quickly [111]. Furthermore, it is required ensuring that an attacker cannot modify or delete its client's reputation information, so designers must distribute this information among the other clients using protocols that prevent tampering. Since attackers can delete clients and reinstall new ones, a reputation system should also maintain information for the machines on which clients run. This does not fit well with anonymity. A malicious node might give erroneous responses to a request at two levels: network, i.e. returning false routes, and at the application level, returning false content. Finally, content authentication is commonly uncertain and current research efforts have adopted popularity-based ranking

systems to help users discover desired contents.

2.3 P2P Security Services

As the bandwidth available per user increases and Internet becomes anywhere accessible, the lack of scalability and fault tolerance given by the classic client–server paradigm motivates the exploration of fully distributed schemes, which often require the cooperation among the network participants. As single nodes become essential for system operation, their responsibility, costs, and involvement rise, and therefore their vulnerability to malicious attacks and unreliability also increases. Particularly in P2P file-sharing applications, distributing control means relying on the cooperation of participants, and it is a significant challenge to design mechanisms that affect the overall behavior in a population of rational, and therefore selfish nodes, and also to incentive contributing to the public good by the sharing of files. The aim of this section is to overview the most representative proposals which use cooperation as a main building block in order to provide security services.

2.3.1 Secure Routing in Self-organizing Overlay Networks

In P2P computing, the general function of an overlay network is to promote and support adaptive self-organization and maintenance of a (un)structured network formed of logical relationships among components. The basic principles to provide self-organization are to dynamically discover potential communication nodes and available services (generally starting from a few basic mechanisms such as broadcasting), and efficiently navigate with independence of the physical network [128].

Most approaches tend to require a priori assumptions on the network configuration that would limit its self-organizing nature and therefore its degree of adaptivity and robustness. In an ad hoc wireless network, no pre-deployed infrastructure is available for routing packets, instead routing relies on intermediary peers. In particular, a collection of mobile nodes join together and create a network by agreeing to route messages for each other. All networking functions must be then performed by the nodes themselves in a self-organizing

way. This operating principle involves cooperation among nodes as an essential requirement [117]. In consequence, instead of considering all peers trusted by default or caring the trustworthiness of the immediate neighbors, ad hoc routing proposals should attach more importance to security requirements [84, 98, 59].

P2P architectures use data encryption too to prevent adversaries from observing or modifying network-level communication between legitimate nodes. We must assume that the attacker is able to use these properties for linking messages and, correspondingly, the pseudonyms used with them. So far, `onion routing` provides application independent, real-time, and bi-directional anonymous connections (not anonymous communications) that are resistant to both eavesdropping and traffic analysis. Some existing systems which tackle anonymity issues are: `Crowds` [100], `Hordes` [73], `Tarzan` [56], `Freedom` and `FreeHaven` [42]. On the other hand, the idea of using pseudonyms rises when solving attacks against anonymity, and also the use of *blacklisting*. A digital pseudonym could be conceived as a public key with the aim of testing digital signatures. Pseudonyms attacks involve several scenarios, such as *cheating* and *Sybil* attacks (a malicious abuse of different pseudonyms), and *free-riding* where non-cooperative users benefit from others' resources [45, 51].

2.3.1.1 Stimulating Wireless Forwarding

The problem of stimulating cooperation in MANETs has been extensively addressed, since networking nodes are not naturally motivated to cooperate towards a common goal, but for saving own resources (i.e. memory, CPU cycles, battery power, etc.) This is especially critical in wireless communications since any node can be used as a relay to forward packets. The work presented by in [26] is centered in stimulating the packet forwarding function, assuming that every mobile node has a tamper resistant hardware module. Basically, this security hardware has a counter which is protected from illegitimate manipulation, and also establishes cryptographic associations (i.e. link-by-link encryption of the packets using a session key) with neighbors. The counter is decreased when the node wants to send its own packets, and increased when the node is an intermediary and forwards packets for the benefit of other nodes (see Fig. 2.5.) However, the value of the counter must remain positive whenever the node wants to send own packets, just as the energy of the node.

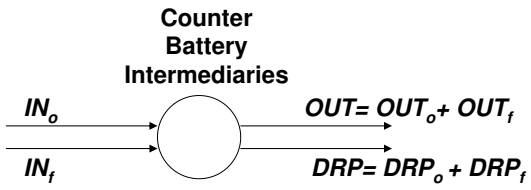


Figure 2.5: Model of a node [26]: Two incoming and two outgoing/droping flows of (***o****wn* and ***f****orwarding*) packets.

Furthermore, in order to send own packets, the estimated number of intermediate nodes that are needed to reach a destination will reduce the counter. The key idea is to equilibrate the number of packets (own as well as forwarding) that the node can send using its remaining energy, and also maximizing its own benefit, i.e. the number of own packets sent.

On the other hand, a critical situation occurs when the arrival rate of forwarding packets is too low, and then the node cannot earn enough to send all of its own packets, even if it forwards all packets received for forwarding [26]. On this matter, experiments show that there is usually a small fluctuation in the ratio between the number of forwarding packets and the number of own packets, due to the random manner in which the packets arrive. A collection of forwarding rules are proposed to assist decisions about when forwarding or dropping packets received for forwarding, while own packets that cannot be sent immediately (due to the low value of the counter) are dropped. For example, when the node reaches optimal counter to drain its battery out by sending only its own packets, it is not necessary to forward more packets. Results from simulations show that the most cooperative the rule is, the best the performance it achieves. In summary, it can be seen experimentally that the network can tolerate less cooperative nodes quite well, even though the throughput of the network decreases as the fraction of less cooperative nodes increases.

2.3.1.2 Authenticated vs. Trusted Routing

Authentication in routing presents several challenges due to the fact that each user brings to the network his own mobile unit, without relying on a centralized policy or control such as those present in a traditional network. In fact, many different types of attacks have been identified against the most impor-

tant routing protocols for MANETs, e.g. those which are under consideration by the IETF for standardization: the Ad hoc On-demand Distance Vector routing protocol (AODV) and the Dynamic Source Routing protocol (DSR.) Concretely, some analysis focus on DoS attacks based on interception and noncooperation, derived from redirection of network traffic, and by altering control message fields or by forwarding routing messages with falsified values. Modification attacks combined with impersonation and spoofing are also studied. Possible solutions of these attacks range from securing routing through pre-determined cryptographic certificates that guarantee end-to-end authentication [110], to applying mechanisms based on watchdogs and distributed reputation systems [84].

An example of such mechanisms is the CONFIDANT protocol [24], which aims at detecting and isolating misbehaving nodes in a DSR-based network, thus making unattractive not to cooperate. The watchdog is a neighborhood monitoring function which observes the behavior of neighbors and identifies misbehaving nodes by promiscuously listening to communications of nodes in the same transmission range. A reputation system maintains global values for each node according to the information collected.

Nevertheless, these reputation-based solutions present some limitations. First, the ephemeral nature of wireless connections and the presence of collisions may cause monitoring system to fail. Second, there are severe drawbacks in terms of traffic overhead and the potential spreading of wrong accusations. Finally, they eventually can reach a situation where all the traffic is deviated to well-behaving nodes, with the result of overloading them and the links between them. On the other hand, proactive protocols (e.g. DSDV, OLSR [31]) are subject to misbehavior caused by selfish nodes that do not cooperate in the propagation of routing information through the network. We will explore and identify the problems potentially affected by self-interested behavior in a subsection bellow.

2.3.2 Admission Control in P2P and MANETs

Another main factor attracting research attention to P2P security is to enforce appropriate access control policies, which limit the activity of legitimate users while serve as an obstacle to dishonest requesters. In such a scenario, content owners may determine if the requester attempting to download a particular

content is actually authorized to access it. On the other hand, ensuring that each one of these new potential providers will behave accordingly to the owner's access policy gets more complicated (and definitely hard since a global security infrastructure is unavailable.)

Similarly, the increasing content availability gives rise to an attracting interest in Digital Rights Management (DRM) and, therefore, in providing "controlled" P2P computing services. In fact, several current research proposals work on facilitating the emergence of new models that maintain some control over the file-sharing process. So far, only few works have dealt with admission control (or any other form of authorization) in P2P and MANETs [121, 70, 38]. A malicious node may coax a man-in-the-middle-based situation where he can send an unsolicited response to a query, or can attempt to forge a message to a requester with incorrect results. The best defense against this would be to employ standard authentication techniques, such as digital signatures or message authentication codes (MACs.) However, MACs require shared keys.

In most systems, the absence of authentication is solved by distributing appropriate keys into groups of authorized users for granting access to the shared content. `Oceanstore` is an example of system in which each owner assigns contents an access control list using digital certificates. Every content alteration is verified against the access control list (ACL), ignoring non-authorized updates. A different approach is presented by Pathak and Iftode [97], in a Byzantine fault-tolerance public key authentication protocol. The lack of TTP motivate users to classify nodes into three categories: "trusted", "untrusted" or "others", after reaching an agreement through proofs of possession of a legitimate copy of an untrusted node's public key. The other side of the spectrum is represented by the applications where users are known (or "friends") and share their friends with new friends; we talk over F2F protocols already discussed [47].

Other approaches provide admission to secure peer-groups based on threshold signatures, such as the scheme proposed in [90]. Newly joining members must acquire a share of the group secret for themselves in a distributed manner. A newly joining member will receive t partial secret shares together with a membership certificate from each of the t members, who should admit her to the group. The possession of a partial share enables him to participate in future voting procedures, in order to admit new members. Authors examined and compared two threshold signature schemes: one based on RSA signatures

and the other based on DSA signatures. We refer the interested reader to the original work for further details.

In summary, the decentralized and anonymous characteristics of common P2P file sharing systems enforce an inherent "open-access" policy which does not provide any mechanism to support access control decisions. This lack is critical in collaborative applications, especially in team-working and virtual workplaces where peers' privileges may not be the same, as well as the sensibility of each resource in the system.

2.3.3 Content Authentication in P2P Networks

The possibility of replicating the same content among different nodes, and download a specific content at any moment, is an attractive distinctive feature in P2P file sharing scenarios. In the vast majority of current systems, these tasks are not performed in a proactive way, but they are simply the result of the existence of a search and location mechanism. Once a user gets the file, it is usual that a local copy will remain in the node, in such a way that future queries will identify the node as one of the various locations from which the content can be obtained.

This fact presents some interesting properties. Faced with different locations of the same content, an application can grant priority to that which offers a less expensive path (e.g. in terms of bandwidth and/or number of network hops.) To some extent, replication also guarantees some sort of fault tolerance, since information can be available even if some parts of the network are temporarily disconnected.

In a collaborative working environment, the previous features are highly desirable [130]. However, it is unrealistic to assume that every integrating node will exhibit a honest behavior, even if they have always behaved correctly in the past. Once that a content is replicated through different locations, the originator loses control over it. A malicious party can modify the replica according to several purposes:

- To claim ownership over the content.
- To insert malicious software into a highly demanded content. Not in vain, P2P networks are becoming an important medium to propagate recently developed viruses, spyware, etc.

- To boycott the system by offering fake contents. Eventually, this can generate distrust and bad reputation in the community.

Secure content distribution protocols are highly desired for these environments. The scheme presented in [94] shows how digital certificates can be first used for content authentication, and also be extended to provide authorization capabilities, much in the way a X.509 public-key certificate can be used as an attribute certificate. Briefly, the main objective of this content authentication protocol is to maintain content integrity, ensuring its authenticity and avoiding non-authorized content alterations. This proposal is similar to PGP in the sense that both the content and the corresponding authorization certificates are issued by the users.

However, as opposed to PGP, the proposal does not rely on certificate directories for the distribution of certificates. Instead, in this model, certificates are stored and distributed directly by the users. This is achieved through the collaboration of a fraction of peers in the system. Previously to content distribution, the legitime owner of a given content generates a content certificate. For this, the owner first selects a subgroup of signing nodes from her group of previously contacted peers. The certificate contains a number of fields: The identity of the originator (which ultimately establishes who has generated the content and is its legitime owner); the identification and hash of the content (ensuring its integrity); and an ordered list of signers, who in turn provide a joint signature certifying that the content is authentic and belongs to its owner. After downloading a replica of a content (and its associated certificate), the user is encouraged to verify its authenticity.

This service can be extended to provide authorization capabilities by increasing the complexity of the certificates (and, therefore, their manipulation.) Attributes encoding permissions can be easily inserted into authorization certificates, which are discretionally issued by the owner.

2.3.4 Combating Selfish Behavior in P2P Networks

A number of works have focused on quantifying the optimal cost/benefit tradeoff that would lead nodes to share theirs resources, especially in collaboration-based systems where peers are assumed to be rational [114, 58, 125]. Schemes based on *micropayments* protocols tend to meet fairness, assuring that

providers are guaranteed to be paid, while requesters are discouraged to behave as freeloaders because they are refunded for each upload. System users are therefore given an incentive to work together towards a common goal [131].

The lack of cooperation (free-riding) and its complex dynamics has been also studied by adopting a game-theoretic approach. As an example, the reader is referred to [52], where the Generalized Prisoner's Dilemma (a well-known game) [48] is studied from the perspective of its possibilities as a model to encourage cooperation.

Mechanism to discourage selfish behavior have been also proposed in different environments. For instance, in [45] it is introduced the idea of using cryptographic protocols based on proof-of-work to increase the cost of sending email and make sending spam unprofitable. This concept has been extended to more general settings, such as preventing network level DoS attacks for TCP [133]. We have mentioned some of the consequences when a node acts maliciously in flooding-based overlay topologies. Peers can stem the flood of requests by requiring that requests must be accompanied by requester proof of work (e.g. solve a cryptographic puzzle.) Ideas similar to this might be extended to P2P networks, particularly to impede DoS attacks as well as to provide a solution for the free-riding problem [67].

Research on this topic could lead to encourage fair content distribution using cryptographic puzzles, since sharing can be encouraged by imposing a cost on the downloads. An alternative to client puzzles is to use the reputation systems mentioned above to track individual machine's utilization of networks resources. Abadi et al. [7] contribute with an approach based on the application of memory-bound functions to discouraging spam. Possibly, its application for authenticated distribution of contents is feasible and even more convenient than mature reputations systems. However, in order to deploy such schemes in mobile networks we must deal with additional concerns, e.g. those related to the devices' low capabilities. An interesting question is whether this idea is feasible in mobile computation, without a significant lose of efficiency for victims.

2.4 Emergent Trends and Future Research Directions

It is generally expected that new forms of networking infrastructures and applications become soon widespread. Even though research and technical progress on these areas are nowadays very intense, some important problems remain to be solved. In this regard, security concerns are among the most important issues, and, particularly, the problems associated with adapting security solutions which were conceived for completely different scenarios.

Whatever the case may be, it seems clear that, in absence of fixed and centralized authorites, cooperation will play an essential role in the provision of such services. We have identified three main open research lines: the use of self-issued-certificates, group-based, and evolutionary techniques.

2.4.1 Certificate-based Solutions

Alternatives to threshold schemes range from endowing the system with pre-authentication via a location limited channel, to using a tamper resistant storage of group key (specially for nodes with very limited resources e.g. sensors.) A more interesting approach from the security standpoint is to use self-issued certificates, i.e. nodes issue certificates to trusted nodes. The use of certificates has been already addressed in trust establishment models [22], even under the assumption of existing sparse social relationships among nodes. Nevertheless, main contribution on this topic is the application of Zero Knowledge Proofs. Briefly, an entity "Prover" knows a secret, meanwhile entity "Verifier" wants P to prove his knowledge. Thus, P runs a challenge-response protocol offering a hard problem to V. At next stage, V proposes a random challenge; P provides the solution. V checks and decides to accept (or reiterate) or reject. This way, after the interaction, "verifier" is convinced that "prover" knows the secret, but "verifier" has *zero knowledge* about the secret itself.

Perhaps, the main drawback is the requirement of a bootstrapping phase with a secret dealer. In the cited work [22], the trust establishment for any two nodes turns into a certificate chain detection problem in a certificate graph, since a node obtains his secret short list (k bindings of his identifier and public key) from a secret dealer, and then generates a certificate locally. However, to

manage trust and reputation locally with minimal overhead, in terms of extra messages and time delay, is still an open issue. In particular, high latency, bandwidth usage and energy consumption are not acceptable in situations with strict real-time requirements, like MANETs. In this sense, this proposal tries to restrict flooding to a subset of nodes. How to determine the subset, such that it covers the sufficient number of nodes holding the required trust information, becomes a problem. Consequently, novel works should have to lead towards "one-to-one" models, where an arbitrary transaction (interaction between two nodes, e.g. a requester and a provider) is triggered by the requester and may be accepted/rejected individually by the provider [92].

In addition, revocation is an important process that generally must accompany the use of certificate-based models. It refers to the procedure for downgrading or eliminating nodes' privileges. Classic methods based on distributing Certificate Revocation Lists (CRL), Certificate Validation Protocols (SCVP), or Online Certificate Status Protocols (OCSP) may not be applicable for dynamic environments in which there are no centralized repositories. A recent work, presented in [14], addresses the issue of certificate revocation without input from external entities, employing threshold cryptography and profile tables.

2.4.2 Social and Group-based Solutions

Some of earlier cooperation-based approaches focused on discovering community social patterns. In social networks phenomena, node popularity can be stemmed from the position of a node within the network, i.e. the degree (the number of neighbors), and the number of hops he is to every other networking node. However, since nodes are autonomous, selfish, rational and strategic, the optimal action for the individual does not produce the best outcome for the population as a whole. In this situation, most popular nodes (from different studies [30], there are no more than 1% of popular nodes) are acting as centralized hosts for the network, and therefore lead to the network losing the benefits of a decentralized architecture. Rather than use a pre-established social network, the network should tend to be dynamically created at runtime.

Intra-group communications have become an important issue in wireless networks. Traditional multicast protocols have been developed and extensively evaluated. In order to prevent attackers from paralyzing the network

and services by manipulating multicast communication, typical scenarios demand the design of protocols that cannot base their security on the existence of setup information, and having no pre-established trust relationship either [33]. In particular, this kind of environments, in which multiple groups may coexist interacting via multicast traffic, should enforce security, and should be protected by means of cryptography, e.g. counting on secret keys. In this sense, robust cryptographic solutions are applied more and more in MANETs specially to provide key management schemes [123, 124].

2.4.3 Using Evolutionary Algorithms

The idea of self-organizing cooperation is being currently addressed by different research directions, e.g. mobile robots connect with each other in a MANET to coordinate their movements. Other trends, such as the Adaptive Multi-Agent System theory [21], focus on the design of cooperative systems in which agents cooperative interact. Each agent has the ability of self-organization, and it is locally "cooperative", but not altruistic. They also has the ability of locally rearrange its interactions with other agents according to its individual task, and this can indeed lead to a change at the global function.

On the other hand, several approaches have already taken inspiration from a number of natural phenomena, which give potential emergence of possibly self-organizing behaviors and tools to be adopted in decentralized deployments [128]. For example, ant foraging has motivated the design of some adaptive P2P agent-based frameworks by relaying on distributed mobile agents ("ants") which can directly interact and cooperate, even re-organize, while leaving and retrieving bunches of data in the visited hosts. Moreover, some proposals in the area of sensor networks exploit learning theories and evolutionary approaches to have systems autonomously learn how to self-organized itself. However, since no possibility of control is assumed, and due to the complexity (often non-linearity) of the phenomena involved, it is difficult to prove that the system will behave as needed. In such cases, some proposals are reasonably (i.e. probabilistically) confident that the global evolution of the system will eventually lead to the desired coordinated behavior.

Additionally, novel approaches are being studied by means of mechanisms inspired by biological evolution such as mutation and recombination, natural selection and survival of the fittest. In particular, some works address the

	ARCHITECTURES AND SYSTEMS								
	Structured						**Unstructured**		
	[99]	[118]	[101]	[132]	[69]	[56]	KaZaA	[40] [42]	**GRID** [13]
ID Assignment									
ID spoofing	♢	△	♢	♢	–	♢	–	–	–
Pseudospoofing	–	–	–	–	–	–	◯	♢	–
Routing									
Churn	♢	▽	–	♢	–	–	–	–	♢
DoS	♢	△	♢	–	△	△	♢	♢	–
Dishonest Nodes									
Cheating	◯	◯	◯	◯	–	△	–	♢	♢
Sybil	–	–	–	♢	–	△	–	–	–
Man-in-the-Middle	–	◯	△	–	–	△	–	–	–
Properties									
Availability	♢	◯	♢	–	♢	–	♢	♢	♢
Integrity	–	♢	♢	♢	–	♢	–	♢	–
Authentication	–	◯	–	–	–	–	–	♢	♢
Confidentiality	–	–	♢	♢	–	♢	–	–	–
Anonymity	◯	♢	◯	◯	–	♢	♢	◯/♢	–

Table 2.1: Security properties considered by overlay architectures, according to if they apply detection methods(△), protection mechanisms (▽), both applied (♢), when be deficient (◯) or when not (–).

evaluation of certain protocol's suitability using evolutionary techniques, with the aim of find the optimum sequence of premises, e.g. using random graphs in a Game theory approach. Concretely, candidate solutions to the optimization problem using a play the role of individuals in a population, and the cost function determines the environment within which the solution "lives" [48]. In fact, P2P and MANET's general principle, i.e. self-organization, paves the way for considering the use of evolutionary techniques in analysis processes as a open research challenge. In addition, evolutionary techniques mostly involve meta-heuristic optimization algorithms such as self-organization. A range of scenarios are already under formation assuming a spontaneously networking environment. Those scenarios may exhibit spontaneously emergent behaviors which need for methodologies to predict them, like a self-adaptive system.

2.5 Analysis

In this section, we analyze the security properties considered by the three categories which this survey has discussed, i.e: overlay routing (see Table 2.1), user community (see Table 2.2), and content distribution (see Table 2.3) in a P2P network. Note that the study takes into account several dimensions according to the structure, architecture and system affected by some security

	Trust	**Anonymity**		**Authentication**		**Middleware**
	[47], μ-payments	[100], [73]	[56], [42]	Oceanstore, [97]	[47]	
Traceability						
ID spoofing	▽	–	♢	–	–	◯
Pseudospoofing	▽	◯	♢	–	–	◯
Man-in-the-Middle	♢	–	–	♢	♢	◯
Availability						
Attrition (DoS)	–	–	♢	△	–	–
Dishonest Nodes						
Cheating	–	◯	◯	–	–	◯
Sybil	◯	◯	◯	–	–	–
Free-riding	◯	◯	◯	–	◯	♢
Fairness	♢	–	–	♢	–	–
Properties						
Availability	–	–	–	◯	–	♢
Integrity	♢	▽	▽	♢	♢	–
Authentication	♢	–	–	♢	♢	♢
Confidentiality	♢	♢	♢	♢	♢	–
Anonymity	◯	♢	♢	–	◯	–

Table 2.2: Security properties considered by P2P architectures and systems for user community management. (Same legend as in Table 2.1).

attacks. Thus, for each approach, we analyze the degree of detection and protection, even the absence of countermeasures against the exposed attacks.

Each row in the tables corresponds to a particular class of attack, while columns indicate if a specific architecture, model or system implements mechanisms for defending against it. At first sight, from Table 2.1, it might appear that current proposals explore the benefits of enhanced request routing in P2P file sharing, most of them against DoS attack and ID spoofing. The most significant proportion of the research efforts are essentially worried about availability and integrity properties. Current efforts in overlay are focused on authenticated query routing, while some of them are only studying the consequences of malicious actions and proposing protection models against cheating and Sybil attacks (e.g. `Gnutella` and `Tarzan`.) Analogous difficulties arise in the real application of anonymity.

Table 2.2 contains the analysis of the peers' behavior at most popular reputation systems, anonymity architectures and application-specific models, showing that there is significant heterogeneity in peer traceability, availability, and vulnerabilities. Confidentiality, anonymity and integrity are not taken into account in most systems. Nevertheless, to understand which issues account for these misses, we first explored the relationship between the protection against DoS attacks provided by main P2P overlays, obtaining poor matches. Unfor-

	Storage and Replication		**Search and Retrieval**		**Integrity and Authentication**	
	[34], [81]	[40]others	[40], [118], [99]	*Ranking*, Similarity	[67], [7]	Reputation S.
Identification						
Pseudospoofing	♢	–	–	–	–	◯
Blacklisting	–	♢	–	♢	–	♢
Service Avail.						
DoS	–	–	♢	–	▽	–
Attrition	♢	–	♢	◯	▽	–
Churn	♢	–	–	△	△	◯
Dishonest Nodes						
Cheating	◯	♢	♢	♢	–	♢
Sybil	–	–	–	–	▽	–
Free-riding	◯	–	♢	–	–	–
Man-in-the-Middle	–	♢	–	–	–	♢
Fairness	♢	–	–	♢	▽	–
Properties						
Availability	♢	♢	♢	♢	–	–
Integrity	–	–	–	♢	–	♢
Authentication	–	△	–	♢	♢	♢
Confidentiality	–	–	–	–	–	△
Anonymity	◯	◯	◯	◯	–	◯

Table 2.3: Security properties considered by P2P architectures and systems for contents management. (Same legend as in Table 2.1).

tunately, in any case the protection against dishonest nodes manipulations are devoid of any detection mechanism. Fairness begins to be taken into account to control aggressive behavior ("antisocial") between peer connections.

Concerning content management, since the performance is sensitive to the degree of user cooperation, it makes sense to provide incentives to users to share their resources. In particular, an adequate option would be to increase their download allocations in a manner that depends on their contributions. As a result, most P2P systems manage efficiently the content availability and a fair sharing (see Table 2.3.) However, anonymity is less considered and, therefore, many attacks are not applicable. This uncertainty is not worrisome, for Table 2.3 does not include systems based on anonymity protocols. We have thoroughly examined the range of activities performed by dishonest nodes against almost all the security properties of the content. Based on these results, we can conclude that every system discussed in Table 2.3 is vulnerable, at least, to one of the mentioned attacks.

2.6 Conclusions

Infrastructure-less networks, on which, in general, one cannot assume the existence of centralized services such as those provided by TTPs, present a challenge in terms of formalizing security protocols in P2P networks. Among most canalized security problems at present, we enumerate goals in adaptability, self-management, scalability, fault-resilience in the presence of network and computing failures, and availability in the presence of peers' transience, all with the lack of a CA. These motivations are mainly attributed to organization, location and routing algorithms studied in this chapter. Next, we summarize main contributions of this chapter, as follows:

- We performed the present study of P2P content distribution systems and infrastructures by identifying the feature space of their functional and non-functional characteristics, linking them to current security challenges (anonymity, fairness, scalability, performance, content management, etc.) and without forgetting emergent applications such as MANET, GRID, and collaborative environments.

- We have therefore presented a survey of existing security approaches in P2P networks according to the P2P architecture adopted. It was proposed to categorize the most popular protocols depending on how they detect and protect.

- Our analysis summarizes the security characteristics adopted by those P2P structures.

- We have stressed the most representative cooperation-dependant security approaches using P2P scenarios in order to highlight the benefits of such a dependence. In our opinion, these schemes we have outlined in this proposal offer major advantages over other existing ones.

- Providing efficiency and optimization for the nodes is quite challenging, given the computational and communication overhead that cooperation-based security schemes generally incur, and also considering nodes' limitations. Nevertheless numerous members within the research community are currently working on optimal solutions for maintaining reasonably

high levels of cooperation. The future of wireless/mobile ad hoc systems relies on the ability to provide efficient security support that can be performed even in the presence of colluders.

Future work should focus on extensions to the following items:

- Study emergent topics related to the dynamics of cooperation and fairness, and which strategy leads to the formation of interesting social profiles, and content integrity protocols based on cooperation.
- As shown, although most approaches assume rational behavior, we must be able to consider non-collaborative peers and to measure the effect they might have on the overall system performance. Therefore, we have observed that Game Theory may serve us in analyzing the suitability of the protocols we propose in this thesis.
- P2P systems for mobile ad-hoc networks introduce a number of new issues related to naming, discovery, communication and security. In particular, these systems require a lightweight and efficient architecture due to their highly dynamic nature.
- The idea of using cryptographic puzzles for decreasing spam is being extended to P2P networks. This idea could provide a form of access control and detect DoS attacks in advance.

Chapter 3

A P2P Content Authentication Protocol based on Byzantine Agreement

One of the features offered by nearly all P2P networks is the possibility of having several replicas of the same content distributed among multiple nodes. Despite the fact that this functionality has many advantages (e.g. robustness and fault tolerance, which allow to access contents even when some nodes are disconnected), a crucial requirement is to guarantee basic security properties, such as content authenticity and integrity. A high degree of content redundancy implies that it is necessary to apply some security mechanisms in order to avoid attacks based on non-authorized content access and modification. These mechanisms would pave the way for new models in which content providers can exert some control over the replication and file sharing process.

In most existing P2P file sharing systems, verifying the integrity of contents depends on a correct node authentication. Unfortunately, no infrastructure there exists for identifying peers and providing them with digital certificates. A peer can publish fake or junk files with the names or keywords of some popular files, causing normal users to frequently download wrong files. This quickly makes peers lose trust and interest in the community [130]. The objective of checking content integrity is not only to verify that data is not corrupted, but also to validate that contents are really what one has requested.

However, the extremely decentralized nature of these environments makes impossible to apply classic solutions that rely on some kind of fixed infras-

tructure, typically in the form of on-line TTPs. In this chapter, we propose a content authentication protocol for pure P2P systems based on the use of attribute certificates without relying on the existence of a PKI or any other form of centralized authority. Instead, our scheme maintains content integrity based on the collaboration among a fraction of peers in the system, who play the role of a distributed CA. Under certain restrictions, our proposal assures content integrity in a P2P file sharing system, i.e. a guarantee that the file has not been altered even if it is a replica of the original, and therefore the owner has lost control over it. Finally, we provide an analysis concerning the efficiency (computational effort and communication overhead) and the security of our proposal.

Next, we first introduce the background and motivation of our approach, and then describe our solution in an extensive form. Subsequently, we provide the performance and efficiency analysis, and informally discuss a security analysis through a number of attack scenarios. Finally, some open issues and future work conclude the chapter.

3.1 Background and Motivation

A digital content in a common file sharing scenario is straightforwardly alterable; it can be manipulated, so that a binary stream looks like the original. The solutions to achieve data integrity and/or content authentication, as many other network security services offered today, rely on public key cryptography. Once that a public key can be securely associated to any given party, the integrity of any content generated by her can be ensured through her signature, thus maintaining the correctness and consistency of global data structures and shared contents, even when peers independently and unpredictably join and leave the system. Nevertheless, public key certification authorities do not, traditionally, certify the behavior of the entities that possess their certificates.

3.1.1 Classic Approach

In classic networking paradigms, guarantees of authenticity and integrity can be provided by digital signatures. If an authenticated user, A, wishes to offer a content m, she can rely on a CA to generate and sign an associated evidence,

which can be checked by the rest of the community and also ensures that m has not been modified.

The integrity of a content generated by a source can be ensured as follows. First, the source:

1. Computes a hash value $h(m)$ from m.
2. Encrypts $h(m)$ with her private key, obtaining a digital signature.
3. Creates a message containing m (probably encrypted using a session key), its hash, and her signature.
4. Sends this message (including the session key encrypted using the CA's public key) to the CA.
5. The CA looks up author's public key and verifies her signature.
6. The CA signs $h(m)$, $s_{CA}(m)$, and sends this signature to the source.
7. Finally, the source can assure the integrity of the content publishing:

$$\langle m, s_{CA}(m) \rangle$$

Then, the receiver:

1. Retrieves m and the CA's signature.
2. Decrypts the enclosed digital signature by using the CA's public key certificate, thus generating a first hash value.
3. Computes a second hash value from the received content.
4. Compares both hash values to confirm the non-alteration of the content.

Even though the previous approach has been successfully applied in several domains (i.e. for public key authentication), it requires the existence of at least one TTP. The reasons why A cannot sign her own evidences are simple. First, because she can misbehave, offering something different of what she announces. Furthermore, her signature alone does not prevent from manipulation. Suppose that A offers m in the form of a pair:

$$\langle m, s_A(m) \rangle$$

Once B obtains m, she can modify it and generate a new signature over the altered content. Moreover, even if B does not modify m, she can just remove $s_A(m)$ and add her own signature. As a result, several –and probably different– copies of m claimed by various parties may be circulating through the network.

3.1.2 Overview of our Approach

One of the most important issues when dealing with public keys is ensuring their authenticity. In environments such as small/medium size networks, the classic solution relies on the existence of a PKI. A hierarchy of CAs can assure whether a public key belongs to someone or not. Since in pure P2P networks, especially in the case of MANETs, there is neither any fixed infrastructure nor TTPs [91], we have to deploy such functionalities applying novel alternatives, or even solutions derived from well-studied problems such as those of reaching consensus in presence of traitors. Due to the relevance of public key authentication as an essential building block in the protocol proposed in this chapter, we have further elaborated on this proposal in previous chapter (see Section 2.2.)

Particularly, since malicious attacks and dishonest peers can cause faulty nodes to exhibit Byzantine behavior, fault-tolerant algorithms are becoming increasingly important in many environments. As an example, the work described in [76] proposes a mechanism based on erasure code replication for an effective replica distribution and storage. This technique is based on breaking the data into blocks and spreading them over many servers. The objects and their associated fragments are then named using a secure hash over the object contents, giving them globally unique identifiers. This provides with data integrity by ensuring that a recovered file has not been corrupted (for a corrupted file would produce a different identifier.) The authors use a Byzantine agreement protocol to guarantee Byzantine-tolerant consistency as well.

Similarly, the public key authentication scheme adopted in our model depends on an honest majority of a particular subgroup of the peers' community, labeled "trusted group". Concretely, honest members from trusted groups are used to provide a functionality similar to that of the PKI through a consensus procedure. However, in P2P systems an authenticated peer could create multiple fake identities and act maliciously in the future. A periodic pruning of

the trusted group is therefore required in order to ensure the honest majority. Obviously, this updating process leads to some performance drawbacks that we must analyze.

In summary, content authentication confirms non-alteration and source identification of the content, implemented through a digital evidence. The owner of the content is responsible of generating such an evidence containing the most important features of the content, while a selected subset of the community signs it. Even though several signers do not constitute by themselves a proper TTP, some security properties can be ensured if the group has honest majority.

3.2 Working Assumptions

Before presenting the details of this proposal, we assume the following five working hypotheses:

1. Assured transactions without rejections. The absence of a message can be detected. This can be provided by using a scheme based on timeouts and/or an appropriate transport protocol such as TCP.

2. Identification of all participants is required through a unique pseudonym, the IP address, a network name, etc. Anonymity is not desired by now.

3. Identification of contents is also required. A unique name, which is also used for searching the content, is associated with the content.

4. Digital signatures cannot be forged unless the attacker gets access to private keys.

5. Anyone can verify the authenticity of a node's signature by applying the Byzantine fault tolerant public key authentication protocol presented in [97].

3.3 Content Certificate

The details of a content certificate and a full description of the scheme are provided below. The basic idea is that contents will be associated to a digital

certificate ensuring properties such as integrity and authenticity, much in the way an X.509 public key certificate can be used to ensure these properties for a public key. According to RFC 3281 [50], the attributes are digitally signed and the certificate issued by an attribute authority –an entity that pure P2P networks do not have. Instead, our scheme uses a classic challenge-response procedure among a subgroup of peers until reaching a consensus.

Let A be the legitime owner of a given content m. After A joins the system and shows interest in distributing m, she must first produce two main items, as follows:

- A digest of the file, $h(m)$, using a one-way cryptographically strong hash function (such as any of the SHA-2 family.)
- A subgroup $\{n_1, n_2, \ldots, n_k\}$ of k cooperative nodes picked up from her group T_A of trusted nodes. These k nodes are also called *signers*. The way in which this selection is carried out may consider the following criteria:
  - Past transactions history. Basically, nodes use information about past transactions to make decisions regarding current interactions.
  - Availability level and reputation score level. Among the information that most commonly used P2P systems present, one can find several weighted values: the reputation of files and nodes, and the fraction of time a piece of data is accessible. These weights may serve to compute an estimation of the expected success for the current transaction.

The content certificate C_m (Figure 3.1(a)) is structured in two parts, as follows:

- The certificate C, containing the following fields:
  1. The identity of the originator, which ultimately establishes who has generated the content and is its legitimate owner.
  2. The identity of the content.
  3. A hash, $h(m)$, of m, assuring its integrity.

Content certificate $\mathbf{C_m}$

`Certificate` C:
`Holder:` A
`ID:` I_m
`Content:` $h(m)$
`OLS:` $A, n_1, \ldots, n_k$
`Validity Period:` (ts_1, ts_2)
`Signing Algorithm:` *AlgorithmDesc.*
`Signatures:`
$E_k = E_{K_{n_k}^{-1}}(\cdots(E_{K_{n_1}^{-1}}(E_{K_A^{-1}}(h(C)))))$

(a) Content certificate structure.

Table of signed certificates $\mathbf{S_{n_i}}$

Date	Owner	Content Hash	Certificate received	Signature $\mathbf{E_{n_i}}$
$Time_1$	n_1	$h(m_1)$	C_{m_1}	E_{i1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$Time_n$	n_n	$h(m_n)$	C_{m_n}	E_{in}

(b) Local register for signers.

Figure 3.1: Content certificate and local database maintained by each certification node.

4. An ordered list of signers (OLS) of the certificate. It contains the identity of $k+1$ network nodes, denoted by $n_0, n_1, \ldots, n_k$, being $n_0 = A$ the content originator.
5. The validity period (ts_1, ts_2) for C_m, establishing that the certificate is valid from ts_1 until ts_2.
6. Description of the hash and signature functions which have been used.

- Finally, the previous items are recursively signed by the nodes listed in the OLS. First, a message containing C is signed by A. The result, along with m, is then sent to the selected k participants. The resulting signature is subsequently checked and signed by n_1, and so on. We further elaborate on this signing process in the following sections.

Furthermore, as it will be justified below, each node n_i must maintain a local register S_{n_i} with the certificates it has previously signed. Fig. 3.1(b) shows the fields that this table should store:

- A time-stamp, showing the time when the certificate was signed.

- The content owner identification.
- The hash of the signed content.
- The received certificate (including the signatures contained in it so far), and
- The signature generated by the node.

3.4 Content Certificate Generation

Before C_m can be used to ensure content authenticity and integrity, it must be progressively signed by the nodes included in the OLS. At each stage, the next node in the OLS adds its signature to the previous ones. Due to the structure of the chain of signatures, this task cannot be carried out in parallel[1]. We will denote by

$$C_0, C_1, \ldots, C_k = C_m \tag{3.1}$$

the successive versions of the certificate as it passes through the list of nodes.

The certificate is initialized by the originator, A, who selects an appropriate value for the number k of signing nodes and their identities (see discussion above.) Next, A generates C_0 by providing the first signature and passes it to the next node in the OLS. The chained signature is defined as follows:

$$\begin{aligned} E_0 &= E_{K_{n_0}^{-1}}\big(h(C)\big) \\ E_{i+1} &= E_{K_{n_{i+1}}^{-1}}(E_i) \end{aligned} \tag{3.2}$$

3.4.1 Local Verification

Signers have to perform a *local verification* stage in order to ensure that they are not being cheated on. This consists in the following three steps:

1. *Certificate verification.* Each participant/signer, $n_i \in$ OLS, should verify the correctness of the received certificate C_{i-1}. This includes:

   1.1 Obtaining the owner's public key, and authenticating previous signers as well.

[1] We discuss later about group signatures.

1.2 Computing $h(m)$ and comparing it with the value contained in the received certificate.

1.3 Checking its table of signed contents, and verify that no entries exist corresponding to the same content.

2. *Signatures verification.* Each peer verifies the signatures contained in the received certificate according to the list order. If any public key is unknown, it can be acquired with an instance of Pathak and Iftode's public key authentication protocol.

3. *Local management.* If previous verifications succeed, the node adds its signatures to C_{i-1}, thus creating C_i. Furthermore, each n_i dumps the received certificate (including the signatures contained), a timestamp, and the generated signature $E_{K_{n_i}^{-1}}(E_{i-1})$ on his local table of signatures S_{n_i} (Fig. 3.1(b).)

Finally, A publishes the content m, along with the content certificate:

$$(m, C_m)$$

For the sake of illustration, we include the following table which summarizes the content certificate signing process. Moreover, Figure 3.2 schematically overviews the entire process that signers in the OLS must perform.

Certificate generation: Signers' deliverables

	Receives	Generates
n_0		$C_0 = \langle C, E_{K_{n_0}^{-1}}\big(h(C)\big)\rangle = \langle C, E_0\rangle$
n_1	C_0	$C_1 = \langle C, E_{K_{n_1}^{-1}}(E_0)\rangle = \langle C, E_1\rangle$
n_2	C_1	$C_2 = \langle C, E_2\rangle$
$\vdots$	$\vdots$	$\vdots$
n_k	C_{k-1}	$C_k = \langle C, E_k\rangle$
n_0	C_k	

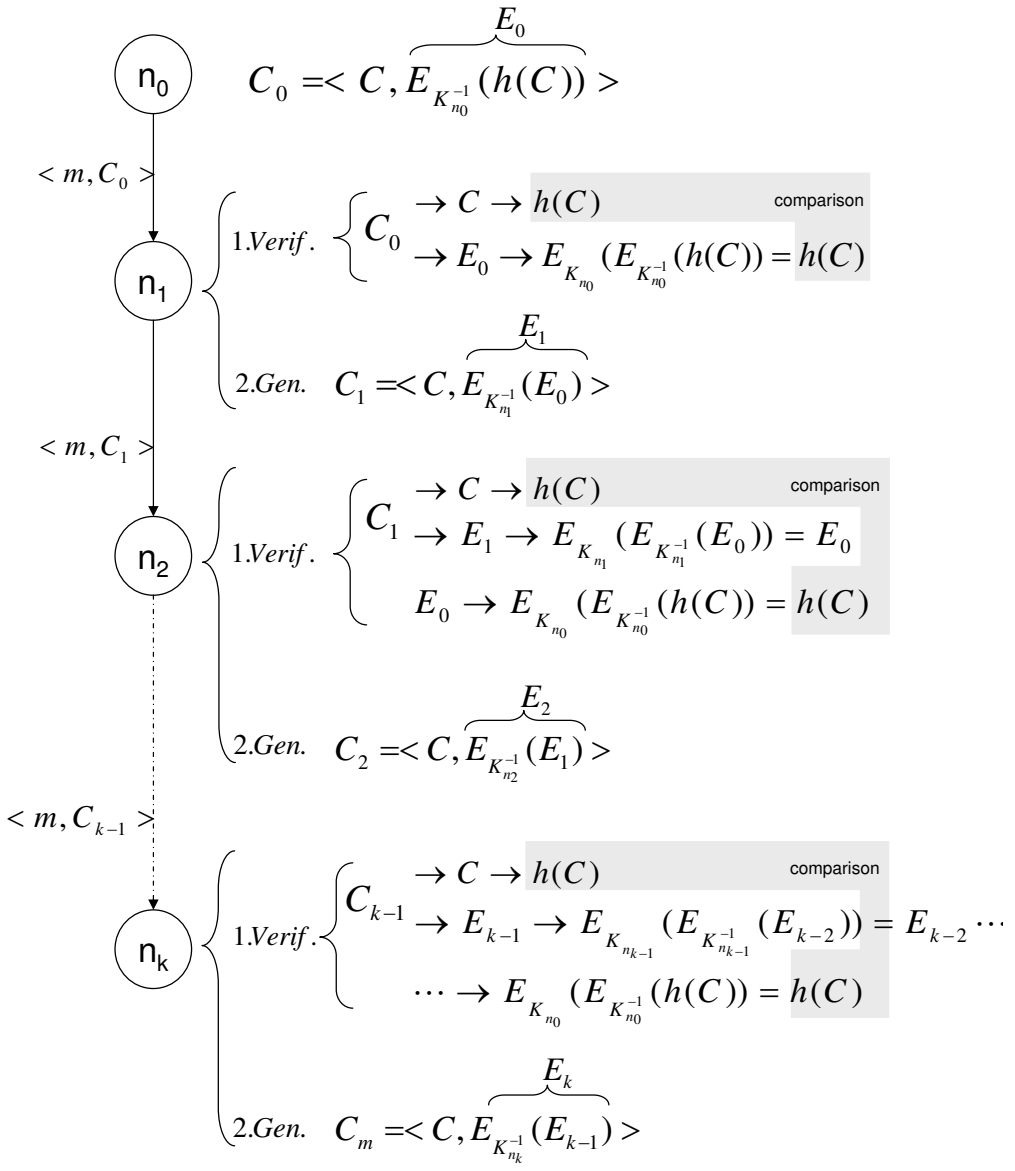
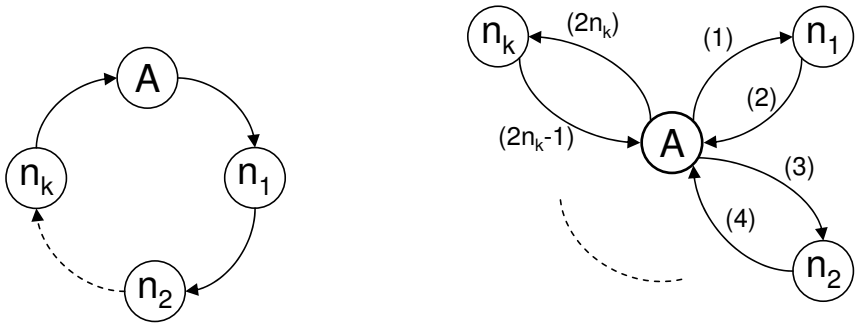


Figure 3.2: Summary of the content certificate generation process.

3.4.2 Signing the Certificate

We have identified two different ways in which the certificate can be signed by the nodes included in the OLS. Fig. 3.3 graphically illustrates both alternatives. The main differences are the following:

- In the first one, illustrated in Fig. 3.3(a), each node is responsible of sending the signed certificate to the next node in the list. In this way, A



(a) (b)

Figure 3.3: Alternative procedures for content certificate generation.

simply sends the initial certificate, C_0, to the first signer, n_1, and waits until C_m arrives from n_k (as shown in Fig. 3.6.) Although it is not explicitly pointed out in the figure, we assume that each peer must send a notification message to A when it passes the certificate to the next node. This, together with appropriate timeouts, allows A to be aware of the current state of the process.

- In the second alternative (Fig. 3.3(b)), A is responsible of sending C_{i-1} to each node and receiving C_i. Now, A can check whether the received certificate has been properly signed or not, thus having a higher level of control over the process. However, note that each node still has to perform its local verification stage in order to ensure that it is not being cheated on.

A more precise description for both alternatives is provided in Figs. 3.4 and 3.5. Moreover, Fig. 3.6 depicts graphically the signing process of content authentication protocol.

3.5 Certificate Verification

Let B be the requester who wishes to access m. We can assume that, at this time, B has already completed a searching process and obtained a list of sources that keep a replica of the desired content. A query result should return the content's descriptor, which could be necessary to rank the relevance

Protocol for distributed content certificate generation
(Note: $A = n_0$)

1. n_0 generates C and signs it: $C_0 = \langle C, E_{K_{n_0}^{-1}}(h(C))\rangle = \langle C, E_0\rangle$
2. n_0 sends (m, C_0) to n_1
3. For $i = 1$ to k
   (a) n_i performs the local verification stage on C_{i-1}
   (b) n_i adds its signature and generates $C_i = \langle C, E_i\rangle$
   (c) n_i updates S_{n_i} with the tuple $\langle ts_i, A, h(m), C_{i-1}, E_{K_{n_i}^{-1}}(E_{i-1})\rangle$
   (d) n_i sends (m, C_i) to $n_{i+1(mod\ k)}$
   (e) n_i sends a notification message to n_0

Figure 3.4: Distributed content certificate generation.

Protocol for centralized content certificate generation
(Note: $A = n_0$)

1. n_0 generates C and signs it: $C_0 = \langle C, E_{K_{n_0}^{-1}}(h(C))\rangle = \langle C, E_0\rangle$
2. For $i = 1$ to k
   (a) n_0 sends (m, C_{i-1}) to n_i
   (b) n_i performs the local verification stage on C_{i-1}
   (c) n_i adds its signature: $C_i = \langle C, E_i\rangle$
   (d) n_i sends C_i to n_0
   (e) n_i updates S_{n_i} with the tuple $\langle ts_i, A, h(m), C_{i-1}, E_{K_{n_i}^{-1}}(E_{i-1})\rangle$
   (f) n_0 verifies the correctness of the signature of n_i

Figure 3.5: Centralized content certificate generation.

of the resulted content to the query, as well as the list of identities of the source nodes. Together with each query result, B obtains sources' published information, C_m with all the information associated with the content, and m, as explained previously.

B first selects a source A among those returned by the query. This selection could be done taking into account the knowledge (reputation) of some of the signers, of the source, or simply at random. Whatever the case may be, B finally recovers m. The correctness of the certificate should be verified to ensure the authenticity and integrity of m, so B must verify the correctness either of some or all of the chained signatures and their identities as well. For this, B performs the following steps:

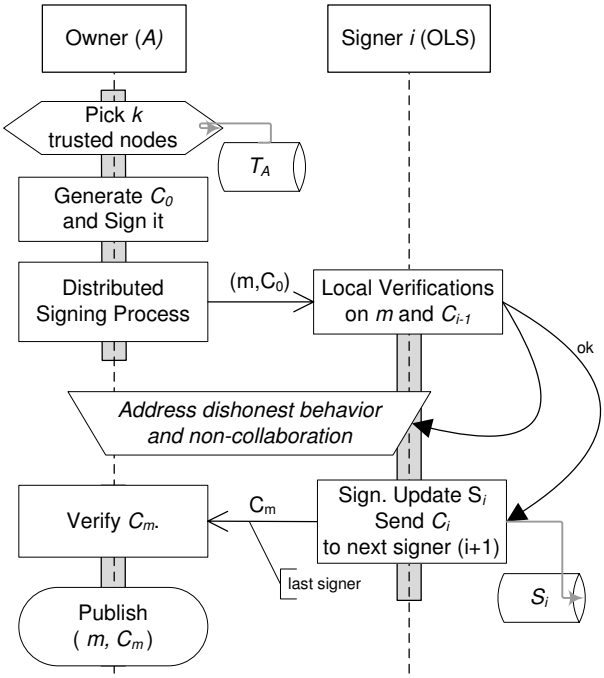


Figure 3.6: Content authentication protocol: Certificate generation

- *Step 1.* B computes $h(m)$ from m and compares the result with that included in the certificate. If both values differ, then either m has been altered or C_m is not an authentic certificate for m.

- *Step 2.* B should authenticate the peers listed in the OLS. At worst, B knows none of the signers' identities, having to run Pathak and Iftode's protocol, and pre-evaluating the new members' trust based, for instance, on the weighted values given by a Trust Management System (TMS) [112].

- *Step 3.* B verifies the chain of signatures by recursively encrypting C with the ordered list of public keys.

3.6 Efficiency Analysis

This section presents an efficiency analysis considering both the computational (especially cryptographic) effort required by the nodes, as well as the communication cost of the scheme.

Algorithm	ms/operation
RSA 2048 Encryption	0.08
RSA 2048 Decryption	2.78
RSA 2048 Signature	2.78
RSA 2048 Verification	0.08
	MB/second
AES-256	113
SHA-256	106

Table 3.1: Speed of cryptographic primitives used in our simulations [5].

3.6.1 Computational Effort

First we analyze the theoretical efficiency according to time, memory and computational resources required by each protocol operation.

Table 3.2-(a) presents the content access life cycle summarizing the time sequence, the number of cryptographic operations and the computational complexity for each stage of the protocol. Concretely, the computational complexity of evaluating the cryptographic operations required by our content authentication protocol is $\mathcal{O}(|m|k^3 \log k)$, which linearly depends on the content size, and is bounded by a polynomial in the amount of protocol participants.

Furthermore, we include in Table 3.1 the speed benchmarks corresponding to the cryptographic primitives used. We use previous tables with the aim of measuring the computational cost for content certificate generation process.

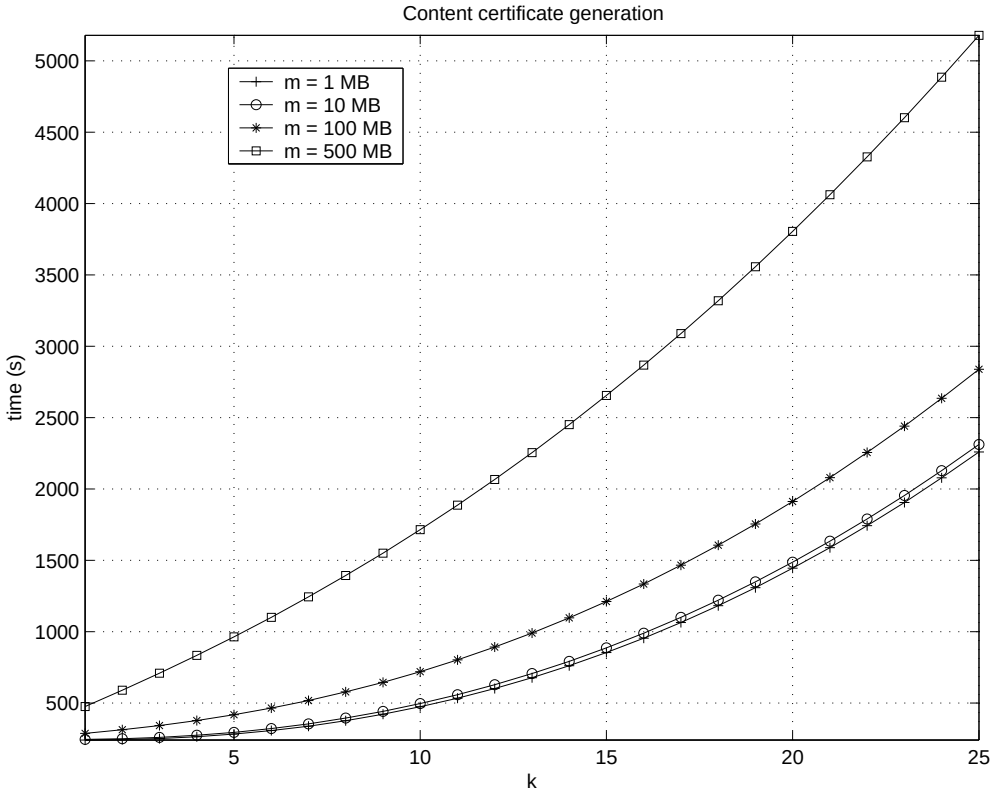
Figure 3.2-(b) shows the cost in the *worst* case (no signers are known and all the verifications must be performed), so the curve must be seen as an upper bound. In fact, content authentication protocol performs a number of hash generations, signature generations and verifications which depends on the number k of signers. Of course, this also implies that k instances of Pathak and Iftode's protocol must be executed, plus the execution of several symmetric encryptions.

For instance, generating a certificate for contents between 1 and 100 MB takes less than 1 minute with the cooperation of 10 signers. As content's size and the number of signers increase, the computational cost increase significatively: A certificate for a content of 500 MB and 100 signers takes approximately 1 hour in the worst case, and 3 minutes when no signer is unknown. See, however, that this task is carried out just once, and that content access is considerably faster.

Stage	No. crypto operations	Complexity
1. C_m generation	$1H + 1S$	$\mathcal{O}(\lvert m\rvert k^3 \log k)$
2. Signature process	$kS + kH + \frac{k(k+1)}{2}V + \frac{k(k+1)}{2}PI$	
3. Content publishing	0	
Subtotal:	$(k+1)(H + S + \frac{k}{2}V) + \frac{k(k+1)}{2}PI$	
4. B's Checking on C_m	$1H + kPI + kV$	

Legend: H: Hash generation; E: Symmetric encryption; D: Symmetric decryption; S: Signature generation; V: Signature verification; k: No. of signers; PI: Pathak and Iftode's protocol ($k \log k$)

(a) Computational complexity for each stage of content authentication.



(b) Computational cost (worst case) for content certificate generation.

Table 3.2: Efficiency analysis (computational effort).

On the other hand, Figure 3.7 shows the computational cost in the best case, i.e. no instance of public key authentication is needed since signers are mutually known. In this case, generating a certificate for contents between 1 and 100 MB takes less than half a second with the cooperation of 10, 20 and 50 signers. The upper bound reached in simulations has been half a minute to

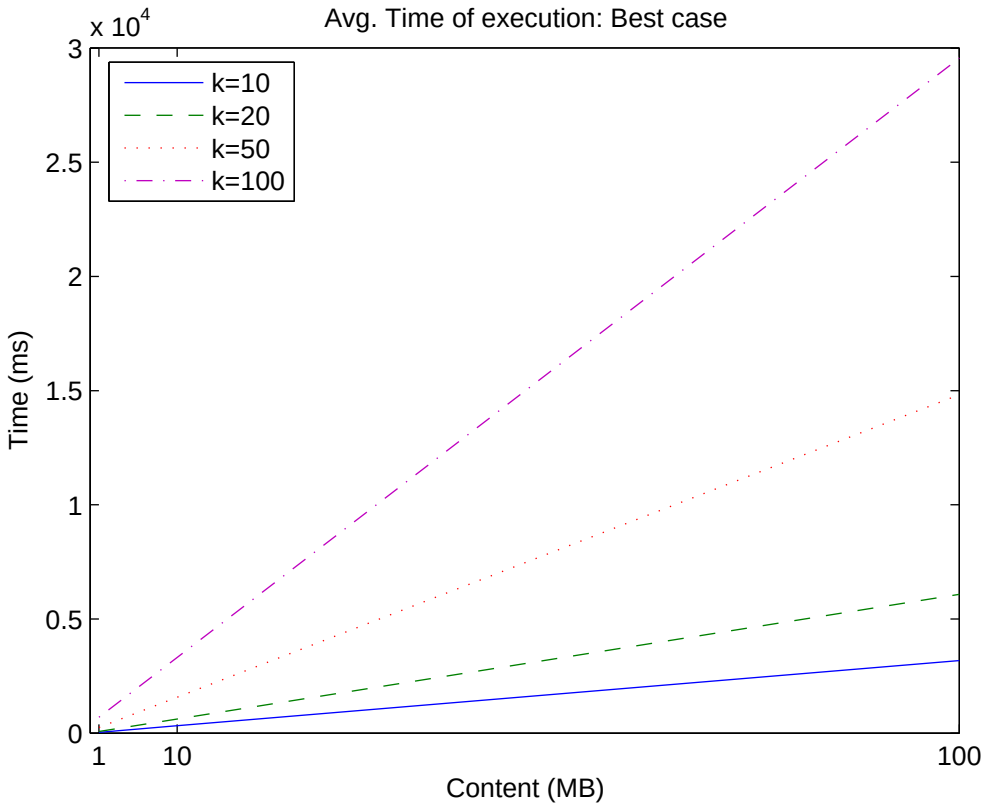


Figure 3.7: Computational cost (best case) for content certificate generation.

sign a 100 MB content by 100 mutually known signers.

3.6.1.1 Average Time of Execution

Even though the use of digital signatures and encryption based on public key cryptography require extra computational cost in our proposal, there are two main responsible factors for the high computational complexity. First, the execution of several instances of Pathak and Iftode's protocol increases the cost $O(k \log k)$. Recall that this protocol must be initiated in order to assist each node' public key verification. Secondly, as the content certificate generation requires the cooperation of k participants, faulty, dishonest or malicious nodes, and network failures as well, involve additional complexity, as shown in Figure 3.8. In such a scenario, the total cost is comprised of an additional, wasted cost which is the result of the unsuccessful instances Δk.

Now, we first evaluate the decrease in the number of public key authenti-

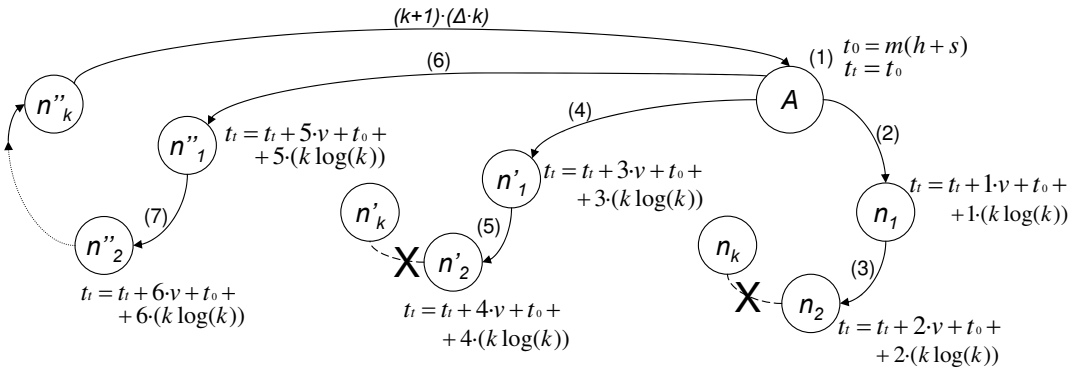


Figure 3.8: Content certificate generation procedure when a signer does not collaborate: Time computation.

cation instances with respect to the proportion of trustworthy peers which are prone to collaborate. Besides, this will allow us to analyze the impact that non-collaborative behavior has on the overall system.

We then make a comparison between simulations of the content certificate generation and signing stages where all participants know each other (best case) with simulations which require public key authentication (worst case.) Moreover, we consider different sets of k participants (10, 20, 50 and 100 signers), and several content lengths (1MB, 10MB and 100MB.) Table 3.3 summarizes the average time of execution in each case. We can derive some conclusions from these results. First, contents' size is actually an influent factor concerning time of execution due to its involvement in signatures by each and every k signers, which also carry weight in the total cost. Nevertheless, we can notice the growing differences between simulations with and without running the public key authentication protocol with respect to the number of involved nodes. For instance, generating a certificate for 1 MB contents, and with the participation of up to 100 signers takes less than one second in both cases. The cost increases for values of k up to 50 and having to authenticate public keys.

A key problem, however, is to guarantee that all the community members will collaborate to jointly carry out a process. We explore the probabilistic nature of non-collaborative behavior in a P2P community. On the one hand, we may apply strictly probability to next interactions and simulate several rounds of instances using randomness and varying the threshold.

10 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	36.1460	321.8600	$3.1790 \cdot 10^3$
with PI	$1.3026 \cdot 10^3$	$1.5883 \cdot 10^3$	$4.4454 \cdot 10^3$
20 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	77.4060	622.8600	$6.0774 \cdot 10^3$
with PI	$1.2659 \cdot 10^4$	$1.3205 \cdot 10^4$	$1.8659 \cdot 10^4$
50 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	249.1860	$1.5739 \cdot 10^3$	$1.4821 \cdot 10^4$
with PI	$2.4964 \cdot 10^5$	$2.5097 \cdot 10^5$	$2.6421 \cdot 10^5$
100 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	695.4860	$3.3189 \cdot 10^3$	$2.9553 \cdot 10^4$
with PI	$2.3263 \cdot 10^6$	$2.3289 \cdot 10^6$	$2.3552 \cdot 10^6$

Table 3.3: Average time of execution (ms): Best and worst case with 100% collaborative nodes, i.e. all participants collaborate.

On the other hand, we simulate experiments based on some kind of conjecture over the collaboration nature of the whole community that one is immersed. The underlying idea is that a node A will try to reduce the cost when she estimates it is highly possible to interact with non-collaborative peers of the community. These experiments work as follows. A content owner first elaborates a conjecture $\zeta \in \{0.0, 0.1, \ldots, 1.0\}$ over the community's collaboration nature. All possible thresholds represent different community profiles, i.e. 0.2 means the existence of a 20% of cooperative nodes out of the total networking nodes in the community. At this point, the collaboration of a certain participant may be modeled using a uniformly defined value $\theta \in U(0, 1)$. i.e. the probability at which a node is assumed to be collaborative. Thus, we experimentally estimate the likelihood of finding k collaborative (consecutive) nodes, i.e. k nodes whose value θ is above the given conjecture ζ set by the user, and jointly cooperate in signing the content certificate. So, if the computed value θ is higher than the threshold ζ, node n_i is supposed to be collaborative and, therefore, the protocol continues.

Experiments were conducted varying the maximum number k of required participants from 10, 50 to 100, and 1, 10 and 100 MBs for content lengths.

$m = 1$MB				
	Best case		Worst: with PI	
	$k = 10$		$k = 10$	
	μ	$\pm\sigma$	μ	$\pm\sigma$
100%	36.15	0	$1.30{\cdot}10^3$	0
90%	55.96	27.04	$1.84{\cdot}10^3$	$7.94{\cdot}10^2$
80%	110.78	80.14	$2.98{\cdot}10^3$	$1.89{\cdot}10^3$
70%	250.30	215.00	$6.19{\cdot}10^3$	$5.29{\cdot}10^3$
60%	783.80	762.31	$1.50{\cdot}10^4$	$1.43{\cdot}10^4$
50%	$3.11{\cdot}10^3$	$3.12{\cdot}10^3$	$5.10{\cdot}10^4$	$4.98{\cdot}10^4$
40%	$1.91{\cdot}10^4$	$1.92{\cdot}10^4$	$2.63{\cdot}10^5$	$2.59{\cdot}10^5$
30%	$2.14{\cdot}10^5$	$6.02{\cdot}10^5$	$2.56{\cdot}10^6$	$2.53{\cdot}10^6$
20%	$7.39{\cdot}10^6$	$7.00{\cdot}10^6$	$7.74{\cdot}10^7$	$8.10{\cdot}10^7$

Table 3.4: Average time of execution (ms): Proportion of collaboration

The experiment was run 1000 times for each threshold ζ. Table 3.4 shows the average values for 1MB contents signed by 10 nodes. The expressed cost takes into account the following computational operations: hashes, signatures and verification of signatures (according to Table 3.1), and also includes the public key authentication cost.

We also ran similar experiments with $k = 50$. Some results were obtained: in the best case with 90% of collaborative nodes out of 50, the process of signing the content certificate takes 6 seconds on average, while with 10 participants and the same percentage of collaboration the average time of execution is 36 ms. Moreover, the fast-growing cost is clear when increasing the percentage of non-collaboration in the community: with 70% and 50 signers (known among them) we obtain a computational cost about $7.24{\cdot}10^4$ seconds.

Now we provide a characterization of the distribution of our results. The purpose of this is to show the most extreme values in the experimental data set (maximum and minimum values, which we can consider them as outermost situations or potential outliers), the lower and upper quartiles, and the median. The results are displayed as boxplots, where the box represents the interquartile range, and whiskers extend to the 25th and 75th percentile of the simulations. Around the median, we have an equal number of values above and below, forming the interquartile range upon which we can consider an associated probability of occurrence.

For instance, Table 3.5 shows the computational cost (time of execution)

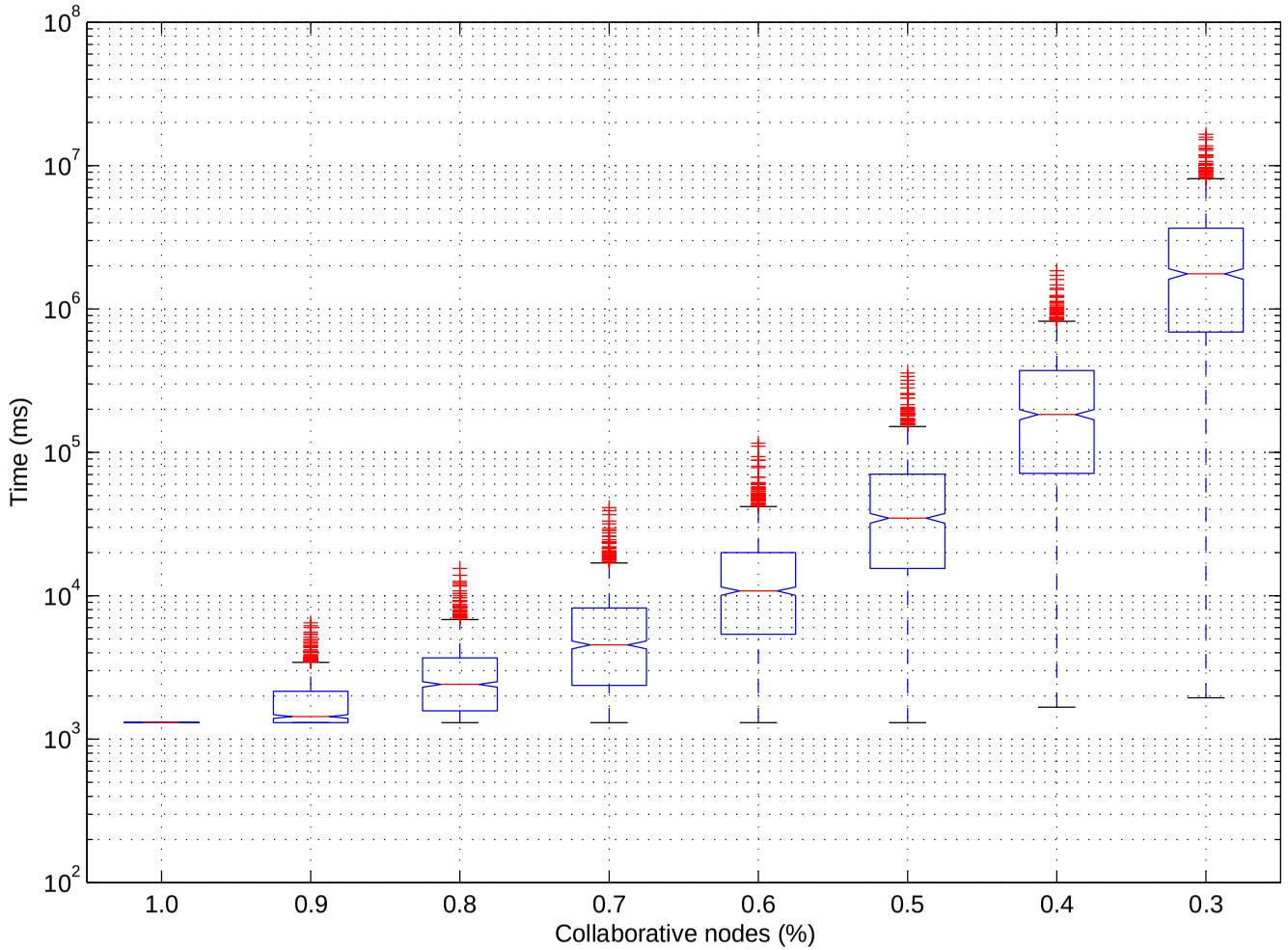


Table 3.5: Distribution of the time of execution for 1MB contents, 10 participants, and different percentages of collaborative nodes in the community: Worst case.

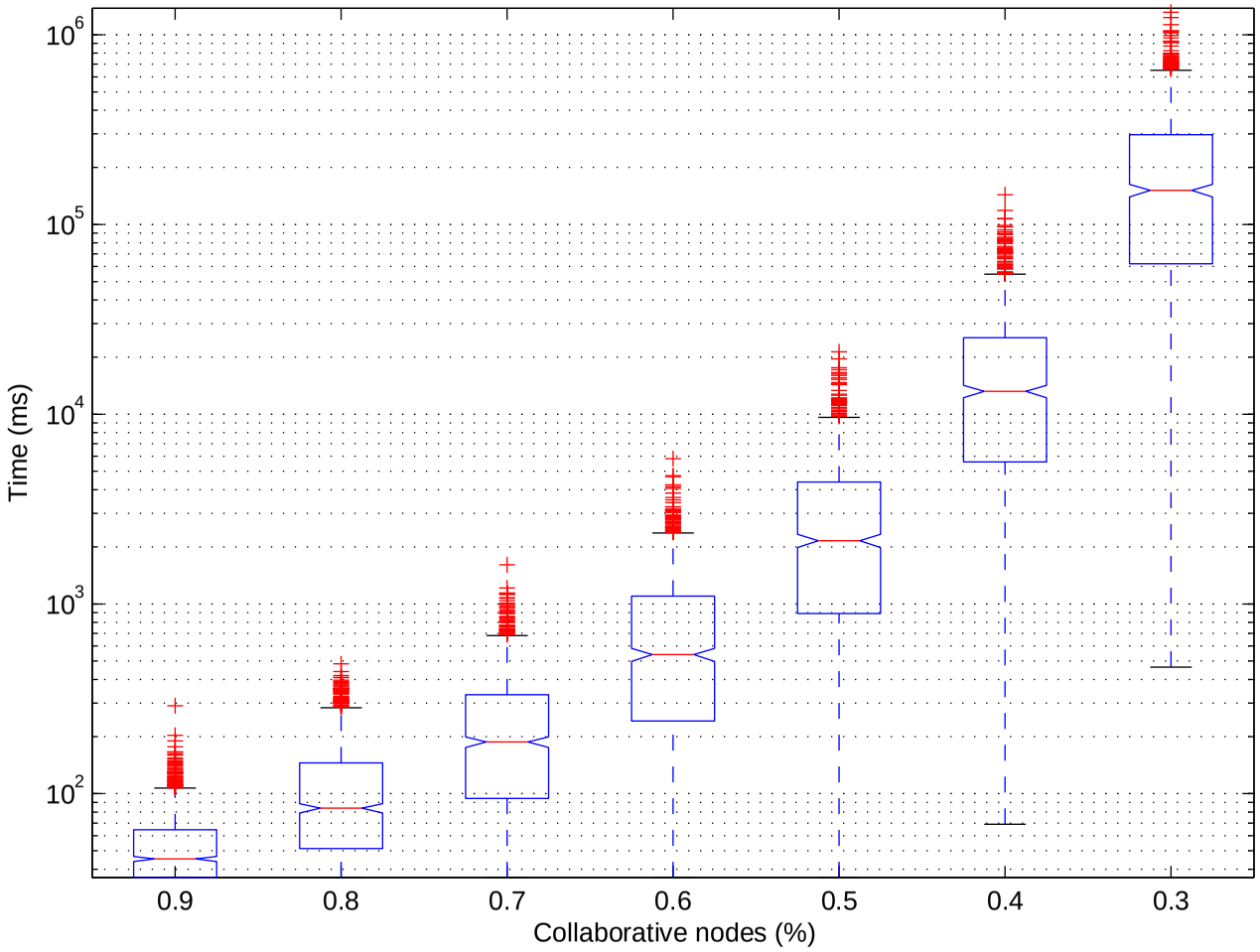


Table 3.6: Distribution of the time of execution for 1MB contents, 10 participants, and different percentages of collaborative nodes in the community: Best case.

distribution for the 1000 experiments run in each proportion of collaborative nodes in the community (from 100% to 30% of collaborative nodes) for 1MB contents, and 10 participants taking into account the complexity of the public key authentication. We can appreciate a "mild" variability of the simulated data, indicated by wider boxes. However, the 90% of collaboration gives us guarantees for, at worst, the cost of creating a 1MB-content certificate takes no more than 2 seconds with the cooperation of 10 signers which are unknown among them. On the other hand, though the unrealistic 40% of collaborative nodes also gives a less probable time of 2 seconds, the vast majority of the simulations ranges from 1.3 to 10 minutes, at worst.

Similarly, Table 3.6 shows the same scenario at best cases, i.e. when all participants are known and therefore there is no need to contemplate the complexity given by Pathak and Iftode's protocol. The best among them constitute certainly an interesting result. Comparatively, the 90% of collaboration gives almost all computational cost values of less than 0.1 seconds, while the 40% less than 1 minute.

Several social-inspired approaches have run similar experiments in order to demonstrate that it is possible to maintain reasonably cooperation level not only among nodes perceived as being of a cooperative type, but even with unknown or possibly deviant nodes, although only with a certain probability [48].

3.6.1.2 Availability of Signers

The failure of a node/network link can be a serious problem as the proposed content authentication scheme depends on that network component for its execution. Particularly, faulty (maliciously or not) nodes incur additional concerns in content certificate generation and verification stages. In the former, unavailable signers imply new instances of the signing phase initiated by the owner, incrementing the whole overhead. On the other hand, in the latter, at worst, a certain requester B, after a successful download, knows none of the signers' identities, having to run Pathak and Iftode's protocol. Faulty signers (included in the OLS) do not allow honest requesters to correctly verify a successfully downloaded content certificate.

Many works have so far addressed availability for predicting per-node resource burdens and selecting appropriate data replication strategies. Some

techniques basically lead to enhance routing schemes to route around faulty nodes or use multiple pass through the networks in order to avoid faulty networking elements, among others. However, it is not realistic to use just any of them in pure P2P networks due to the highly dynamism of the node community. Some research efforts concentrate on providing unreliable fault detectors, which may suspect a correct "process" or it may fail to suspect a faulty "process".

Although it is out of the scope of this proposal, we sketch some key ideas to deal with the discussion above. Let B be an honest requester. Since signers' signatures are chained in the content certificate, B should verify them in the order established by the OLS. If any signer n_i is (temporarily or persistently) unavailable, there is no way to authenticate its public key by means of the protocol adopted. A possible solution to this problem is based on the use of others' repository, and on reaching a consensus. Let k' be the amount of nodes $\{n_1, n_2, \ldots, n_{k'}\} \in T_B$ selected by B among his group of previously contacted nodes. B may securely ask them to provide him with a local copy of the required key K_{n_i} from their own database. Thus, B should (securely) receive k' responses, as follows:

n_1	n_2	$\cdots$	$n_{k'}$
K_{n_i}	K_{n_i}	$\cdots$	K_{n_i}

Whenever the selected k' participants are honest, available, and have at their disposal the requested key, then there will be consensus on the accuracy of this particular public key.

Obviously, this process implies an additional effort to take in conjunction with the computational and communication overhead already measured. Nevertheless, note that public key authentication must be executed just once by each node, and for each key in the network.

3.6.1.3 Determining k

This is a challenging problem, especially in a fully distributed, decentralized and dynamic system. A trivial way to solve it is by assuming or imposing a certain interval which limits the maximum and minimum number of signers according to the current community size. However, the highly transient community makes this implementation unrealistic in such environments.

Applying the same ideas to the Byzantine type of failure [17], it is shown that $n > 3t+1$ (where t nodes could be faulty and behave in arbitrary manners, and n is the total number of nodes) must hold even in the case of only needing to reach approximate agreement. As we showed before, the expected number of rounds to reach agreement may be exponential depending on the proportion of non-collaborative/faulty signers. However, some early works presented so far propose several theorems, based on probability, which guarantee the following [19]:

1. As long as a majority of the processes continues to operate, a decision will be made.

2. If the number of faulty nodes is $O(\sqrt{N})$ then the expected number of rounds to reach agreement is constant.

3. Moreover, if t satisfies $5t < n$, then completely asynchronous agreement is possible.

On the other hand, we can apply similar approach, but backwards, as computing the average time of execution in order to estimate an appropriate number of signers according to the time a particular owner is willing to spend, i.e. how many signers the owner has to collet if she wants to spend no more than a certain period of time. We simulate, as in previous sections, different percentages of collaborative nodes in the community, and consider the best and worst case according to the requirement of instancing the public key authentication protocol. Table 3.7 presents these results for 0.5, 1 and 30 seconds.

Note that the best case, in which there is no need to instance Pathak and Iftode's protocol, the number of nodes can be significantly higher, especially when having at one's disposal more than 30 seconds, e.g. we can use up to 400 nodes for generating a 10MB-content certificate in 30 seconds with mutually known signers immersed in a 60% collaborative community. On the other hand, we can consider 0.5 and 1 seconds as excellent lower bounds to generate 10MB(or less)-content certificates by collaborative communities and team-working. Moreover, 30 seconds may be a good mark to apply in common file sharing systems, even in the worst case.

% of collaboration	Acceptable Time Threshold					
	0.5 s		1 s		30 s	
	1MB	10MB	1MB	10MB	1MB	10MB
100%	[81, 18]	[16, 11]	[127, 25]	[33, 18]	[832, 118]	[578, 112]
90%	[75, 15]	[14, 8]	[118, 12]	[30, 11]	[786, 30]	[536, 30]
80%	[70, 14]	[13, 7]	[109, 12]	[27, 10]	[740, 29]	[493, 29]
70%	[63, 13]	[12, 7]	[101, 10]	[23, 9]	[689, 28]	[449, 28]
60%	[57, 12]	[11, 6]	[92, 10]	[21, 9]	[635, 27]	[400, 26]
50%	[50, 11]	[9, 6]	[82, 9]	[18, 8]	[578, 25]	[350, 25]
40%	[43, 9]	[7, 5]	[70, 9]	[14, 7]	[512, 23]	[294, 23]
30%	[34, 7]	[5, 4]	[58, 8]	[10, 6]	[440, 21]	[235, 21]
20%	[25, 5]	[4, 3]	[44, 7]	[7, 5]	[352, 19]	[168, 19]

Table 3.7: Determining k in the [best, worst] cases.

3.6.1.4 Memory Consumption

The certificates involved in the proposal are small enough to be stored even on nodes with low storing capabilities (e.g. mobile devices.) The size of a content certificate is around 600 bytes, depending on the number k of signatures and the cryptographic signing algorithm chosen.

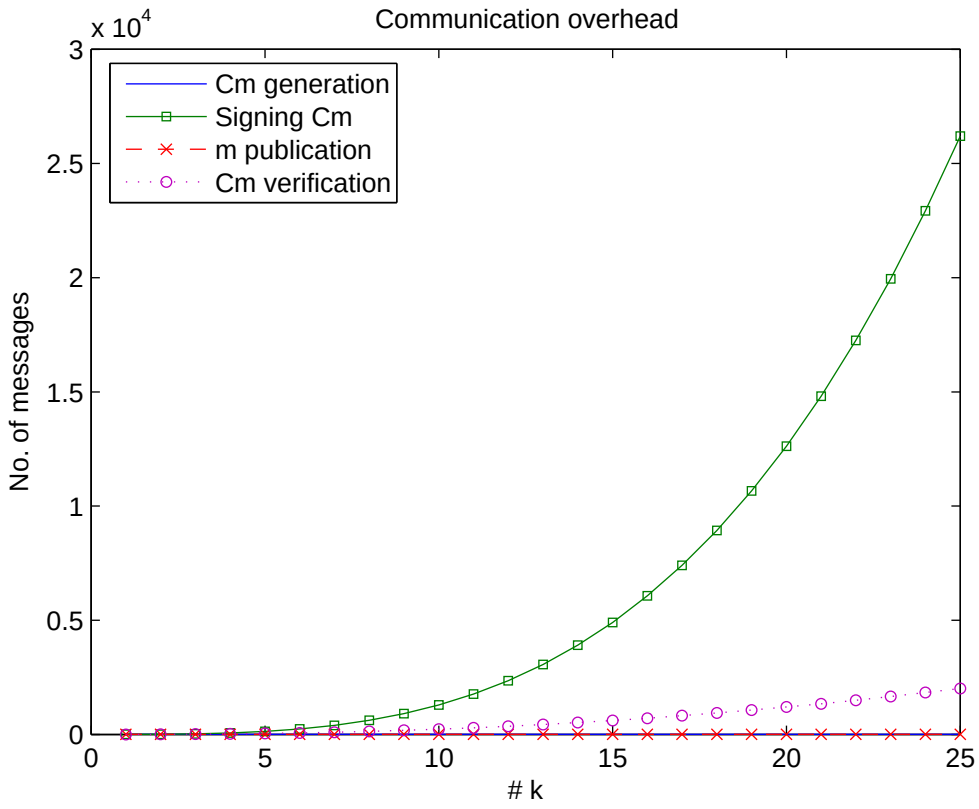
Content providers and signing nodes must maintain a number of local databases (tables T_i and S_i.) Their length greatly depends on the dynamics of the network and the extent to what the node is involved. However, it is reasonable to assume that a peer who offers a large number of contents also has at her disposal a good amount of computational resources.

3.6.2 Communication Overhead

We analyze efficiency according to the communication overhead imposed by the proposal. Table 3.8-(a) presents the number of messages and the complexity in terms of communication transmission for each protocol stage. For example, the content certificate verification process increases slowly as $\mathcal{O}(k^2 \log k)$ compared to the complexity of the signing stage. $\mathcal{O}(k^3 \log k)$. The cubic factor results from the possibility of, at worst, all the signers had to execute Pathak and Iftode's protocol, $\mathcal{O}(k \log k)$. Just like similar protocols implementing distributed authentication, this public key authentication protocol is expensive in messaging cost. However, authors, in their analysis, try to reduce this cost

Stage	No. messages	Complexity
1. C_m generation	$1+1$	$\mathcal{O}(k^3 \log k)$
2. Signature process	$2k + \frac{k(k+1)}{2} PI$	
3. Content publishing	$1+1$	
Subtotal:	$2k + 4 + \frac{k(k+1)}{2} PI$	
4. B's Checking C_m	kPI	

(a) Complexity of communication for each stage of content authentication.



(b) Communication cost (number of messages transmitted in the worst case) for each stage.

Table 3.8: Efficiency analysis (communication overhead).

using an epidemic algorithm called *Public key infection* for lazy propagation of protocol messages. By means of this improvement, the complexity gets reduce to $\mathcal{O}(k \log k)$. The related analysis is included in Appendix A. As an example, let k be the number of signers and also the number of participants

in the public key authentication. These signers have to perform the following set of messages, at worst i.e. they do not know previous signers, for signing the content certificate:

$$1 + 1k \log k + 2k \log k + 3k \log k + \cdots + (k-1)k \log k = \frac{k(k+1)}{2} k \log k$$

On the other hand, certificate generation and content publishing remind as a constant value.

Figure 3.8-(b) plots how the number of transmitted messages increases depending on the number of signers involved in each stage. At worst, the entire content authentication protocol requires a high number of transmissions, e.g. 10 signers have to exchange at worst 1000 messages. The communication overhead actually depends on the specific content certificate's length and the complexity of the challenges in Pathak and Iftode's messages. On the other hand, in the best case where all signers are mutually authenticated before, the number of messages decreases significatively to 19 messages, using 10 signers.

3.7 Security Analysis

In this section, we provide an informal analysis about the security of the proposed scheme. For this purpose, we discuss several attack scenarios and forms of malicious behavior which can occur during each phase of the protocol execution.

3.7.1 Eavesdropping

An attacker can listen the messages transmitted during the content certificate generation, for all the communication between A and $n_i \in OLS$, since these messages are not protected neither by any network/transport layer nor any other bootstrapping phase allowing parties to mutually authenticate and establish a session key under which all the communication is encrypted.

3.7.2 Message Modification Attacks

An attacker can try to modify some messages with the hope of modifying any message or field in the content certificate. As in the previous case, there is not secure channel established between both parties prevent this situation to occur. However, such a modification can be detected due to it leads to an incorrect message authentication in the verification stages of proposal, but never to any gains for the attacker.

3.7.3 Message Reply Attacks

An attacker first listens the messages exchanged between A and any other participant and then tries to reproduce the session by using some of them. The protocol is robust against such a situation due to the identities of both parties are included in the messages, not being useful for anyone different from them.

3.7.4 Message Insertion Attacks

An attacker can try to generate fake messages and insert them in the channel. However, she is unable to do that for the same reason previously described for the case of *message modification attack*.

3.7.5 Message Dropping Attacks

Given enough control over the network infrastructure, an attacker can try to delete some of the messages exchanged. By doing so, the only result achieved is a failure in the correct execution of the protocol (which can be viewed as a denial of service), but cannot enable the attacker to gain any useful information.

3.7.6 Impersonation Attacks

Assume that an attacker can success in hijacking a session between any pair of participants. Even in this case, messages exchanged among trusted peers are safe from spoofing, for they are signed by authenticated public keys. In other words, the attacker cannot generate the correct digital signatures. Note

that this is a particular case of trying to exhibit malicious behavior against the other party.

3.7.7 Attacks Against the Public Key Authentication Process

At some stages of the protocol, both parties are required to authenticate the other's public key in order to verify the signatures exchanged. However, Pathak and Iftode's protocol can fail due to the impossibility of getting an honest majority. In this case, an adversary might convince a honest peer that the public key of a node is K'_{n_i} when it is not. This kind of man-in-the-middle attack is detected if at least one peer (apart from A) is honest.

3.7.8 Assurance of Content Authenticity

The protocol is started by A and ultimately relies on her honesty. In case of A being honest, we can assume that the original content m released by her is authentic. We also assume that the hash function cannot be manipulated, and that the OLS contains A's trusted members. In this case, the initial content certificate, C_0, is correct. It is straightforward to see that any modification on the certificate performed by a signing node can be easily detected.

The originator might try to exhibit a malicious behavior. Any modification of the certificate fields will be eventually detected by, at least, one node in the OLS, since it is assumed honest majority in the signing community and, therefore, A cannot collude with a sufficient number of traitors. This also prevents a group of malicious nodes to collude during the signing process in order to forge a false certificate.

3.7.9 Modified Replicas

Consider the following scenario, where B gets access to m and its associated certificate C_m, and tries to generate a new, fake certificate C'_m, in which B appears as the legitimate owner of m:

`Certificate` C':
`Originator:` B
`ID:` I_m
`Contents:` $h(m)$
`OLS:` $B, n_{1'}, \ldots, n_{k'}$
`Validity Period:` (ts_1, ts_2)
`Signing Algorithm:` $AlgorithmDesc$
`Signatures:`
$E_k = E_{K^{-1}_{n'_k}}(\cdots(E_{K^{-1}_{n'_1}}(E_{K^{-1}_B}(h(C')))))$

This will be detected by a subset of nodes in the OLS during the local verification stage, as at least one of them has previously signed C_m and will refuse to cooperate.

Note that this scheme does not prevent from B modifying m into m', changing the identifier I_m into I'_m, and subsequently executing another protocol instance with the aim of obtaining a different content certificate, C'_m. This kind of manipulation cannot be avoided exclusively by external means, but by inserting appropriate mechanisms *inside* m.

3.7.10 Incorrect Sending and No Cooperation

Considering off-line periods, although our scheme provides efficient and secure content replication, but is lenient toward non-collaborative and faulty peers: a proportion of signers must cooperate in content integrity process at each time. Cooperation is then crucial during the content certificate generation stage. A dishonest signer can delay the content authentication by not signing the certificate, not sending any message to the next node in the OLS, and/or not notifying anything to the originator A. Surely, this behavior will produce a kind of DoS attack [81], so A must check nodes' availability when this situation appears.

A dishonest signer could send its participation randomly or maliciously, instead of a properly constructed signature. This is indeed a point of failure of the proposed scheme, even though this fact can be trivially detected, since cooperation is required among the $k+1$ nodes to achieve a successful execution of the protocol. In other words, the protocol actually *detects* these forms of misbehavior and no cooperation, but it cannot enforce nodes to behave properly. Even though this is totally related with the decentralized nature of P2P systems, nodes can be provided with incentives for cooperation in the way

Attack	Vulnerable	Comments
Eavesdropping	Yes	But not serious
Modification	No	This attack is detected
Reply	No	This attack is easily detected
Insertion	No	This attack is detected
Dropping	No	A type of DoS can be mounted
Impersonation	No	This attack is detected
Against PKI	No	Requirement: Honest majority
Authenticity	No	This attack is detected
Modif. of replicas	Yes	It is only detected
No cooperation	Yes	A type of DoS can be mounted
Collusion	No	Requirement: Honest majority

Table 3.9: Summary of the informal analysis about the security of the proposed scheme.

it is done in some current P2P systems for encourage file sharing. Nevertheless, this is something external to our proposal.

Each intermediate peer must ignore any messages that do not have the proper form of content and a signed certificate. Besides, peers know that the originator A is malicious if her signature is incorrect. However, rational parties would have no reason to deviate from the protocol.

3.7.11 A Note on Replication

After a successful execution of the protocol, a node B obtains a copy of m. This replica can be published by B, thus contributing to increase the availability of m throughout the network. For this, B cannot modify this replica, otherwise C_m would not be valid. Note that by doing this B is not claiming to be the originator of m; information about the true originator is enclosed within the signatures.

On the other hand, we could be interested in controlling every transaction (desirable condition in a collaborative working environment.) Without a clear data ownership, consumption is infinite, and that is not a desired state in a mobile environment. Our protocol does not apply any capabilities delegation over contents among nodes.

Furthermore, our proposal alone cannot prevent m from being modified by B once she has got access to it. However, if B publishes a modified replica, any

requester will be able to detect it just by checking the associated signatures.

By way of illustration, we include Table 3.9 which summarizes the vulnerability of our protocol to the attacks analyzed.

3.8 Conclusions

Due to the very nature of P2P systems, it is usual that several copies of the same content exist at different network locations. Despite the advantages derived from replication, in general it is unrealistic to assume that every node will behave correctly. In this way, once a malicious node gets access to a content, its integrity can be compromised in several ways.

In this chapter, we have proposed a content authentication protocol based on Byzantine agreement, especially oriented to pure P2P systems. This scheme allows for a secure content replication among peers, which is able to detect if non-authorized alterations have been made on the published contents. Furthermore, our proposal does not rely on the use of a TTP, an assumption that would be totally impractical for decentralized P2P environments. The key features of our proposal are:

1. Integrity of content is assured: it is a feature currently demanded by users of P2P file sharing systems.

2. The absence of fixed infrastructure makes it suitable for ad hoc domains, such as MANETs.

3. It is fault tolerant, but depends on collaboration.

4. It is probably an excellent building block for considering security to control content replication/distribution in future research designs.

We have also analyzed the proposal in terms of the computational and communication overheads. We have provided brief summary of the results in tables and figures. Moreover, the security of the scheme have been informally analyzed, enumerating possible vulnerabilities and threats.

3.8.1 Including a Bootstrapping Phase

We have mentioned the need of modifying our scheme either using (very costly) asymmetrically encrypted messages (which avoid the threat of eavesdropping during content certificate signing process) or assuming the existence of a bootstrapping phase in the form of that provided by Transport Layer Security (TLS.) In summary, we require such a mechanism to transmit secret messages between two partners without introducing a secret dealer.

Some solutions to overcome the aforementioned problem have been proposed so far based either on the use of public key cryptography or on the provision of secret handshakes in a fully decentralized environment. We gather together some of them:

- Traditional approaches, e.g. the (unauthenticated) two-party Diffie-Hellman key agreement, are vulnerable to active and passive attacks in ad-hoc exchange.

- On the one hand, the majority of the proposals assume that the participants already have access to authenticated copies of each other's public keys. It is relatively simple then to establish secure and authenticated channels, given that these public keys have already been exchanged and, therefore, participants know who they are supposed to be talking to. Nevertheless, this involves a non-trivial cost.

- Furthermore, in MANETs and ad hoc wireless networks, the problem of secure communication and authentication without TTP has been addressed as well [15]. Evolutionary game theory also can be applied in various application context, especially in self-organized socially cooperative networks, aimed at encoded a social behavior into an automatic model, such as the automatic social bootstrapping service proposed in [60].

Our content authentication protocol may then adopt any of them previous to the certificate signing stage.

Chapter 4

Access Control in Pure P2P Networks

Despite the intense research activity that has been devoted to develop applications aimed at facilitating collaboration among multiple, distributed users, relatively few works have concentrated on controlling access to the collaboration environment and shared resources. Most collaborative systems give all participants the same rights to access all objects, and expect that access issues will be controlled by a social "protocol." Thus, they do not provide support for preventing mistakes, conflicting changes or unauthorized access.

Just like it is crucial to guarantee properties such as content authenticity in a P2P file sharing system, it is also interesting to enforce appropriate access control policies, which limit the activity of legitimate users while serve as an obstacle to dishonest requesters. In such a scenario, owners may determine if the requester attempting to download a content is actually authorized to access it. On the other hand, ensuring that each one of these new potential providers will behave according to the owner's access policy gets more complicated (and definitely hard should a global security infrastructure be unavailable.) Thus, a secure replication scheme protects distributed replicas from dishonest activities, assuring basic security properties such as content authenticity and integrity, as well as to protect the unauthorized use of the resource.

Many inherent characteristics of fully distributed and collaborative environments introduce new requirements for access control, such as different applications running on each node, control decentralization, and off-line working, to name a few. Mobility (e.g. by using wireless ad hoc networks) adds additional

concerns: support for mobile devices, the sporadic nature of connectivity, the dynamically changing topology or the low computational power of the devices, among others. These features pose a challenge from the point of view of the security mechanisms that should be applied [66].

The most accepted solution for authentication and authorization services in classic distributed environments relies on the existence of a PKI or, more generally, a Privilege Management Infrastructure (PMI), since PKIs are not appropriate to provide authorization services. However, the main features offered by most P2P networks –replication and self-organization– involve some difficulties regarding the provision of these security services [27]. In general, the extremely decentralized nature of a pure P2P network makes impossible to apply solutions that rely on some kind of fixed infrastructure, such as on-line TTPs.

As a result, authorization in such environments cannot be generally provided by means of existing models, which were developed for centralized systems or, at most, for distributed systems in which the existence of on-line services is a prerequisite. In common distributed systems, this task is developed by issuing privileges to users by means of authorization certificates, which they must present to gain access to resources. However, in a chapter before, the idea of X.509v3 certificates is used for the protection of contents' integrity and prevent non-authorized manipulations in fully distributed systems. In this chapter, we show how those certificates can be also used for the establishment and transference of the privileges referring to a node. Briefly, our approach allows the content owners to define and enforce restrictions on how the content is used, and also provides a secure access to the content on the basis of proofs of computational effort. Both, contents and peers, are required to posses a certificate assuring properties such as integrity and authenticity (in case of contents), and privileges (in case of users.)

The rest of this chapter is organized as follows. Section 4.1 briefly overviews traditional approaches to implement authorization in practical systems, and some related work on P2P privilege management.

The description of our P2P access control approach is first introduced in Section 4.2 and finally detailed in Section 4.3.

Moreover, the proposal provides support for disconnected sources by including delegation of privileges, and also allows to revoke both authorization

and replication rights. Particularly challenging problems in our proposal are how to carry out revocation of certificates and delegation of privileges. In order to seize replication advantages such as high availability of shared resources, our proposal provides a support for disconnected sources, and deals with traitors and whitewashers. For this, a node (the delegated) will act eventually on behalf of another node (the delegating) with some of her privileges. Moreover, malicious actions may be punished by means of downgrading the misbehaved node's security label. The delegation and revocation models are presented in Section 4.4.

In Sections 4.5 and 4.6, we summarize the experimental results concerning the efficiency and security, respectively, of the access control and authorization services integrated in our scheme. Finally, Section 4.7 concludes the chapter and outlines some future research directions.

4.1 Overview of Authorization and Access Control Models

Research on security has developed so far a number of models to deal with access control solutions. The authorization process is used to protect the system resources, allowing those resources to be used by consumers that have been granted authority to use them.

4.1.1 Definitions and Concepts

This section introduces concepts, terms, and basic definitions that are commonly used in the access control community. Basically, these are the following [64]:

- **Object:** Any kind of system resource (e.g. files, directories, registers, process, printers, ...) Access to an object potentially implies access to the information it contains.
- **Subject:** Typically, users and/or process who initiate actions or operations on objects, and therefore cause information to flow among objects.
- **Operation:** Active process invoked by a subject (e.g. read, write, ...)

- **Permission or privilege:** Authorization to perform some operation. Generally, it refers to some combination of object and operation.

Furthermore, there are two main components in an access control system, as follows:

- **The policy.** The policy is a plan of action, like a high level guideline, usually a sequence of rules, that establishes how to determine the access decisions. The suitability of a given policy depends on the particular protection requirements for a given system, environment and users. On the other hand, policies are formally presented as **models**.

- **The mechanism.** Mechanisms implement models. Most mechanisms are based on the assumption of an authorization relation between objects and subjects. These actions are permitted or denied according to the authorization privileges established in the system, and expressed in terms of access rights [108]. We will briefly describe the most important approaches.

Generic access control models have been extensively studied both in collaborative and non-collaborative domains, providing the basic framework to describe protection systems such as classic access matrix models (subject-object-right) and reference monitors [6, 87]. In general, subject and object attributes can take on a wide variety of forms, e.g. sensitivity labels, types or lists.

Concretely, there are two basic types of classical access control models: discretionary and mandatory access control (DAC and MAC, respectively.) User rights and permissions are implemented differently in systems based on each of them. For example, in determining a subject's ability to perform read/write/execute operations on an object, following a MAC model, the access control mechanism should compare the object's attributes to those of the object based on a previously determined set of rights.

Next section provides additional information on access control policies and gives examples of several types of policies.

4.1.2 Discretionary Access Control

Discretionary protection policies govern the access of users to the contents on the basis of the user's identity and the authorizations she possesses. DAC allows an object's owner to determine the access rules for that object. It is generally used to limit a user's access to a file, being the owner who controls other users' access to the file.

Discretionary methods must deal with two main drawbacks, namely:

1. The real flow of information in a system [109], and therefore the possibility of transitivity. While yielding more flexibility than MAC systems, DAC loses the ability to provide provable security to resources. A user who is able to read data can pass it to other non-authorized users. This dissemination of data is controlled in mandatory systems through a lattice of security labels [107].

2. Malware, software designed to infiltrate or damage a computer system can inherit the identity of the invoking user, without his informed consent. This is the so-called problem of the Trojan horse.

On the other hand, two main data structures have been chosen to represent the access matrix: capability lists and access control lists. The first stores the matrix by rows, i.e. each subject is associated with a list of pairs (object, rights) called capabilities. The second approach stores the matrix by columns, i.e., each object is associated with a list of pairs (subject, rights), an access control list (ACL.) The latter is the most common mechanism for implementing DAC policies.

However, models based on ACLs present some drawbacks when applied in collaborative environments, mainly due to the impossibility of relating access rights to contents, attributes of resources, or any other contextual information. It is difficult to determine all privileges for a user, i.e. one would have to search all of the ACLs. Furthermore, these approaches lack the ability to support dynamic changes of access clearances, and also lack of portability. This is especially critical for a large system where a flexible and constantly changing access control policy is required. For teamwork applications, for example, some collaborative frameworks use roles in conjunction with ACLs, such as SUITE and Intermezzo frameworks [46].

Other well known DAC control scheme is that used in UNIX-like operating systems by means of protection bits. These bits specify the permissions of read/write/execute for different classes of subjects: the owner, groups and the rest. Some limitations regarding the impossibility of specifying separate permissions lead to significant vulnerabilities such as the violation of the principle of "least privilege", which states that each user should have the minimum set of privileges needed.

4.1.3 Mandatory Access Control

MAC policy decisions are made by a central authority, and, therefore, users are not able to change or reassign access rights. The security level related to a user, also denominated *clearance*, determines the user's trustworthiness not to disclose sensitive data to others not cleared to access it. Moreover, objects in the system are assigned a security label as well. There is, therefore, a strict relationship between the security levels associated with the two, object and subject.

Moreover, MAC can also be applied for the protection of data integrity and confidentiality by preventing information stored in low objects to flow to high objects, and vice versa, though data integrity in general requires additional techniques.

However, in MAC, we need a security administrator who establishes the security classification of subjects and objects. This actor assigns security levels to users, can change these levels, similar to the levels for resources will be determined by the system in accordance with the clearances of the users creating them. Thus, the protection of such label and the access control decision logic from corruption are MAC-critical issues and require appropriate robustness. DAC and RBAC models, on the contrary, allow several environmental alternatives [108].

4.1.4 Role-based Access Control

Role-based access control (RBAC) mechanisms were proposed as an alternative to the models based on users' identities. In RBAC policies, permissions are assigned to roles (based on job competencies or responsibilities) rather than to individual users [46]. RBAC simplifies the administration and management

of permissions, e.g. user membership into roles can be easily revoked.

In the case of collaborative environments, it is insufficient to have role permissions based on object types. Moreover, RBAC is difficult to use for supporting a DAC policy, due to the large number of resources it would require.

4.1.5 XML-based Access Control

A number of research efforts have led to the development of Extensive Markup Language (XML)-based frameworks which unify basic access control specification. Several standards are currently maintained by different corporations such as the Organization for the Advancement of Structured Information Standards (OASIS.) The three most important advances are Security Assertion Markup Language (SAML), eXtensible Access Control Markup Language (XACML), and XML Digital Signature Recommendation (XMLDSig.) These languages define the representation of authorization information, and also provides general-purpose protocols for expressing access control policies. In particular, the request and response formats represent a standard interface for XACML-based access control systems, such as `Cardea` [72]. For further details refer to [78].

4.1.6 Centralized vs. Distributed Implementation

It is possible to combine ACLs and capabilities. In particular, this approach involves advantages especially in distributed systems due to the fact that repeated authentication of the subject is not required. Note that access control relies on and coexists with other security services, particularly with authentication. Interested readers can find further details in [121].

There is not, therefore, an strict classification which identifies the most suitable implementation for a certain environment, and no classification defines the space of possible distributed implementations either [68]. There are, however, some principles, which are built on the basic access control mechanisms, well suited for most of the distributed system environment: user group, access rules and centralized control. The use of delegation in administration provides flexibility and scalability, as well as reduces the need of control.

A distributed access control model is addressed in [77] through the idea of authentication and authorization infrastructures (AAI.) An AAI is the most

significative evolution of PKIs, and may be seen as the result of the union between PKI and PMI. ITU-T proposal defines four PMI models according to the application: general, control, role and delegation. An interesting point is that AAIs provide a delegation procedure by which an owner delegates authorizations to another without being involved. However, there is some problems with the application of existing delegation mechanisms in pure P2P environments. For example, the delegate could masquerade herself as the delegator, or impersonate her, since there is no control on what others can and can not do. A possible solution would be that the delegate should act in his own name, not in hers.

Some cryptographic-based mechanisms have been suggested to solve the problem of content distribution, such as Broadcast Encryption [54]. This is a cryptographic technique for implementing compliant authorized domains, and can be used as a replacement for public key cryptography in certain applications.

4.1.7 P2P Privilege Management

Just like the existence of an access control system is essential in a distributed environment, equally important is the question of how this information access security can be managed. Distributed systems require a medium for transferring access control information concerning a node with a level of self-administration. Traditionally, this can be achieved through the use of digital certificates basically containing the identity (and other information) of their owners. The most well-known proposals covering this functionality are X.509v3 and attribute certificates (an overview is presented in previous chapters), which are digitally signed by a CA. Concretely, an X.509v3 certificate except for the core fields (issuer, license, public keys, etc.), it also contains some fields for extension [119]. Thus, we can use the extension fields to store other service-dependant data, e.g. all of the needed access control information (including the delegation and revocation information.)

4.1.7.1 Delegation

Propagation of access rights in decentralized collaborative systems presents difficult problems for traditional access mechanisms, where authorization de-

cisions are made based on the identity of the resource requester. Unfortunately, there is a problem in the application of existing delegation mechanisms in pure P2P environments, for access control based on identity may be ineffective when the requester is unknown to the resource owner. For example, the delegated node could masquerade herself as the delegating user, or impersonate her, since there is no control on what others can do. Further works have addressed delegation services in a fully distributed environment, such as the complete framework for role-based delegation addressed in [129], where all behaviors are governed by a set of explicit rules.

4.1.7.2 Revocation

Revocation is an important process that generally must accompany delegation, i.e. a system that supports delegation of rights, should also support their revocation. It refers to the procedure for downgrading or eliminating the delegated node's privileges. There are several revoking schemes: strong and weak revocations, cascading and non-cascading revocations, among others [62]. Classic methods based on distributing certificate revocation lists (CRLs), certificate validation protocols (SCVP) or online certificate status protocol (OCSP) may not be applicable for dynamic environments in which there are no centralized repositories. Revocation is always a difficult problem. This problem becomes more troublesome in modern distributed systems, since the access decision engine is distributed among autonomous users for which there exists no centralized authority.

4.1.8 Trust-based Authorization Services in P2P Networks

Many algorithms have been proposed to compute the trustworthiness that a node has over others based on their past interactions. A key aspect of trust management is delegation, i.e. the transference of limited authority over one or more own resources to others. For example, AAI provides a delegation procedure by which an owner delegates authorizations to another without being involved.

Moreover, the work introduced in [74] presents trust management as an approach for RBAC credential delegation in decentralized distributed systems.

A recent work, presented in [14], addresses the issue of certificate revocation without input from external entities, employing threshold cryptography and profile tables. Nevertheless, each node must possess the public keys of the CA that issued the other node's certificates. Among the data collected in the profile tables, the number of accusations launched against the nodes is used to determine whether or not a given certificate should be revoked. In this regard, maintaining a high trust rating can be used as an incentive to reduce the degree of selfishness or altruistic behavior of peers [131].

Concerning P2P file sharing, [122] proposes a trust-based access control framework extending the DAC model. The proposed reputation model maps two dimensions into a *rating certificate*: trust (a proportion of satisfied transactions) and contribution (measured in megabytes.) The problem arises when a source encounters a requester she has never met, especially in MANETs. For this reason, we have tried to avoid the use of trust-based scoring models to differentiate peer behavior since they are vulnerable to several attacks, such as collusion. Moreover, this framework assumes the existence of a *host* who classifies users, assigns each user different access rights, and authenticates nodes. This does not preserve the P2P decentralized structure.

4.2 Overview of our Access Control Approach

We extend our previous work by showing how digital certificates can be modified to provide authorization capabilities. Essentially, both contents and peers will posses a certificate ensuring properties such as integrity and authenticity (in case of contents), and authorizations (in case of users.) We also shall explore further extensions which help to reduce the effectiveness of dishonest behavior by means of a content access procedure based on proofs of computational effort. In particular, sharing can be encouraged by imposing a cost on the downloads (e.g. resolving a cryptographic puzzle.)

By balancing the main features of collaboration and security, our solution establishes a discretionary access control scheme for pure P2P networks, providing at the same time the ability of detecting non-authorized content modifications. For this, our protocol is based on the use of authorization certificates. This kind of user certificate is based on an attribute certificate: a digitally signed electronic document ensuring that its holder has been given a

particular set of attributes by the issuer.

Roughly, the basic idea that underlies to our approach is the following. Each peer classifies her contents according to several security labels. Labels are ordered, for example as in a lattice-based access control model. An authorized peer will be able to get access to a given content if her security clearance is higher or equal than the content's label. These security clearances, which take the form of attributes in public key certificates, can be discretionally issued and signed by the content provider. Apart from this, our scheme uses a cryptographic proof-of-work mechanism to discourage selfish behavior and to reward cooperation, as well as maintaining a reputation-based classification over the community. We shall describe how nodes interact following our proposed scheme in a generic P2P file sharing system. Furthermore, we analyze some security aspects of the protocol itself, and also present an efficiency analysis considering both the computational (especially cryptographic) effort required by the nodes, as well as the communication cost of the scheme.

4.2.1 Extended Assumptions

Throughout this work, we will assume the following working (operational) hypotheses, which extend those assumptions enumerated in Section 3.2:

1. **Identification**. Every participant needs a certificate for accessing a desired content of a certain provider's directory. These certificates are discretionally issued by the content owner. Contents owners control the access to their contents by means of authorization certificates which include the requester clearance, L_i. To some extent, the clearance represents the trustworthiness of a holder (also called subscriber) not to disclose non-authorized contents. Furthermore, this clearance may be upgraded/downgraded or revoked after visible misbehavior.

   After several transactions with different providers, it is expected that a node will have a "portfolio" of authorization certificates.

   Contents owners control their contents integrity as well (i.e. that m has not been modified in an unauthorized way) by means of content authentication certificates.

2. **Authorized providers**. Owners also control their replicas located on other providers by means of replication privileges. Replication privileges

are essential for delegating the file distribution task and providing fault tolerance.

3. **File sharing system requirements**. We have established some requirements that a secure file sharing system should support: user clearances and content security labels. Our proposed system should allow users to infer the access rights locally. Thus, users should be able to specify access policies. As a result, it should allow users to take multiple security clearances simultaneously and change these credentials dynamically during different collaboration phases. The resulted file sharing system transmits contents in clear (i.e. not encrypted) but always accompanied by the corresponding content certificate; and, similarly, the system always presents the aforementioned credentials for requesters and providers before granting access.

4.3 Authorization Extensions

In this section, we first overview the full protocol operation by means of several scenarios. Subsequently, we will describe in detail each building block of our proposal: Join subprotocol, Content Authentication subprotocol, Content Access subprotocol, Delegation subprotocol, and Revocation subprotocol (see Figure 4.1.)

The entire scheme works as follows. Consider a common file sharing scenario wherein, for each transaction, node A, which provides the service (i.e. the content), is called the *provider*, while node B, which requests the content, is called the *requester*. We assume that B has at her disposal some kind of search mechanism to engage in a searching process aimed at locating contents. The usual situation is that, due to replication, the search engine returns a list of sources from which the content may be obtained.

In previous chapter, we showed how digital certificates can be used for P2P content authentication, and how a content certificate C_m will serve to guarantee the integrity of m. Now, we have to include in it additional components in order to provide authorization capabilities. Concretely, we need to incorporate the security clearance required to access m. The content certificate therefore shall include a field containing the security label of the content. We further explain how to assign it.

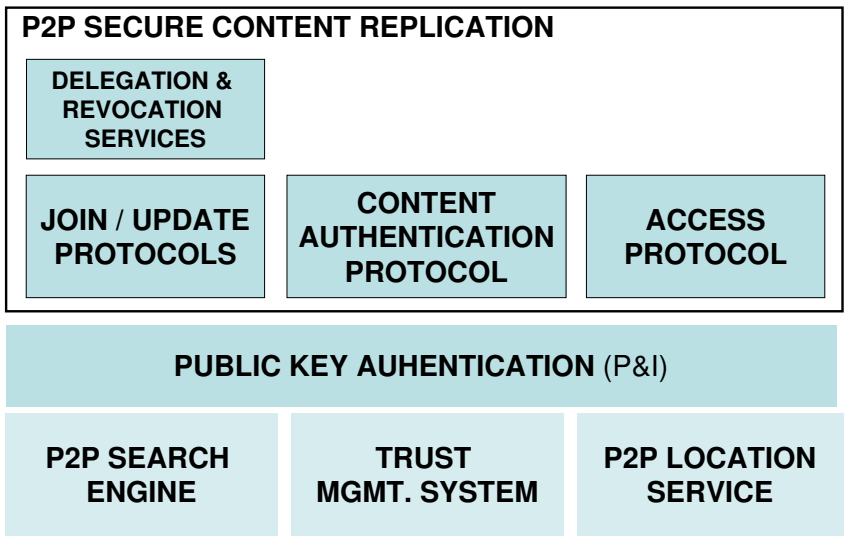


Figure 4.1: Main building blocks.

On the other hand, providers control access to their contents by means of authorization certificates, which include the requester clearance L_i. In most situations, this credential issued to new users will have the lowest security clearance. Briefly, we can identify three different possible scenarios after downloading the desired content. If user B has a security clearance issued by A and with level equal or higher than L_i, then B can access the content. On the contrary, if B's clearance is not high enough, B can engage in the updating process; the provider A can then update B's clearance or simply refuse the proposal. Finally, if B has no security clearance issued by A, she should initiate the Join subprotocol to ask for one. Furthermore, clearances can be revoked either when the validity period had expired or by a deliberate revocation.

In any of these scenarios, once B gets access to m, she must verify its authenticity by means of the content authentication subprotocol described in Chapter 3.

The high transient node community does not allow us to assume that any content owner will be highly available. In this case, an authorized delegated node will act on behalf of the delegating node with some replication rights. Delegation is one of the main mechanisms for privilege management in our distributed DAC model. Thus, with the replication privilege, an owner can delegate some of his authority to another node. In particular, owners assign

their providers a certain security label. Thus, a provider will be allowed to hold replicas of the contents of that specific level according to the owner's desire. Moreover, the replication right is contained within the authorization certificate structure which is issued by the corresponding owner.

In summary, our secure content distribution protocol is suitable for self-organized mobile ad hoc networks, and is similar to PGP in the sense that content and authorization certificates are issued by the users. However, as opposed to PGP, we do not rely on certificate directories for the distribution of certificates. Instead, in our model, certificates are stored and distributed directly by the users. Next, we introduce in detail each of the scheme subprotocols.

4.3.1 Join Subprotocol

This stage is initiated when a node B joins the community and desires a specific content m. Each owner classifies her contents according to several security classes, such as those described with a lattice-based access control policy [107].

Definition 4.3.1 (Access control policy [107])**.** A triple $< SC, \rightarrow, \oplus >$ where $SC = \{L_1, L_2, \ldots, L_n\}$ is a set of security labels, $\rightarrow \subseteq CS \times SC$ is an order relation on SC, and $\oplus : SC \times SC \rightarrow SC$ is a binary join operator on SC.

For example, a classic set of labels is "confidential", "restricted", "secret", and "top-secret", but can also be extended to support additional usage classifications depending on the owner's desires. B needs an authorization certificate issued by A to get access to m. Once B successfully obtains his authorization certificate benefits from it in the sense that the P2P file sharing system will recover m.

We describe the Join subprotocol more formally. The search engine will return a list of available nodes with their contents and replicas. At the first step, each user creates her public key K_B, and the corresponding private key K_B^{-1}, as some P2P file sharing systems already do [9]. Thus, B, who is still unknown by A, must send some information to A:

$$B \rightarrow A : m_1 = E_{K_A}(\langle B, A, RF \rangle), s_B(\langle B, A, RF \rangle)$$

where RF is a request form in which B formally expresses her desire to access A's contents, and $s_B(\cdot)$ is the signature provided by B. Upon reception, A must check B's signature. As in other cases, all messages are asymmetrically encrypted using the recipient's public key.

The situation is analogous to people who have several admission cards for accessing libraries, museums, clubs, etc. B will have an authorization certificate for each successfully contacted content provider. Nevertheless, the authorization certificates are not issued neither by the holder (the requester) nor a CA. Authorization certificates are discretionary issued by a content owner for granting access to a specific requester, with a limited validity period and a security clearance L_i. How owners evaluate requesters' intentions is discussed in Section 4.3.1.1.

In case of acceptance, A generates and sends an authorization certificate $C_B : \langle Serial, B, A, (ts_1, ts_2), L_i, AlgorithmDesc. \rangle$ for B:

$$A \rightarrow B : E_{K_B}(C_B), s_A(C_B)$$

where (ts_1, ts_2) is the validity period (i.e. the certificate is not valid before ts_1 and after ts_2) and L_i the security clearance provided. Upon reception, B will check A's signature. If this offer satisfies B, then she will return to A her signature on C_B:

$$B \rightarrow A : E_{K_A}(C_B), s_B(C_B)$$

Finally, after performing similar verifications, A creates B's authorization certificate as follows:

$$C_B^A : C_B, s_A\big(s_B(C_B)\big)$$

$$A \rightarrow B : E_{K_B}(C_B^A), s_A(C_B^A)$$

Summarizing, A generates a certificate C_B^A for node B, containing (see Figure 4.2):

- A certificate serial number, which univocally identifies a particular user authorization certificate.
- The identity of the authorization certificate owner (B), also called holder.

User authorization certificate $\mathbf{C_B^A}$

`Certificate` C_B:
`CertificateSerialNumber` *Serial*
`Holder:` B
`Issuer:` A
`Validity Period:` (ts_1, ts_2)
`Clearance:` L_i
`Signing Algorithm:` *AlgorithmDesc.*
`Signatures:`
$s_A\big(s_B(C_B)\big)$

Table of subscribers $\mathbf{Ss_{n_i}}$

Date	Requester	Clearance
(ts_1, ts_2)	n_{i_1}	L_{i_1}
(ts_1, ts_2)	n_{i_2}	L_{i_2}
$\vdots$	$\vdots$	$\vdots$

Figure 4.2: User authorization certificate and table of subscribers.

- The identity of the issuer (A), who establishes who has generated the content and is its legitime owner.
- Validity period (ts_1, ts_2) of C_B^A.
- B's security clearance L_i.
- Description of the hash and signature functions which have been used.
- Finally, the previous items are signed by the issuer A and the requester B (as shown in Fig. 4.3.)

Moreover, each provider must store members or "subscribers" and their clearances, as shown in Figure 4.2.

Since C_B^A includes the security clearance which allows B accessing some of A's contents, the receiver should have the ability to protect this clearance in order to prevent eavesdropping or even infiltrations of legitimate users. Using a simple encryption technique can be enough, but this is an issue which will not be addressed in this paper.

The full operation of the Join subprotocol is graphically depicted in Fig. 4.3.

4.3.1.1 On Challenges and Cryptographic Puzzles

As explained in section before, the main objective of the Join subprotocol is to allow owners control requesters by keeping track of requests while granting privileges to honest and collaborative requesters. Nevertheless, owners should be able to evaluate requesters' intentions just to prove, for example, that they

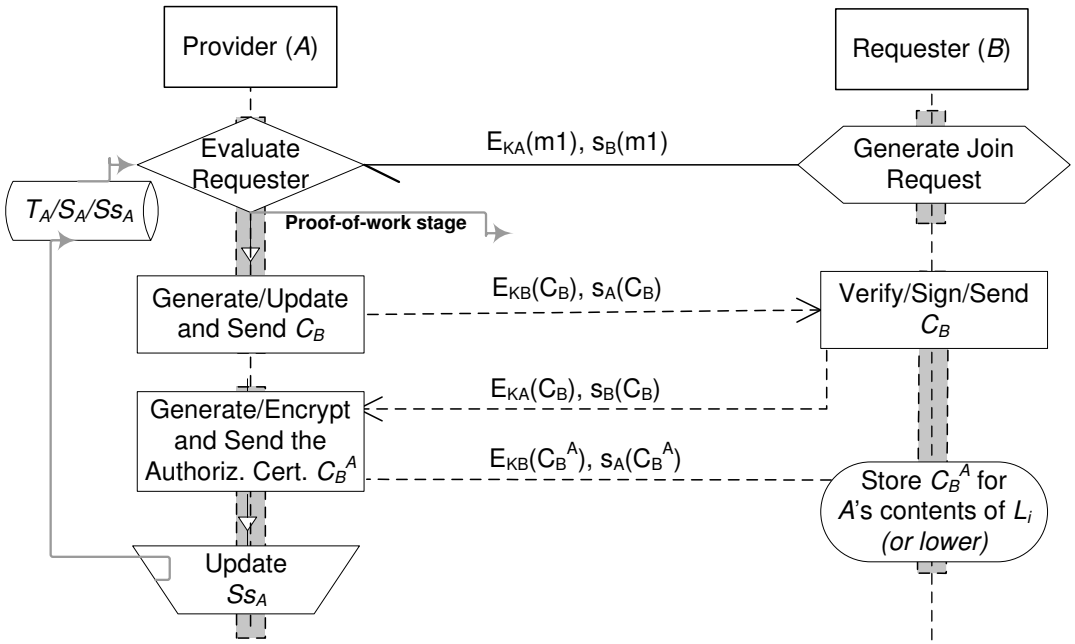


Figure 4.3: Join subprotocol.

are really interested in sharing. The key idea is simply asking each requester to invest an effort, e.g. computational, such as solving a puzzle, as follows.

Once owner A receives a request of join from node B (Fig. 4.4–message m_1), she computes a challenge ς_i and sends it to B (Fig. 4.4–message m_2.) Upon receiving it, B sends back to A the corresponding response τ_i (Fig. 4.4–message m_3.) If A considers τ_i as correct, both engage in the three-step subprotocol described in previous section (Fig. 4.4-messages $m_{4,5,6}$.)

Concerning the challenge ς_i itself, there exist a number of primitives which can be used for this purpose. The basic idea is that the verification by the challenger should be fast, but the computation by the requester has to be fairly slow.

A straightforward and efficient way of implementing such a construction is by using a block cipher (e.g. the AES standard.) The puzzle solution x is used as the plaintext to be encrypted, and the resulting ciphertext $AES_{K_S}(x)$ can be seen as a puzzle. Upon varying the AES encryption key length or revealing a subset out of the total number of bits of the key (i.e. a trapdoor), the effort required to recover x can be adjusted from very hard to a quite hard problem. Formally, A can challenge B using a AES encryption, as follows:

$$
\begin{array}{lll}
1.\ B \rightarrow A: & m_1 = E_{K_A}(\langle B, A, RF\rangle), \sigma_1 \\
2.\ A \rightarrow B: & m_2 = E_{K_B}(\varsigma_i), \sigma_2 \\
3.\ B \rightarrow A: & m_3 = E_{K_A}(\tau_i), \sigma_3 \\
4.\ A \rightarrow B: & m_4 = E_{K_B}(C_B), \sigma_4 \\
5.\ B \rightarrow A: & m_5 = E_{K_A}(C_B), \sigma_5 \\
6.\ A \rightarrow B: & m_6 = E_{K_B}(C_B^A), \sigma_6 \\
\\
\text{where} & \sigma_1 = s_B(\langle B, A, RF\rangle) \\
 & \sigma_2 = s_A(\varsigma_i) \\
 & \sigma_3 = s_B(\tau_i) \\
 & \sigma_4 = s_A(C_B) \\
 & \sigma_5 = s_B(C_B) \\
\text{and} & \sigma_6 = s_A(C_B^A)
\end{array}
$$

Figure 4.4: Proposed join scheme: Asking for an authorization certificate.

$$
\begin{array}{rl}
A \rightarrow B: & E_{K_B}(\varsigma_i), s_A(\varsigma_i) \\
B \rightarrow A: & E_{K_A}(\tau_i), s_B(\tau_i) \\
\text{where} & \varsigma_i = AES_{K_S}(file) \\
\text{and} & \tau_i = h(file)
\end{array}
$$

For instance, if a 256-bits key is used to encrypt challenge x and the user is provided with a trapdoor value that reveals 250 bits of the key, then she has to perform 2^{6-1} AES decryptions on average to find the correct response, x. On the other hand, if we want the party to devote more resources on the computation for the challenge, a lower informative trapdoor value should be used. Another example, by providing an AES encryption puzzle using no more than 32-bits keys, we force requesters to carry out around 2^{32} operations. Thus, the number of bits of the encryption key used as a challenge can be seen, on the one hand, as the difficulty parameter in solving the cryptographic puzzle, and on the other hand, as evaluating the security level for the clearance reached by the requester.

Another practical way to challenge requesters is by means of MHMBFs (moderately-hard memory-bound functions [7].) Roughly, $F()$ is a MHMB function whose domain and range are integers in $0 \ldots (2^n - 1)$, and we suppose that its inverse $F^{-1}()$ cannot be evaluated in less time than a memory access. The key idea is that $F()$ is chosen so that the verification by the challenger is fast, and so that the computation by the requester is fairly slow. In this way, A returns a fresh challenge ς to B. The latter should compute $F(\varsigma)$ and returns τ to A. Finally, A verifies that what it receives is a correct response to ς, and, if so, initiates the generation of the authorization certificate for

B, as shown before. We further elaborate on other examples for evaluating requesters' interest in the section bellow.

4.3.1.2 Updating Security Clearances

Either when a node initiates for the first time the Join subprotocol or when she wants to increase her clearance, the content provider should have some procedure for managing clearances. We sketch some ways for granting or upgrading a clearance, even though this is an external feature to our proposal and falls out of the scope of this paper:

- Using a Trust Management System (TMS) [10]. A trust metric will compute the eventual trustworthiness of a target based on a combination of the direct trust and her reputation. This two components are carefully weighted and the time factor (i.e. last transactions) is usually considered too. Trust information is shared between peers using a decentralized recommendation protocol. Moreover, the quantification of trust can be based on Fuzzy Logic [12].

  The TMS incurs a fixed cost of having to store and exchange all behavior feedback items to generate a reputation value for each requester. However, since a naive n_i will have no trust information for any unknown nodes in its local store, her trust value is initialized with a high distrust or very low trust value. Furthermore, these local values should be periodically updated after a new interaction result, aimed at dynamic trust changes.

  Essentially, higher clearances may be granted according to the past behavior of each peer. In this way, our scheme can be successfully integrated with a TMS, for providers can use it in order to classify peers.

  Trust measures may be applied on various stages of the entire protocol. Nodes use it for classifying other nodes according to the resulted weighted values. This classification makes easy some decisions. Moreover, this community classification also includes the association of a security label with the corresponding group.

- Using proofs of work. In this case, the cost of getting a clearance for any legitimate node B will depend on the time required for solving a puzzle

of a selected level from a provider A's list of challenges; as discussed in Section 4.3.1.1.

- Using a discretional policy. In this case, the content owner discretionary assigns clearances according to her personal criteria.

In any case, the subscriber B must previously send an update request to the issuer A, as follows:

$$B \rightarrow A : E_{K_A}(\langle UPD\_REQ, B, A, ts_0 \rangle), s_B(\langle UPD\_REQ, B, A, ts_0 \rangle)$$

Finally, after receiving a correct puzzle response, A will upgrade B's clearance and then both engage in Join subprotocol explained above.

4.3.2 Content Authentication Subprotocol

The main objective of this subprotocol is to maintain content integrity, ensuring its authenticity and avoiding non-authorized content alterations. We have exhaustively described and analyzed it in Chapter 3.

Nevertheless, checking of content authenticity cannot be done until content access subprotocol (detailed in next section) has been successfully finished.

For the readability and completeness of the entire proposal, we include a summary of the whole proposal in Figs. 4.5 and 4.6 using pseudocode.

4.3.3 Content Access Subprotocol

Usually, there are two ways for initiating file sharing: direct communication with a particular node or using the keyword search browser to look for sources and then contacting one of them. At the search stage, a requester R first performs a search query, which typically leads to a list of sources that keep a replica of the desired content.

First, the search engine will only show those available replicas stored by the owner A and the authorized providers P_i, who have an authorization certificate with the corresponding replication privileges issued by A. This certificate acts as a boarding card for the file sharing system aimed at controlling replication. The possibility of owners unavailability jeopardizes the most interesting feature of P2P file sharing systems, i.e. replication of contents. By means of our

Join

$B \rightarrow A : E_{K_A}(\langle B, A, RF\rangle), s_B(\langle B, A, RF\rangle)$
$A \rightarrow B : E_{K_B}(\varsigma_i), s_A(\varsigma_i)$
$B \rightarrow A : E_{K_A}(\tau_i), s_B(\tau_i)$
$A \rightarrow B : E_{K_B}(C_B), s_A(C_B)$
$B \rightarrow A : E_{K_A}(C_B), s_B(C_B)$
$A \rightarrow B : E_{K_B}(C_B^A), s_A(C_B^A)$
A updates Ss_A
B stores C_B^A

Update: Using puzzles

$B \rightarrow A : E_{K_A}(\langle UPD\_REQ, B, A, ts_0\rangle), s_B(\langle UPD\_REQ, B, A, ts_0\rangle)$
$A \rightarrow B : E_{K_B}(\varsigma_j), s_A(\varsigma_j)$
$B \rightarrow A : E_{K_A}(\tau_j), s_B(\tau_j)$
If A receives a correct challenge response:
A updates C_B^A with $L_{i+\uparrow}$
$A \rightarrow B : E_{K_B}(C_B), s_A(C_B)$
$B \rightarrow A : E_{K_A}(C_B), s_B(C_B)$
$A \rightarrow B : E_{K_B}(C_B^A), s_A(C_B^A)$
A updates Ss_A
B stores C_B^A

Content authentication

1. C_m generation:
   A generates $C =< A, I_m, h(m), OLS, (ts_1, ts_2), AlgorithmDesc. >$ and signs: $C_0 = \langle C, E_0\rangle$
2. Distributed signing process:
   A sends (m, C_0) to n_1
   For $i = 1$ to k
   (a) n_i performs the local verification stage on C_{i-1}
   (b) n_i adds its signature and generates $C_i = \langle C, E_i\rangle$
   (c) n_i updates T_i with the tuple $\langle ts_i, A, h(m), C_{i-1}, E_{K_{n_i}^{-1}}(E_{i-1})\rangle$
   (d) n_i sends (m, C_i) to $n_{i+1(mod\ k)}$
   (e) n_i sends a notification message to n_0
3. Content publishing:
   A publishes (m, C_m)

Figure 4.5: Summary of the proposed content authentication and access control scheme.

delegation subprotocol, an authorized provider can distribute original replicas without modifying neither contents nor their associated certificates. In these cases, when a requester selects a provider instead of the legitimate owner, the system must verify the provider's credential, i.e. $C_{P_i}^A$ has the corresponding

Content access

0. Search response: $P \rightarrow R : C_P^A$
   0.1. C_P^A is not revoked, i.e. its serial number is not in CRL_A
1. $R \rightarrow P : QUERY, C_R^A$
   Case 1: R has an appropriate C_R^A, i.e.:
   1.1. C_R^A is authentic, and
   1.2. C_R^A is not revoked, and
   1.3. $L_i \in C_R^A \geq L_m \in C_m$
   1.4. The system recovers (m, C_m)
   R uses C_R^A for getting m.
   Case 2: R has no valid authorization:
   (a) Initiate Join subprotocol, or
   (b) Try to update/increase her clearance, or
2. C_m verification:
   2.1. $h(m)$ comparison
   2.2. OLS's public keys verification (if not done before)
   2.3. Verification of C_m's chain of signatures

Delegation: On owner demands

1. Replication request:
   $A \rightarrow P : E_{K_P}(\langle REPLI\_REQ, A, P, ts_0 \rangle), s_A(\langle REPLI\_REQ, A, P, ts_0 \rangle)$
   P is already A's subscriber with the corresponding clearance L_i
2. P accepts:
   $P \rightarrow A : E_{K_A}(\langle REPLI\_ACP, A, P, ts_1 \rangle), s_{P_i}(\langle REPLI\_ACP, A, P, ts_1 \rangle)$
3. A includes in C_P^A the replication privilege:
   $A \rightarrow P : E_{K_P}(C_P), s_A(C_P)$
   $P \rightarrow A : E_{K_A}(C_P), s_P(C_P)$
   $A \rightarrow P : E_{K_P}(C_P^A), s_A(C_P^A)$

Revocation

0. Revocation decision by A
   C_B^A is now revoked
1. Update CRL_A with C_B^A's serial number and current time
2. Propagation to providers: $A \rightarrow P_i \subset Ss_A : E_{K_{P_i}}(CRL_A), s_A(CRL_A)$

Figure 4.6: (continued from Fig. 4.5) Summary of the proposed content authentication and access control scheme.

security level in the replication field and is not revoked (further details in Sections 4.4.2 and 4.4.3.) This process concludes with the following messages sent by available providers to the requester's system:

$$P_i \rightarrow R : C_{P_i}^A$$

As in previous chapter, a query result should return the content's descriptor, which could be necessary to rank the relevance of the resulted content to the query, as well as the list of identities of the sources. Note that successful past transactions with a certain owner mean that the requester R already has an authorization certificate to use some of the services offered by A. In fact, R will be able to obtain A's contents only by the possession of a valid C_R^A, particulary if her clearance is high enough. Specifically, R's file sharing system can use the information enclosed into her authorization certificate to get m, whether the following premises were fulfilled:

1. The requester R's authorization certificate issued by m's owner, C_R^A, is authentic. The provider (the owner A or an authorized node P) must check C_R^A's fields and signatures.

2. The certificate C_R^A is not revoked. The provider must validate the timestamp field upon A's CRL. Delegation subprotocol provides this feature (see next Section for details.)

3. The clearance $L_i \in C_R^A$ is greater or equal than m's security label, $L_m \in C_m$.

In case of R either not having any authorization issued by A or her clearance being lower than L_m, she can:

1. Initiate the Join subprotocol to obtain a valid authorization certificate from A, or

2. Try to update or increase her clearance (see Subsection 4.3.1.2.)

Together with a successful query result, R obtains C_m with all the information associated with the content and its owner. Once B gets the desired content, she must verify the identities of signers and the correctness of their chained signatures presented in C_m. For the sake of illustration, in Fig. 4.7 it is graphically shown the content access subprotocol.

Finally, a new generation of P2P middleware is still needed to support these security-specific requirements, as well as distributing and storing the objects automatically serialized together with the corresponding certificates.

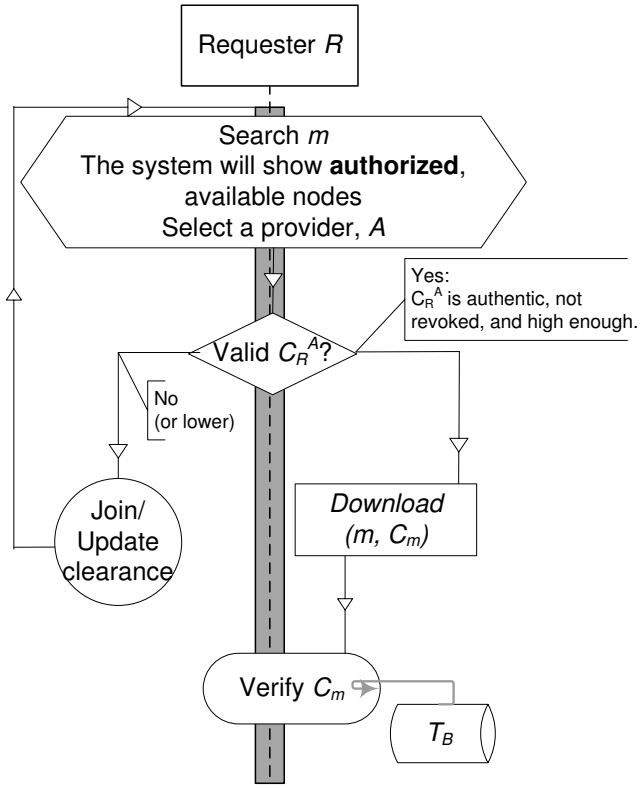


Figure 4.7: Content access subprotocol.

4.4 On the Provision of Privilege Management Services

In common file sharing systems, contents can be soon replicated through different locations, so assuming that each new potential provider will behave accordingly to the legitimate owner's policy gets definitely hard since there is neither a control mechanism on the replication process nor a global security infrastructure. A particularly challenging problem in content distribution is how to carry out delegation of replication privileges. The use of delegation provides flexibility and scalability, as well as a reduction of the need of high levels of availability.

Unfortunately, there is a problem in the application of existing delegation mechanisms in pure P2P environments: classic access control schemes based on identity may be ineffective when the requester is unknown to the resource owner. For example, the delegated node could masquerade herself as the del-

User authorization certificate C_B^A

Certificate C_B: CertificateSerialNumber $Serial$ Holder: B Issuer: A Validity Period: (ts_1, ts_2) Sign. Algorithm: $AlgorithmDesc.$
Privileges: Clearance: L_i Replication: $NULL \| \| L_0 \| \| \ldots \| \| L_n$
Signatures: $s_A(s_B(C_B))$

Figure 4.8: User authorization certificate: New fields.

egating user, or impersonate her, since there is no control on what others can do. However, there is not a strict classification which identifies the most suitable implementation for a certain environment, and no classification defines the space of possible distributed implementations either.

In this section, we present a secure mechanism for providing two additional authorization services: revocation and delegation, especially oriented to P2P file sharing systems. Most works done in this field agree with applying schemes based on X.509v3 certificates and authorization attributes, which they must present to gain access.

4.4.1 Extensions to the Authorization Certificate

As shown before, the authorization certificate indicates that some requester B is eligible for access a particular owner A's contents with a certain security level L_i. Now, owners can delegate some distribution rights over their contents. These rights are stored at the corresponding node certificate as attributes. Initially, when a particular node B requests a join subscription, the replication field is set as $NULL$ by default. Concretely, owner A generates and sends an authorization certificate C_B : $\langle Serial, B, A, (ts_1, ts_2), sign.AlgorDesc, L_i, NULL \rangle$ for B, as we explained in Join subprotocol. Figure 4.8 depicts these new fields.

Furthermore, the receiver should have the ability to protect this clearance in order to prevent eavesdropping or even infiltrations of legitimate users. Using adequately a simple encryption technique should be enough.

On the other hand, owners and authorized providers, after reaching the corresponding replication privilege, must maintain securely the information regarding subscribers and their clearances, and contents and their certificates as well for each owner. We further explain these data structures in next section.

4.4.2 Delegating Distribution Privileges

The delegation relation involves a set of three actors: the owner or delegating node, the delegated node or provider, and the associated constraints. We have identified two ways of operation according to who initiates the delegation request:

1. On providers offering. After a successful download, a requester B can choose to contribute on replication by means of sharing a replica of the downloaded content. For this, B must send a *replication offer* message to the particular content owner A.

$$B \rightarrow A : E_{K_A}(\langle REPLI\_OFF, B, A, ts_0 \rangle), \\ s_B(\langle REPLI\_OFF, B, A, ts_0 \rangle)$$

   If B belongs to A's table of subscribers, it means that B has a clearance with a specific label L_i issued by A. Thus, A will evaluate B's aptitude for replicating her contents of that security label. In case of acceptance, A re-generates and sends a modified authorization certificate $C_B^A : \langle B, A, (ts_1, ts_2), sign.AlgorDesc, L_i, L_j \rangle$ to B (following the same message exchange as in Join subprotocol.)

   On the other hand, if B is not a subscriber or is unknown, A first can initiate an instance of Join subprotocol, and evaluate the trustworthiness of B. I any case, A can reject the offer.

2. On owners demand. To simplify the discussion of the delegation procedure, we assume a node can be an owner's delegated node if that user is already a member of that owner's trusted group. Typically, a content owner demands this kind of distributed storage request whenever she needs availability support.

   A particular owner initiates a replication request choosing one or more

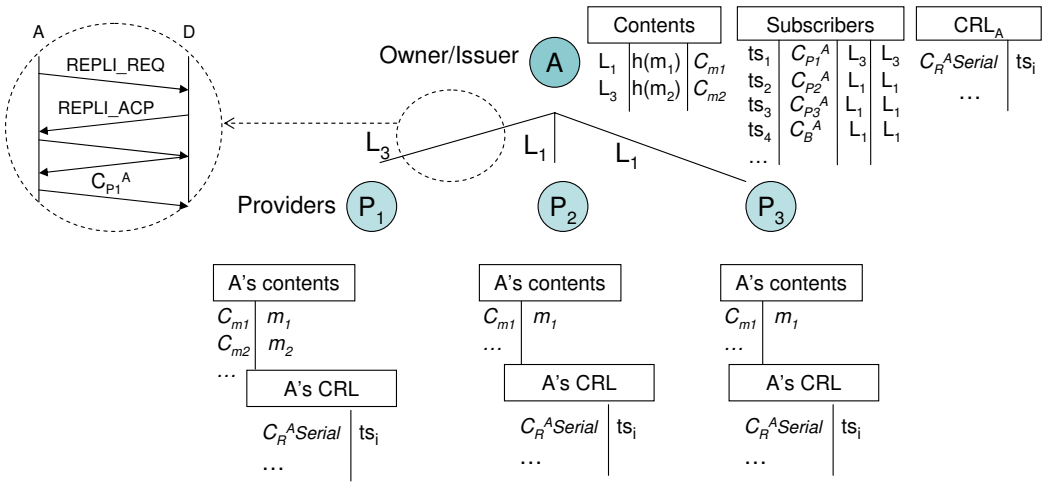


Figure 4.9: Owner's information transmissions at delegation.

providers, according to the required security label. First, A chooses k delegated nodes $\{P_1, \ldots, P_k\} \in T_A$, sends them and signs *replication request* messages:

$$A \rightarrow P_i \in T_A : E_{K_{P_i}}(\langle REPLI\_REQ, A, P_i, ts_0 \rangle), \\ s_A(\langle REPLI\_REQ, A, P_i, ts_0 \rangle)$$

If P_i accepts the request, she sends back to A:

$$P_i \rightarrow A : E_{K_A}(\langle REPLI\_ACP, A, P_i, ts_1 \rangle), \\ s_{P_i}(\langle REPLI\_ACP, A, P_i, ts_1 \rangle)$$

A will send a modified authorization certificate as in Join subprotocol.

Note that this delegation process does not allow the replication right to be further delegated. Thus, the maximum delegation depth is defined to impose only one level, i.e. providers may deliver neither clearances nor replication privileges; they cannot delegate this task to others. A replication tree is shown in Figure 4.9. Moreover, requesters still require the authorization certificate issued by the corresponding owner for accessing that owner's contents, either from the owner's repository or from the authorized providers'.

Table of subscribers $\mathbf{Ss_{n_i}}$

Date	Requester	Clearance	Replication
(ts_1, ts_2)	n_1	L_{n_1}	L_j
(ts_1, ts_2)	n_2	L_{n_2}	$NULL$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 4.1: Table of subscribers: New fields.

Content owners must keep a list with providers, the expiration date for replication, and the associated level, as shown in Table 4.1. This structure then includes a new field which represents the replication security label for each subscriber.

On the one hand, the delegating node A should send her subscribers along with their current clearance (2nd and 3rd columns in table of subscribers Ss_A) to her providers:

$$A \rightarrow P_i \subset Ss_A : E_{K_{P_i}}(\{Subscriber|Clearance\} \subset Ss_A), \sigma$$

where $\sigma = s_A(\{Subscriber|Clearance\} \subset Ss_A)$.

This database should be periodically updated from each owner to her providers due to the latter are not allow to update clearances. On the other hand, however, we can use an alternative data structure, as we will show later, due to the fact that it is possible to revoke a replication privilege if the owner wants, e.g. she suspects a provider has been compromised. Moreover, the validity period included into the authorization certificate should be refreshed by owners. This task involves the updating of user certificates in order to avoid undesirable accesses to fake replicas published by non-renewed/revoked providers. We further elaborate on revocation in next section.

Next, discovering and verifying delegation chains are two open issues: How requesters will know about owners' providers, and where this information is published. First, verification is easy by means of the authorization certificates. Nevertheless, it is unrealistic to provide a global storage of delegation credentials in such environments. Instead, we have to distribute them across the network. Our solution proposes the adaptation at the application layer, as follows. Before presenting the search result and therefore allowing downloading, the P2P client system should require, on the one hand, the existence of providers' authorization certificate, together with, on the other hand, the

appropriate security label in the replication field.

4.4.3 Revoking Clearances

Unexpected dishonest behaviors and some malicious actions, such as Sybil attack, force the requirement of either downgrading the privileges of a particular subscriber or revoking her granted clearance. Revocation implies some changes at the table of subscribers, and consequently implies the propagation of these modifications to the delegated nodes.

A traditional way such authorization certificates can be revoked is to use an online CA to sign negative certificates. However, not counting on a CA, a node would generally be unwilling to distribute its own negative certificate [65]. Thus, the proposed revocation process consists in the following tasks: first, the decision of revocation, and, finally, the propagation of the revocation information.

4.4.3.1 Revocation Decision

Although an authorization certificate is expected to be in use for its entire validity period, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. A content owner can compose *revocation requests* either periodically (depending on the authorization certificates' validity period she issued) or when she finds evidences of bad behavior into her group of subscribers.

An extended solution may include decisions requests based on the accusations of others. A particular node advises a certain owner (or perhaps a whole community) of an abnormal situation requiring certificate revocation. Such circumstances include to be aware of the facts like a change of name, or a compromise or suspected compromise of the corresponding (own) private key, among other. For the sake of security, this kind on model has to deal with trust management. We do not address this extension.

4.4.3.2 Generating and Updating a CRL

In general, the use of X.509 certificates within Internet applications require appropriate methods in order to assure the certificate validation before fully relying on the authentication or non-repudiation services associated with the

Certificate Revocation List $\mathbf{CRL_A}$

CRL:
`Issuer:` A
`thisUpdate:` ts_1
`nextUpdate:` ts_2 $(OPTIONAL)$
`revokedCertificates:` $SEQUENCE\ OF\{$
`CertificateSerialNumber` *Serial*
`revocationDate` $ts_i\}$
`Sign. Algorithm:` *AlgorithmDesc.*

`Signature:`
$s_A(CLR)$

Figure 4.10: Certificate Revocation List.

public key in a particular certificate. Under such circumstances, typically the CA is in charge of revoking the certificate by means of the CRL which is usually freely available in a public repository. A CRL is a time stamped list of certificates (more accurately: their serial numbers) which have been revoked, are no longer valid, and should not be relied on by any system user (RFC 3280 [63].)

Since there is not fixed infrastructure in our P2P revocation model, owners (particularly issuer which has a her disposal at least one delegated service) must locally carry out the generation and management of this structure. Thus, each issuer should maintain a local data structure identifying her revoked certificates, as shown in Figure 4.10. For example, each revoked certificate of owner A is identified by its certificate serial number in the CRL issued by A. Moreover, this structure is signed by the corresponding issuer, and is issued on a regular periodic basis (e.g., hourly, daily, or weekly.) We further elaborate on the propagation of this data in the following subsection.

4.4.3.3 CRL Propagation to authorized providers

Revocation requires two essential stages: CRL updating and providers updating. A *revocation request* is always sent from the interested issuer to her authorized providers, and, due to the one-level delegation model, revocation only depends on one-level of providers.

We show a typical scenario for revoking a certain subscriber B's clearance of L_i by the issuer A. First, B's clearance will be revoked by A. An entry is added to the CRL_A containing B's certificate serial number along to the

current date.

Now, due to the existence of delegation, there are several replicas of A's contents at different locations, i.e. on authorized providers' repositories. Thus, A must find all of her providers by means of the database containing the authorized delegated nodes and their clearance, and dump the modified CRL to them, as follows:

$$A \rightarrow P_i \subset Ss_A : E_{K_{P_i}}(CRL_A), s_A(CRL_A)$$

On the other hand, another possibility of revocation is downgrading, i.e. the issuer locally downgrades a certain subscriber's clearance, for example, to L_{i-1}. The resulted clearance depends on A's criteria. In such a scenario, the current clearance is revoked, and, therefore, a new one must be created through Join subprotocol.

In any case, the P2P file sharing system at P, after receiving an A-content search query from B, not only checks B's certificate fields and its validity but also acquires the most recently-issued CRL_A and checks that B's certificate serial number is not on that CRL.

Similarly, authorized providers can be also revoked by the issuer. In these cases, the system, at content search stage, will show available replicas located only at not revoked providers.

4.5 Efficiency Analysis

We have evaluated the efficiency of the entire proposal, including the delegation and revocation schemes, comparing them to the initial approach as well. We also consider both the computational (especially cryptographic) effort required by the nodes, as well as the communication cost of the entire scheme.

4.5.1 Computational Overhead

Table 4.2 presents the computational complexity of evaluating the cryptographic operations required, first, by the Join and Access subprotocols. The resulted complexity for the former is given by the unique instance of public key authentication protocol, $\mathcal{O}(k \log k)$. On the other hand, the content access relies especially on the verification of the content certificate, i.e. additional in-

Subprotocol	Stage	Crypto operations	Complexity
Join	1.1 B's Request	$1S$	$\mathcal{O}(k \log k)$
	1.2 A's checking	$1PI + 1V$	
	1.3 C_B^A generation	$3S + 3V + 1DE +$	
		$+(1P_G + 1P_S + 2S + 2V)$	
	Subtotal:	$4S + 4V + 1DE + 1PI$	
Update	2.1 B's Request	$1S$	$\mathcal{O}(\|Pz\|)$
	2.2 A's checking	$1V$	
	2.3 Challenge-Response	$1P_G + 1P_S + 2S + 2V$	
	2.4 C_B^A generation	$3S + 3V + 1DE +$	
	Subtotal:	$6S + 6V + 1DE + 1P_G + 1P_S$	
Content Authentication	3.1 C_m generation	$1H + 1S$	$\mathcal{O}(\|m\|k^3 \log k)$
	3.2 Signature process	$kS + kH + \frac{k(k+1)}{2}V + kPI$	
	3.3 Content publish.	0	
	Subtotal:	$(k+1)(H + S + \frac{k}{2}V)$ + kPI	
Access	4.0 Search response	$1V$	$\mathcal{O}(\|m\|k^2 \log k)$
	4.1 Access Request	$1V$	
	4.2 B's Checking C_m	$1H + kPI + kV$	
	Subtotal:	$1H + kPI + (k+2)V$	
Delegation	5.1 Replication Request	$1S + 1V$	$\mathcal{O}(1)$
	5.2 A's checking	$1S + 1V$	
	5.3 C_P^A generation	$3S + 3V + 1DE$	
	Subtotal:	$5S + 5V + 1DE$	
Revocation	6.0 Revocation decision	0	$\mathcal{O}(p)$
	6.1 CRL update	0	
	6.2 Providers update	$p(DE + V) + 1S$	
	Subtotal:	$p(DE + V) + 1S$	

Legend: H: Hash generation; DE: Asymmetric decryption (encryption is negligible); S: Signature generation; V: Signature verification; k: No. of signers; PI: Pathak and Iftode's protocol; P_G: Puzzle generation; P_S: Puzzle solving; p: No. of providers;

Table 4.2: Efficiency analysis (computational effort) for each stage of the entire proposal.

stances according to the number of signers, and on the content length as well. Note that this task is now executed in the Content Access subprotocol. Thus, this fact involves a quadratic complexity.

We use previous tables (also considering the speed benchmarks corresponding to the cryptographic primitives used, included in Section 3.6) with the aim of measuring the computational cost for content access subprotocol. For instance, Figure 4.11 shows the cost in the *worst* case (no signers are known and all the verifications must be performed), so these curves must be seen as an upper bound. Regarding the best case, when all signers in the content certificate are known for the requester, implies that signatures verifications are

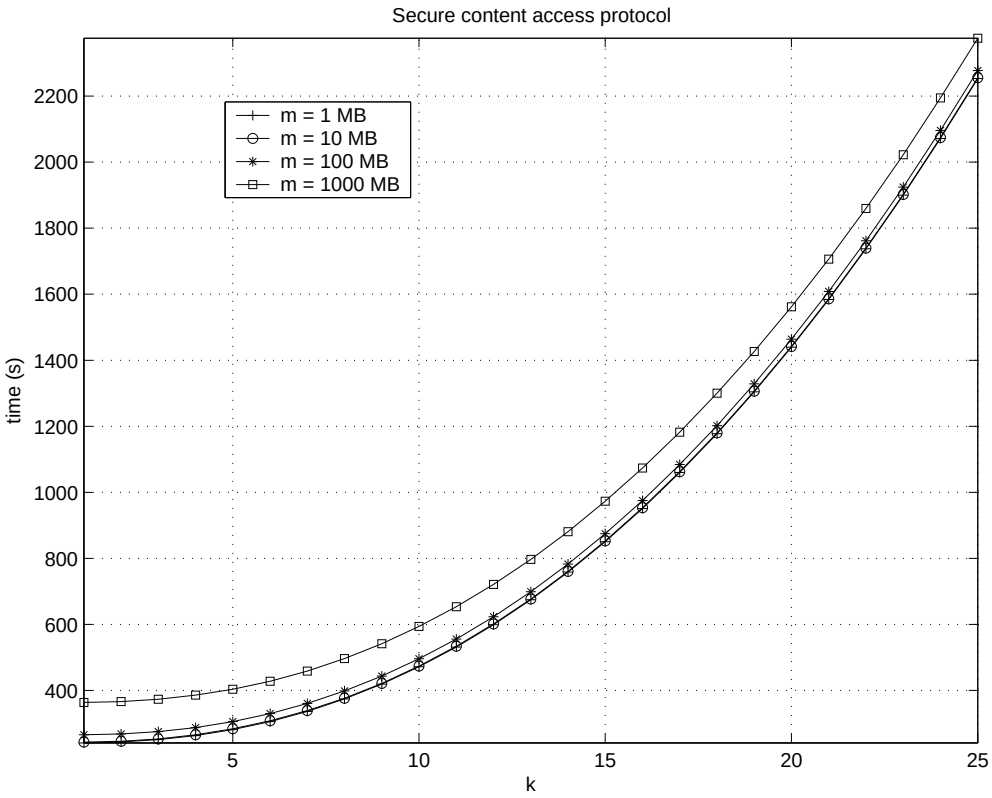


Figure 4.11: Computational cost (worst case) for content access subprotocol.

immediate, e.g. the verification of 100 signatures takes only 8.2660 ms.

On the other hand, delegation subprotocol is less complex, since it requires that both the owner and the (future) provider are mutually authenticated. In fact, we have assumed potential providers to have acquired the objective content. Nevertheless, revocation incurs additional cost which depends on the number of providers. Both processes keep constant.

4.5.2 Communication Overhead

We also analyze efficiency according to the communication overhead imposed by the new subprotocols. Table 4.3 presents the number of messages and the complexity in terms of communication transmission for each stage. As shown, Join algorithm is performed with complexity $\mathcal{O}(k \log k)$ due to the execution of a unique instance of the requester public key authentication by the issuer. On the other hand, the complexity of Access subprotocol varies according to the number k of signers who participated in signing the content certificate. Thus,

at worst, the requester must perform k instances of the Pathak and Iftode's protocol, and therefore this implies a complexity of $\mathcal{O}(k^2 \log k)$.

Figure 4.12 plots how the number of transmitted messages increases depending on the number of signers involved in each subprotocol. Obviously, the Content Authentication subprotocol requires a higher number of transmissions. In the case of Join, communication overhead actually depends on the authorization certificate length and the complexity of the challenges in Pathak and Iftode's messages, but we can consider them negligible.

Similarly to computational analysis, delegation is less complex than revocation, which has to update the CRL at every provider location; however, both communication complexities are still constant.

Subprotocol	**Stage**	**No. messages**	**Complexity**
Join	1.1 B's Request	1	$\mathcal{O}(k \log k)$
	1.2 A's checking	$PI + 1$	
	1.3 C_B^A generation	2	
	Subtotal:	$PI + 4$	
Update	2.1 B's Request	1	$\mathcal{O}(1)$
	2.2 A's checking	0	
	2.3 Challenge-Response	2	
	2.4 C_B^A generation	3	
	Subtotal:	6	
Content Authentication	3.1 C_m generation	1	$\mathcal{O}(k^3 \log k)$
	3.2 Signature process	$2k + \frac{k(k+1)}{2} PI$	
	3.3 Content publishing	0	
	Subtotal:	$2k + 1 + \frac{k(k+1)}{2} PI$	
Access	4.0 Search response	1	$\mathcal{O}(k^2 \log k)$
	4.1 Access Request	1	
	4.2 B's Checking C_m	kPI	
	Subtotal:	$kPI + 2$	
Delegation	5.1 Replication Request	1	$\mathcal{O}(1)$
	5.2 A's checking	1	
	5.3 C_P^A generation	3	
	Subtotal:	5	
Revocation	6.0 Revocation decision	0	$\mathcal{O}(p)$
	6.1 CRL update	0	
	6.2 Providers update	p	
	Subtotal:	p	

Table 4.3: Efficiency analysis (communication overhead) for each stage of the entire proposal.

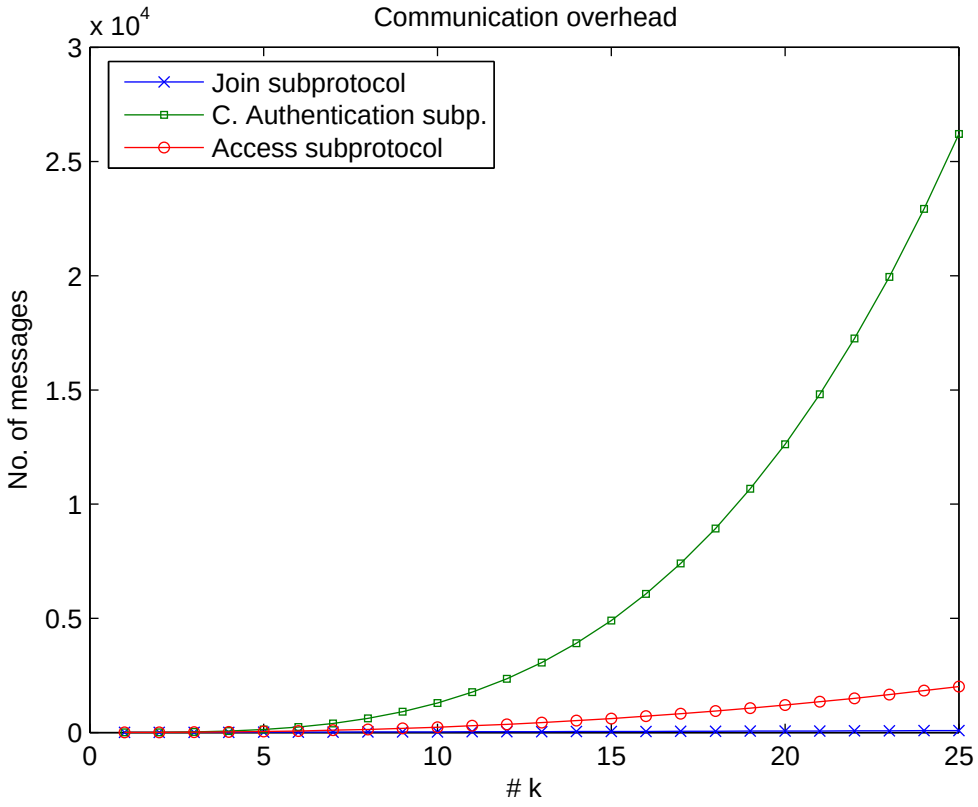


Figure 4.12: Communication cost (number of messages transmitted in the worst case) for each subprotocol.

4.6 Security Analysis

We also provide an informal analysis concerning the correctness of these extensions in terms of security. For this, we discuss several attack scenarios and forms of malicious behavior which can occur during each subprotocol execution, showing how the scheme is prevented against them.

4.6.1 Join and Update Subprotocols

Here, we discuss several attack scenarios that may be launched against the Join subprotocol.

1. *Eavesdropping.* An attacker can listen the four messages transmitted during the Join subprotocol. However, due to all the communication between A and B are asymmetrically encrypted by the sender using

the recipient's public key, an attacker cannot gain any useful information. On the other hand, suppose an eavesdropping peer can observe the messages tagged as $m_{1...4}$ at Fig. 4.5. In theory, these messages reveal nothing about both entities, since they are all correctly encrypted using the recipient's public key. Nevertheless, dishonest nodes may try to use others' computational resources for solving hers challenges and puzzles, like a type of free-riding situation. Basically, a malicious node C, decrypts and uses the challenge received from one of her issuers (say A) and sends it to challenge one of her requesters, B. Since B is honest, he solves and sends the puzzle solution to C who decrypts it and encrypts it again but using K_A. A will receive the correct answer sent by C but solved by B. This is related to the so-called *grandmaster postal-chess problem* [86]. For this reason, puzzles must been associated with both entities, with the content, and with the transaction as well. Consequently, it will not be able to have any "zombie" for delegating puzzle computation and any requesters' resource exhaustion either.

2. *Message modification attacks.* An attacker can try to modify some messages with the hope of modifying the privileges granted by A to B. As in the previous case, the use of encryption between both parties prevent this situation to occur.

3. *Message reply and insertion attacks.* In the former, an attacker first listens the messages exchanged between A and B and then tries to reproduce the session by using some of them. Similarly, an attacker can try to generate fake messages and insert them in the channel between A and B. However, she is unable to do that for the same reason previously described for the case of *message modification attack.*

4. *Impersonation attacks* of the messages exchanged between A and B. A major security concern of challenge-response-based schemes is adversaries who eavesdrop and subsequent attempt to impersonate the role of an honest peer. Impersonation may be trivial if an adversary is able to discover a node's long-term (secret or private) keying material, using for example a chosen-text attack. Briefly, suppose that authentication here consists of A challenging B with a random number r_A, RSA-encrypted under B's public key, and B is required to respond with the decrypted

r_A. If A challenges B with $r_A = h(x)$, B's response to this authentication request may (unwittingly) provide to A its RSA signature on the hash value of the (unknown to B) message x. This may be possible in protocols which are not zero-knowledge, because the claimant uses its private key to compute its response, and thus a response may reveal partial information. Other active attacks may involve the adversary itself initiating one or more new protocol runs, and creating, injecting, or otherwise altering new or previous messages [86]. Witness-indistinguishability and proof-of-knowledge techniques [18] deal with this issues. Our model should deal with them through embedding in each challenge response a self-chosen random number, combining use of random numbers with short response time-outs and using timestamps.

5. *Message dropping attacks* and *Attacks against the public key authentication process* are limited in the same way as analyzing the security of our first approach in Chapter 3.

6. *Dishonest providers.* A real problem occurs when the provider does not send the appropriate user certificate after receiving the correct puzzle response. Fair exchange and zero-knowledge protocols address this kind of misbehavior [11]. Message m_1 could be used as a proof of A's misbehavior. In particular, this message can be used as a non-repudiation evidence by the requester.

7. *Traitors.* Moreover, it is unrealistic to assume that every integrating node will exhibit a honest behavior forever, even if they have systematically behaved correctly in the past. A malicious party can behave properly for a period of time and then, after reaching a good clearance, begin misbehaving. In such scenario, system may provide misbehavior feedback among the peer community aimed at downgrading dishonest nodes. However, we try to avoid using a trust metric due to whitewashers and collusion attacks in which adversaries may impersonate others and use the spoofed identities to launch false accusations.

8. *No cooperation.* Cooperation is critical in both subprotocols, otherwise user clients would show no contents.

Attack	Vulnerable	Comments
Eavesdropping	No	
Modification	No	
Reply	No	This attack is easily detected
Insertion	No	
Dropping	No	A type of DoS can be mounted
Impersonation	No	This attack is detected
Against PKI	No	Requirement: Honest majority
Manipulation on C_B^A	No	
Manipulation on CRLs	No	
No cooperation	Yes	Critical in Join/Update
		Depends on node availability
Collusion	No	

Table 4.4: Summary of the informal analysis about the security of the proposed scheme.

4.6.2 Content Authentication Subprotocol

The discussion about the security of this subprotocol is detailed in previous chapter, Section 3.7.

4.6.3 Content Access Subprotocol

The search and downloading processes is external to our proposal and provided by a P2P file sharing application. Nevertheless, since we assume a certificate-using client, those processes should be adapted to such a middleware. Once the content has been successfully downloaded, there are several scenarios for an attacker:

1. *Manipulation of authorization certificates.* An user B with an insufficient clearance cannot upgrade it exclusively by herself. The upgrading process can only be done by A through the Update subprotocol. Furthermore, as the authorization certificate is signed by A, any modification performed on it will require to generate the new signature, a task that again only can be done by A by means of her private key.

2. *Manipulation of CRLs and/or content certificates.* Just like credentials manipulation, data structures such as CRL and content certificate may

be vulnerable by the aforementioned attacks unless they are not signed by the issuers.

3. *No cooperation.* Cooperation is less critical in Access since it is the client system who is in charge of manage the credential presentation for each situation.

4. *Software Manipulation.* The dependence of such an automatic client system enforces to protect carefully the software from any kind of attack or hole, as well as including correctly our protocol specification. For example, client system should securely provide key pair management, public keys, CRLs, authorization and content certificate repositories, protocol messages formats, etc.

4.6.4 Delegation and Revocation Services

We provide an informal analysis about the security of the proposed authorization services, just like we do with the subprotocols above:

1. *Eavesdropping.* An attacker cannot listen the messages transmitted during the delegation request and the propagation of the CRL for the communication between A and her P_i, since these messages are protected by using public key encryption. Note that revocation and delegation procedures assume instances between previously mutually authenticated parties.

2. *Message Modification Attacks.* An attacker can try to alter some messages with the hope of modifying any message or field in the authorization certificate. As in the previous case, such a modification can be detected due to it leads to an incorrect message authentication in the verification stages of the proposal, but never to any gains for the attacker.

3. *Message Reply Attacks.* An attacker first listens the messages exchanged between A and any other participant and then tries to reproduce the session by using some of them. The protocols proposed for the authorization services are robust against such a situation due to the identities of both parties are included in the messages, and their signatures as well, not being useful for anyone different from them.

4. *Message Insertion Attacks.* An attacker can try to generate fake messages and insert them in the channel. However, she is unable to do that for the same reason previously described for the case of *message reply attacks.*

5. *Message Dropping Attacks.* Given enough control over the network infrastructure, an attacker can try to delete some of the messages exchanged. By doing so, the only result achieved is a failure in the correct execution of the protocol (which can be viewed as a DoS), but cannot enable the attacker to gain any useful information.

6. *Impersonation Attacks.* Assume that an attacker can success in hijacking a revocation instance between A and her corresponding providers and subscribers, aimed at, for example, obtaining a privilege for replication. Even in this case, messages exchanged among trusted peers are safe from spoofing, due to the fact taht they are signed by authenticated public keys and also encrypted using the recipients' public key. In other words, the attacker cannot generate the correct digital signatures. Note that this is a particular case of trying to exhibit malicious behavior against the other party.

7. *No Cooperation.* Lack of cooperation in delegation and revocation processes does not influence as much as in the case of content authentication. Nevertheless, when requesting replication by a particular owner it may be probable none of his trusted nodes were available to response.

For the sake of illustration, we include Table 4.4 which summarizes the vulnerability of our protocol to the attacks analyzed.

4.7 Conclusions

Although access control is a key feature in content distribution, this has not been extensively addressed in current P2P scenarios. Traditionally, business corporations have used service-oriented collaborative tools to protect critical resources in the local communication network. In particular, access control in current P2P file sharing systems is chaotic, or even nonexistent. In fact, access control to the shared resources is a noteworthy problem for users of such

systems. This fact motivates us to propose an authorization protocol which provides access control to contents anonymously.

In our proposal, each collaborative requester is provided with a user certificate and the associated clearance, whose advantages varies according to his privileges. This clearance, the security level related to a user, determines the user's trustworthiness not to disclose sensitive data to others not cleared to access it. Moreover, objects in the system are assigned a security label as well. There is, therefore, a strict relationship between the security levels associated with the two, object and subject. In particular, the clearance provides its holder with the needed credential for the system to correctly download and access the desired content if and only if the clearance is authentic, high enough and not revoked. The resulted file sharing system then ensures that the protected content is only obtained by those users with the appropriate access level, that is, only by authorized users.

The aforementioned tasks are specified and automatically guided by the Join protocol. Experiments show us signs of the reliability of this proposal according to users' computational resources and interests. We, as many authors, try to probe that peers tend to collaborate even if they must spend some resources, and playing down the importance of indirect trust. Therefore file sharing can be encouraged by imposing a cost on the downloads, where each transaction implies to be worthy of access each time. Thus, owners, by means of a challenge-response protocol, can challenge requesters using, for example, a cryptographic puzzle, aimed at asking them to perform a computational effort.

Furthermore, the proposed authorization services, i.e. delegation and revocation, provides the system with a secure replication scheme so that only authorized providers will be able to distribute others' replicas. Issuer can control this privilege by means of the same clearances. Similarly, a clearance may be revoked by the corresponding issuer before its expiration of the validity period, as well as it may be upgraded. The Content Authentication protocol completes the scheme resulting a secure P2P content replication model.

Finally, future work mainly focuses on applying rational models, such as the Prisoners' Dilemma, in order to empirically observe a networking social pattern. In principle, peers generally self-organize in social groups of similar peers, called "semantic-groups", depending on the resources they are sharing. Some works try to exploit this social phenomenon by maintaining social networks

and using these in content discovery, content recommendation, and downloading. These social peer communities lead to challenging security concerns that seem to be more accepted by the real users. It is therefore interesting to study emergent topics related to the dynamics of cooperation and fairness, and which strategy leads to the formation of interesting social profiles, and content integrity protocols based on cooperation. Another interesting research direction is to study additional ways to combine trust schemes and traditional cryptographic protocols for P2P networks, such as multi-signature schemes and threshold cryptography.

Chapter 5

A Software Prototype

This chapter covers the groundwork, the methodologies and structures to create a software product for our proposal. Experiences from the use of an iterative and incremental development in practice appears to be more promising in this case, since the overall software life cycle seems to be incrementally updated. Although an exhaustive protocol's description has been already exposed in chapters above, this chapter addresses it in terms of technical software requirements aimed at developing a software prototype.

5.1 Purpose

The final software prototype should be able to automatically provide authorization and authentication capabilities. Thus, this objective of the entire proposal is structured in two processes: the access control and the content authentication. First, the access control process (Section 5.3.3.5) must apply the requirements established in its conception: issuing authorization certificates, which identify the corresponding user's clearance; assigning security labels to classify contents; and implementing the challenge-response procedure for managing nodes' privileges and access rights. The suggested idea is that content providers challenge requesters before supplying the required content. A requester who do not pass the challenge would be denied access to requested material, like a "proof-of-work". In particular, access control is divided between the process of join the community, the clearance update, the content access and the proposed authorization services, i.e. delegation and revocation (Section 5.4.3.5.)

Similarly, content authentication is represented by a process based on the use of an honest majority of networking nodes to create a digital certificate for an electronic file that can be used to assure the file's integrity and verify the identity of the file's owner. This process is composed of three phases: (1) a content certificate generation; (2) a content certificate signature; and (3) a content certificate verification. In the first, a content owner generates a digital certificate for a certain content m, containing the main information of the content (e.g. the hash of the file) and about himself (see data structures in Section 5.4.2.) In the content certificate signing (Section 5.4.3.4), the owner selects a subset of nodes (also called *signers*) in the community and sends a signed message with that information to them. Selected nodes verify the accuracy of this information, and check some local registers as well. According to the results of these local verifications, signers add their digital signature to the received certificate, and keep track of the added signature.

A user desiring to verify the content's integrity retrieves the file with the content certificate and uses it to verify, first, that the certificate was created by a majority of honest nodes; and second, that the hash of the file in the content certificate is the same as the hash computed from the retrieved file (see Section 5.4.3.7 for further details.) If these two hashes match and the content is correct, then the user is assured that the file was originated from the owner, the content is uncorrupted, and signers may be stored as trusted, to some extent.

5.2 Software Architecture

Software architecture allows us to visualize, understand and reason at a high level of abstraction about the architecturally significant elements and identify areas of risk that require more detailed elaboration. Fig. 5.1 illustrates the main elements (data, actors, procedures) of the software prototype and the interaction among each other and between other system interfaces as well.

In this figure, we can identified the following modules: (1) the three main stages which compose the content authentication protocol; (2) the three main stages which compose the access control and authorization services; (3) roles or actors for each module; (4) data structures; and (5) the interaction with the public key authentication module and the underlying transaction management.

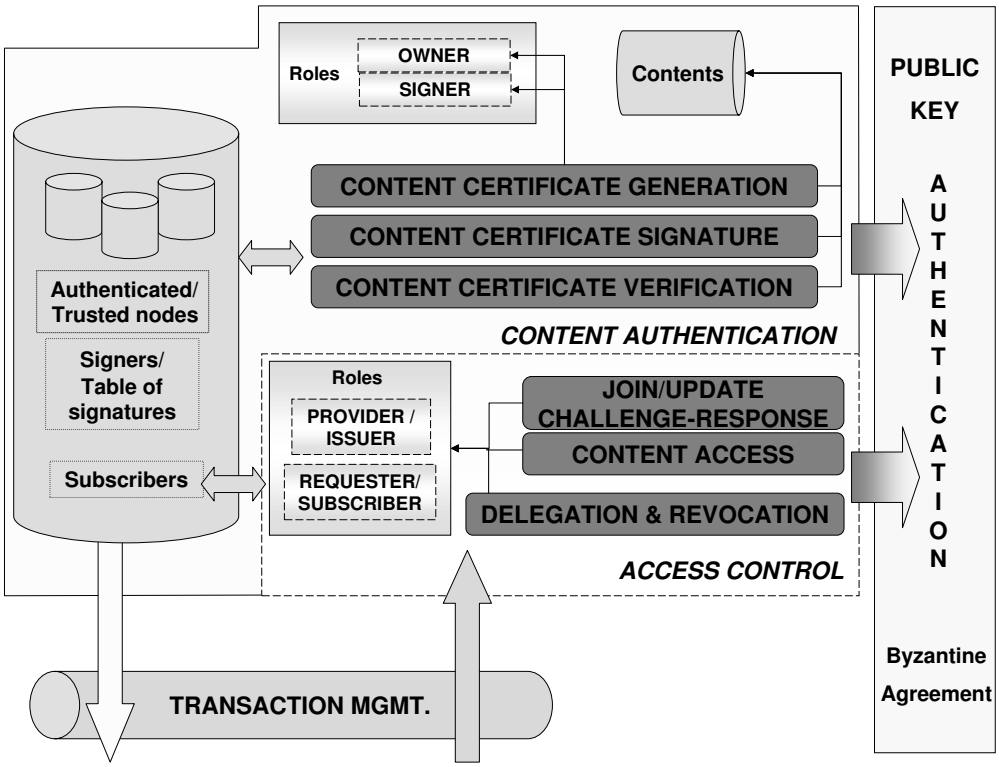


Figure 5.1: Software architecture.

In particular, the content authentication service requires a unique owner actor and a certain number of signer actors picked out from the owner's authenticated/trusted nodes database. During the content certificate signature process, there will be as many instances of Pathak and Iftode's protocol as needed. On the other hand, the transaction management layer creates and maintains transaction states for each protocol instance. Similarly, the access control stages interact with both public key authentication protocol and with the underlying transaction stack. We further elaborate on these modules in next subsections.

5.3 Software Elements Specification

Writing software requirements specifications lays out the foundation for the development of a new software product. This section describes the main structures and stages of the entire protocol.

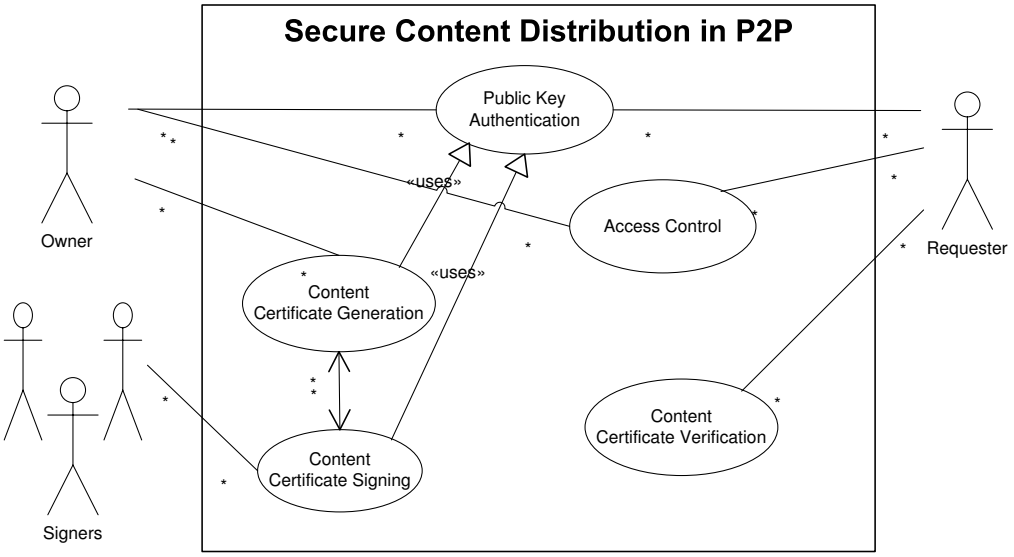


Figure 5.2: Our P2P secure content replication system: Use case.

5.3.1 Content Certificate

A content owner generates a certificate C_m for content m. Moreover, each node must maintain a local register with the certificates it has previously signed, S_{n_i}. The requirements for both structures are detailed in Figure 3.1 in Chapter 3.

5.3.2 Authorization Certificate

A content owner A can generate an authorization certificate for node B, C_B^A. This data structure must maintain both identities, the certificate serial number, the validity period for C_B^A, B's security clearance L_i, the replication privilege if B is an authorized provider, and all fields signed by both entities as explained in Chapter 4. Moreover, each issuer must keep track of subscribers and current clearances at a local structure Ss_{n_i}.

5.3.3 Protocol Stages

We apply use cases and activity diagrams in order to capture the functional requirements of the five main software modules: public key authentication, the three stages for content authentication, and the access control module (see Figure 5.2.) We further elaborate on each use case in subsections bellow.

5.3.3.1 Public Key Authentication

The public key authentication protocol consists of four main phases that have been extensively discussed in Chapter 2.

The deployment of this software module is required in order to exhaustively analyze our protocol. In this section, we present a initial description of the interaction between a primary actor, the initiator of the public key authentication, and the sequence of principal steps.

Figure 5.1-(a) depicts the use case diagram. Table 5.1 briefly documents the most interesting fields of each use case.

Figure 5.3 depicts the activity diagrams illustrating the process for creating a typical scenario and the stages of a public key authentication.

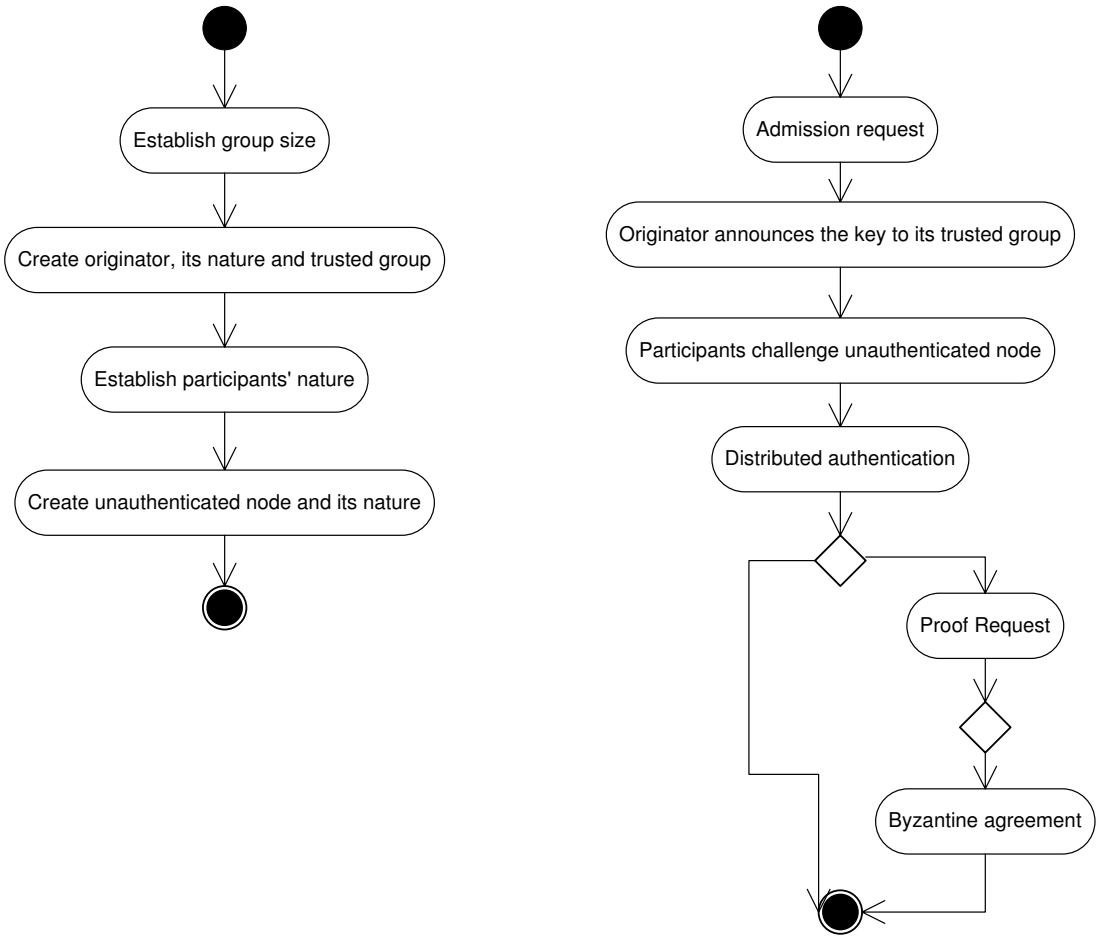
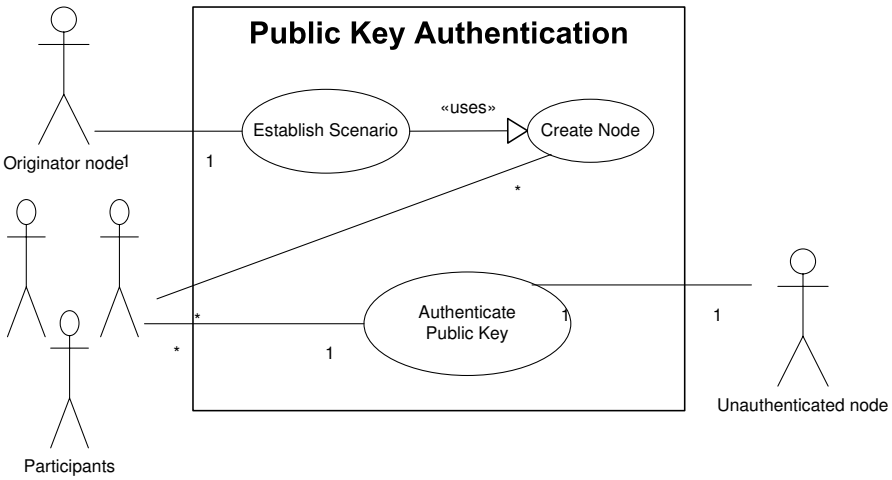


Figure 5.3: Public key authentication: Activity diagrams of Establish context, and Authenticate Public Key, respectively.



Use case name:	**Establish Scenario**
Actors	Originator node
Summary	Create an scenario with all networking nodes and their corresponding roles
Preconditions	−−
Postconditions	Generate nodes: Originator, Unauthenticated, and participant nodes
Notes	The user can define the size of the community, and the nature (honesty) of participants
References	*Create Node*

Use case name:	**Create Node**
Actors	−−
Summary	Generation of all networking nodes
Preconditions	Node's role
Postconditions	Node generation
Notes	The system automatically creates the local information related to each networking node according to the corresponding role: key pair, trusted nodes, etc.
References	*Establish Scenario*

Use case name:	**Authenticate Public Key**
Actors	Participants and unauthenticated node
Summary	Execution of the public key authentication protocol [97]
Preconditions	*Create Node*
Postconditions	Result of public key authentication
References	*Establish Scenario*

Table 5.1: Public key authentication use case: Specifications.

5.3.3.2 Access Control: Join/Update

The proposed access control subprotocol consists of two main phases that have been extensively discussed in Chapter 4: Join and Content Access.

The purpose of Join phase ranges from granting access rights to honest requesters (from the content owner point of view) to reach access privileges by the requesters in order to get the desired content. Requesters who want contents from a particular provider first have to engage in a four-step protocol in order to obtain a authorization certificate from the corresponding content owner. These tasks are represented in Figure 5.2 which depicts the use case diagram, together with the clearance Update procedure.

Messages for Join subprotocol are illustrated in Fig. 4.4. Moreover, by the sake of illustration, the activity diagram of this stage is shown in Fig. 5.4. Update stage just includes a prior request message.

5.3.3.3 Content Certificate Generation

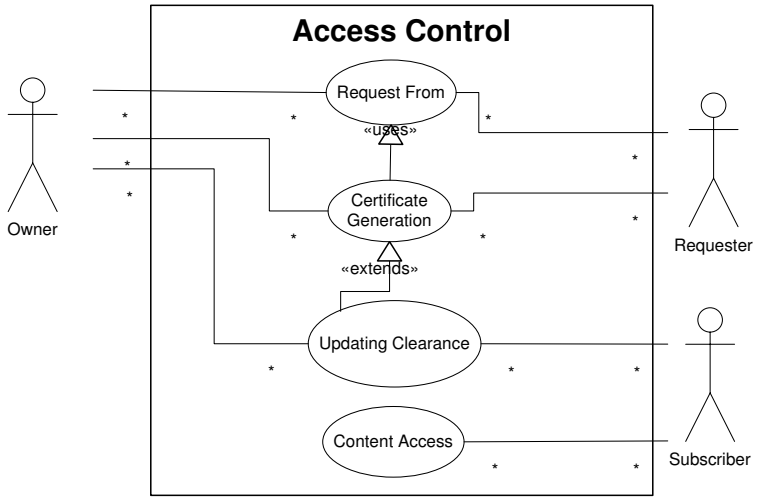
The certificate is initialized by the owner (the protocol originator), who selects an appropriate value for the number k of signing nodes and their identities (see subsection 5.3.1 above.) Next, the owner generates an initial content certificate, denoted as C_0, by providing the first signature (his own signature over the aforementioned content information) and passes it to the next node in the ordered list of signers. The activity diagram of this stage is shown in Fig. 5.5.

5.3.3.4 Content Certificate Signing

When describing the proposal, we identified two main alternative to sign the content certificate. Now, for software construction we use the centralized signing process which simplifies our simulation prototype (based on threads and TCP Sockets.) In this case, the originator is responsible of sending C_{i-1} to each signer and receiving C_i. Concretely, the owner can check whether the received certificate has been properly signed or not, thus having a higher level of control over the process.

Moreover, signers should verify the following:

1. The owner's public key, and authenticating previous signers as well.



Use case name:	**Request Form**
Actors	Requester
Summary	Request for a clearance generation
Preconditions	−−
Postconditions	Authenticated parties
Notes	The requester should accept the assigned clearance
References	*Certificate Generation*

Use case name:	**Certificate Generation**
Actors	Owner and Requester
Summary	Generation of a user authorization certificate after a successful resolution of a puzzle
Preconditions	*Request Form*
Postconditions	Update local databases
Notes	Parties agree with the certificate fields
References	

Use case name:	**Updating Clearance**
Actors	Owner and Subscriber
Summary	Execution of a challenge-response protocol aimed at increasing clearance
Preconditions	*Certificate Generation*
Postconditions	Update local databases
Notes	By means of a challenge, owners evaluate requesters' interest
References	

Use case name: **Content Access** (see Section 5.3.3.5)

Table 5.2: Access control: Use Case.

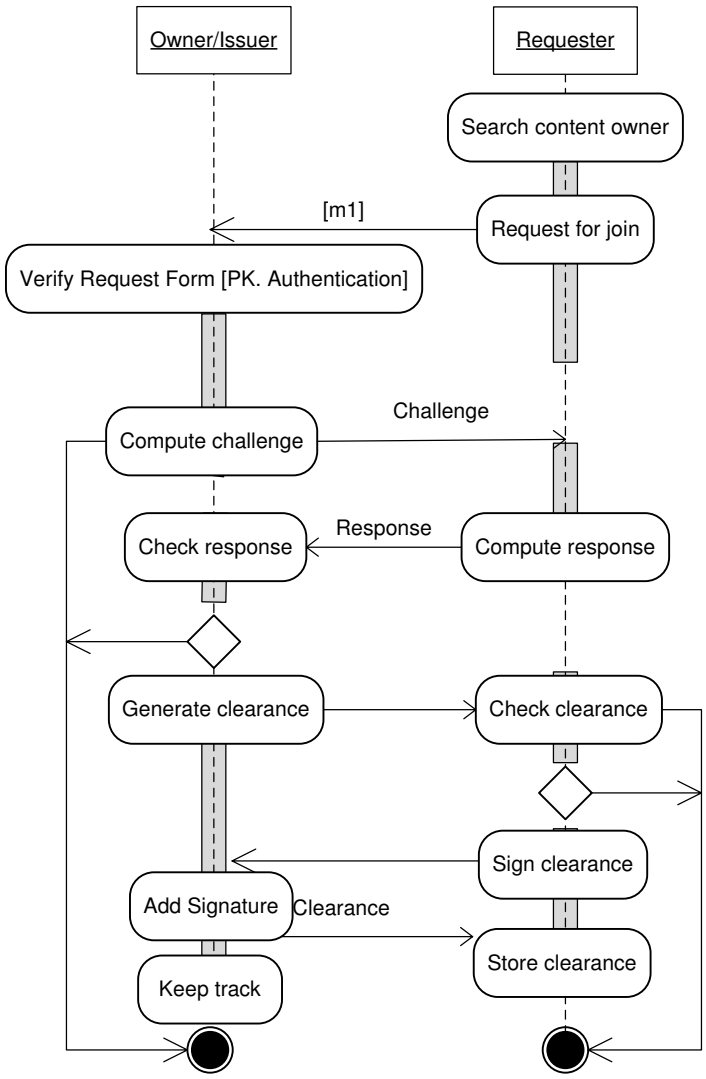


Figure 5.4: Join subprotocol: Activity diagram.

2. The hash of the content $h(m)$, comparing it with the value contained in the received certificate.

3. Its table of signed contents, checking that no entries exist corresponding to the same content.

4. The signatures contained in the received certificate according to the list order.

If previous verifications succeed, each signer adds its signature to C_{i-1}, thus creating C_i. Each peer then stores separately C_{i-1} and the generated

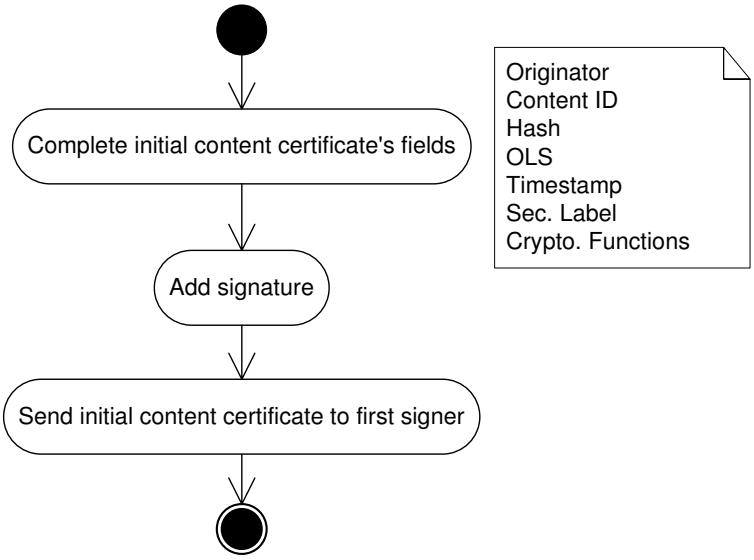


Figure 5.5: Content Certificate Generation: Activity diagram.

signature in her local database S_{n_i}. The resulted content certificate is sent to the originator (in centralized signing process.) Figure 5.6 depicts the activity diagram that signers must tackle to perform this phase.

5.3.3.5 Content Access

In order to set the basis of this stage, we briefly overview the interactions between actors in the sequential order that those interactions occur through the sequence diagram in Fig. 5.7.

5.3.3.6 Content Certificate Verification

Once obtained the content, the associated content certificate should be checked to ensure its authenticity and integrity. For this, a requester performs the following steps:

1. Requester computes $h(m)$ from retrieved content and compares the result with that included in the certificate. If both values differ, then either the content has been altered or C_m is not an authentic certificate for this content.

2. Requester obtains the public keys associated to each of the peers listed in

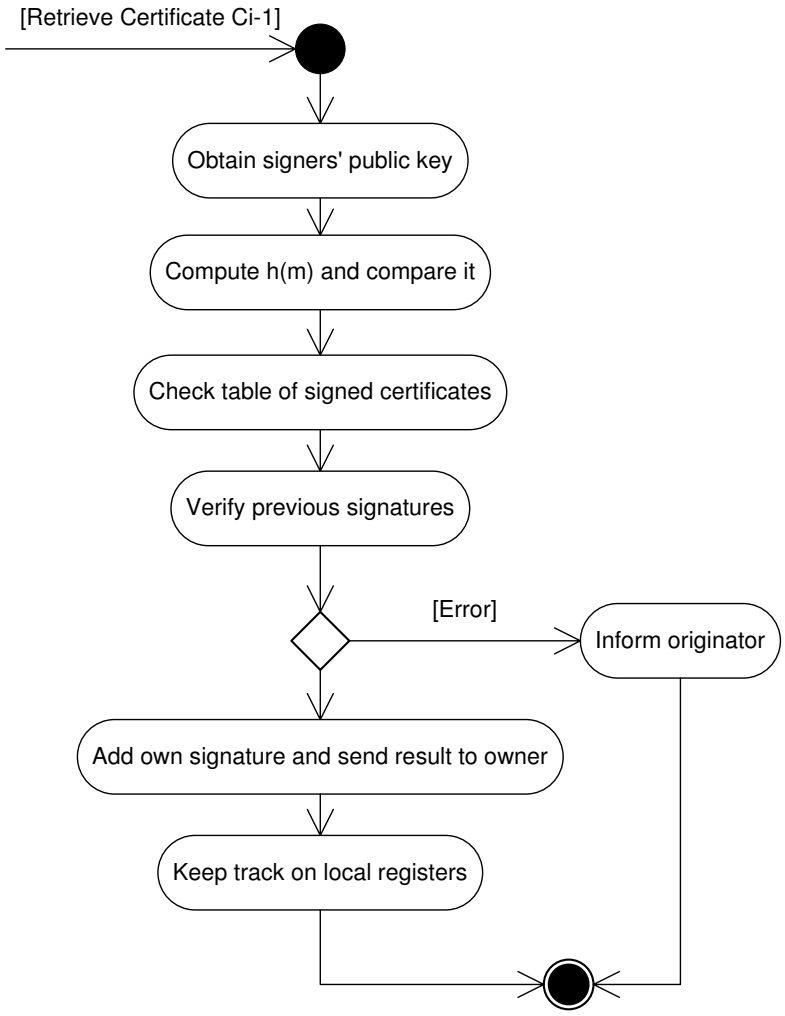


Figure 5.6: Content Certificate Signing: Activity diagram.

the OLS. If any public key is unknown, it can be acquired by executing the public key authentication protocol.

3. Requester verifies the chain of signatures by recursively encrypting C with the ordered list of public keys.

Figure 5.8 depicts the activity diagram that requesters must tackle to perform this phase.

5.3.3.7 Delegation and Revocation Services

The provision of the delegation service is strongly based on Join process. User authorization certificates will register the corresponding holder's replication

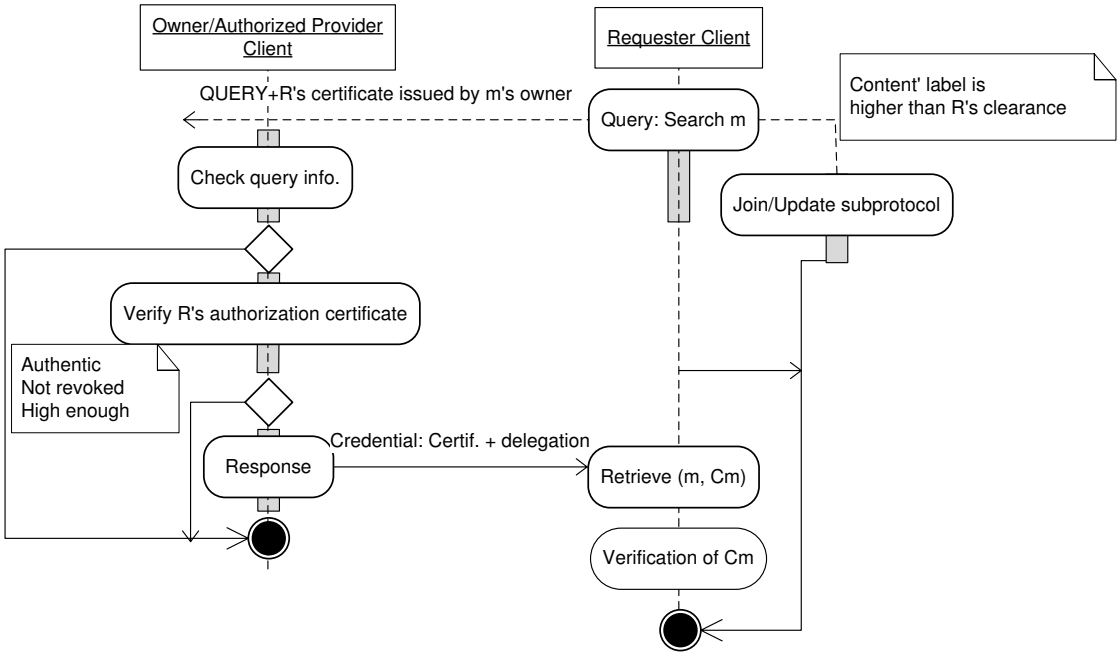


Figure 5.7: Content access stage: Activity diagram.

privilege as a new field. Obviously, as in Join, the issuer first accepts and therefore certifies this permission. On the other hand, both processes require an additional data structure, the CRL. Issuers will periodically refresh their subscribers/providers database, as well as dump it to the providers' repository. Figure 5.9 depicts the activity diagram that both an issuer and a potential provider(s) must tackle to perform theses phases. For implementation purposes, we assume the alternative for delegation based on owner demands.

5.4 Design

Among the most important aspects to consider in the design of the software prototype we should make an effort to modularity. Our resulting software is comprised of well defined, independent components which will be separately tested and then integrated in a final round. In this section, we describe the development environment and main data structures, and the design aspects of the major stages of our content authentication protocol as well.

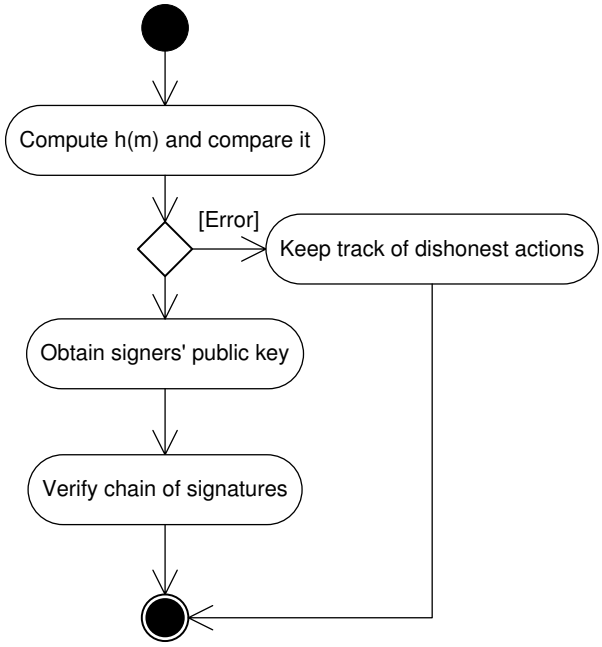


Figure 5.8: Content Certificate Verification: Activity diagram.

5.4.1 Deployment Environment

The environmental requirements are the following:

- The current prototype has been deployed on `Linux` Operating System. However, future developments will be tested on `Windows` as well.
- `C` and `Java` programming languages (mainly using `NetBeans IDE 5.0`) for implementation.
- `GnuPG` and `Java` security packets, `java.security.` and `javax.crypto.`, for cryptographic primitives.
- RPCs (*Remote Procedure Calls*) and/or TCP/IP sockets (*multi-threads* environment, e.g. using packet `java.net.`) for simulating different nodes in the prototypes.

5.4.2 Data Design

Basic data structures are the content and node certificates, and local databases containing signed certificates and trapdoors. We also include the register for

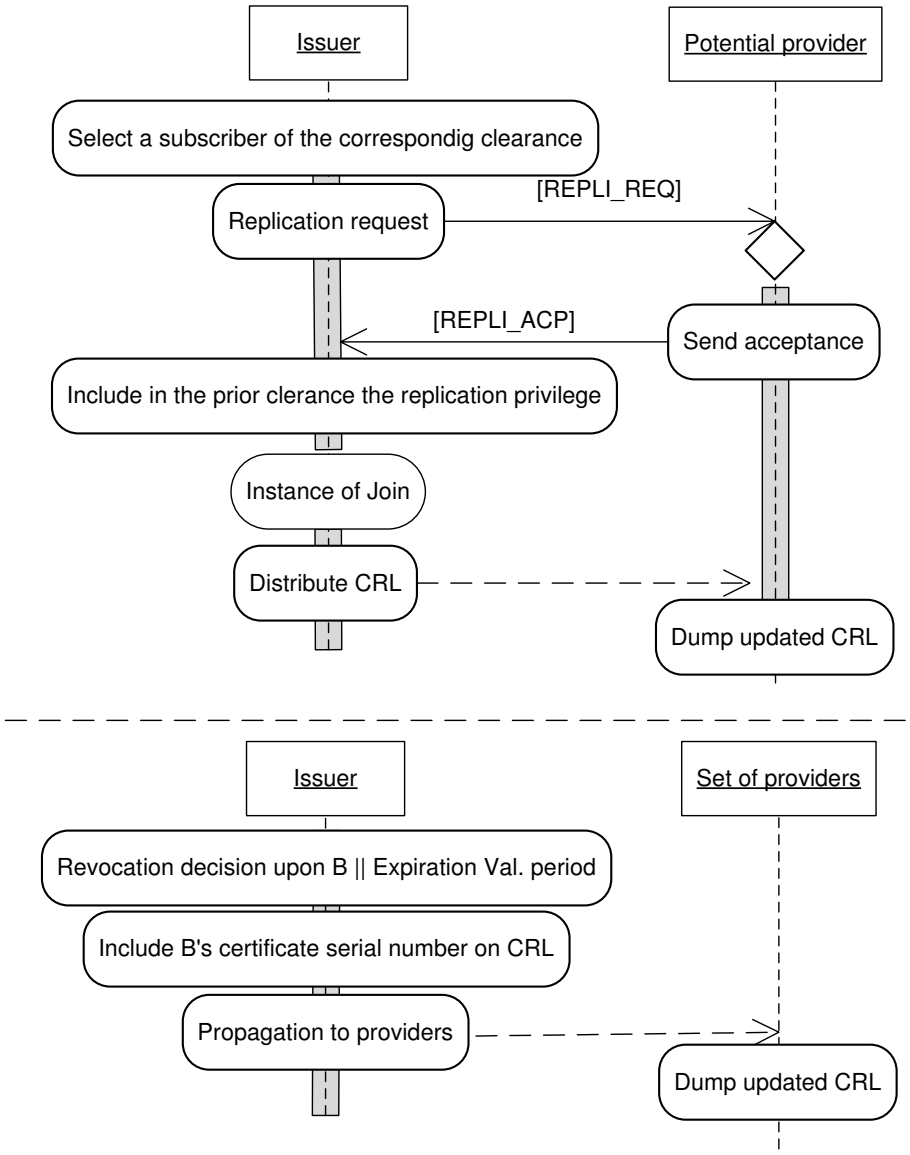


Figure 5.9: Delegation and revocation procedures: Activity diagram.

locally keeping the shared resources, and a structure to locally store the public information of other nodes in the community.

5.4.2.1 Authorization Certificate Structure

A user authorization certificate is structured in the following fields with the corresponding data type:

Data	Type
Holder	`string`
Issuer	`string`
Validity	`date-pair`
Clearance	`string`
Replication	`string`
Signature	`byte array`
Crossed Reference	`integer`

5.4.2.2 Content Certificate Structure

A content certificate is structured in the following fields with the corresponding data type:

Data	Type
Originator	`string`
ID	`string`
Hash	`byte array`
OLS	`string array`
Validity	`date-pair`
Security label	`string`
Sign. Algorithm	`string`
Signatures	`byte array`

5.4.2.3 Signed Certificates Structure

Signers keep track of signer certificates in the following local register:

Data	Type
Register ID	`enumerate`
Date	`System-date`
Owner ID	`string`
Content Hash	`byte array`
Certificate received	`Certificate`
Signature	`byte array`
Crossed Reference	`integer`

5.4.2.4 Public Information Structure

Nodes store public information of last contacted peers, as follows:

Data	Type
Node ID	`string`
Public Key	`byte array`
Security Label	`string`

Moreover, we can keep track of node behaviors according to the results of both content certificate signing process, and Byzantine agreement for public key authentication, if exists. The register may store the following items:

Data	Type
Timestamp	`date`
Node ID	`string`
OLS state	`mark`
Byzantine state	`mark`

5.4.2.5 Shared Resource Structure

Owners and providers (who offer replicas) publish the following data:

Data	Type
Content Hash	`byte array`
Content Certificate	`certificate`
Content Reference	`byte array`
Crossed Reference	`integer`

5.4.2.6 Table of subscribers/providers

Each issuer must store members or "subscribers" and their updated clearances, as follows:

Data	Type
Date	`date`
Requester	`string`
Current clearance	`string`
Replication	`string`
Crossed Reference	`integer`

5.4.2.7 CRLs Tracking

Content owners who have subscribers, i.e. issuers, must keep a local database containing the serial number of the revoked authorization certificates, as follows:

Data	Type
Serial number	`string`
Date	`date`
Crossed Reference	`integer`

This structure must be distributed to authorized providers.

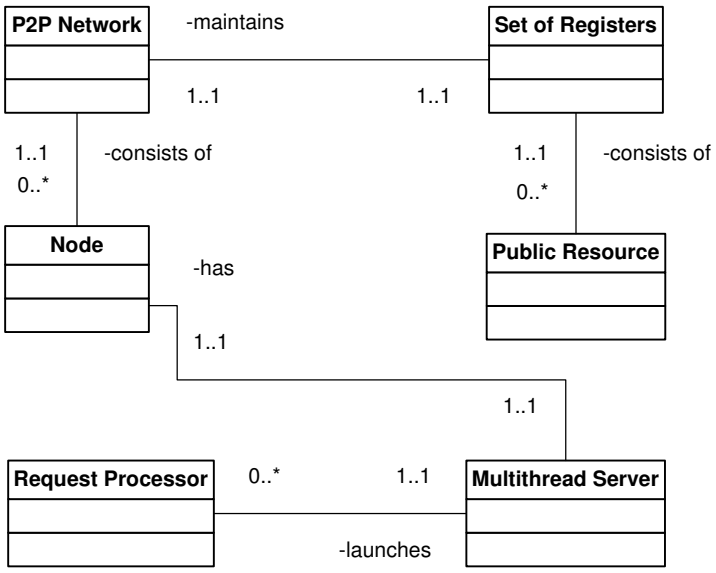


Figure 5.10: Establish scenario: Class diagram.

5.4.3 Process Design

In this section, we overview the design of the protocol stages separately.

5.4.3.1 Establish Scenario and Create Node

At a initial stage, the number of nodes in the community and their corresponding role must be established in order to perform the different protocols in the system. Basically, each node will create several threads during its execution aimed at performing the corresponding protocol task. For example, a node, who desires to publish a content assuring its integrity, has to be identified as the originator in a content authentication protocol instance. In Figure 5.10 we show the class diagram with which we can simulate a P2P network scenario.

On the other hand, the network scenario is established by means of several data registers and functionalities. We have to store on each node every needed information of the particular role, e.g. the name, TCP port, type, set of trusted nodes, data generated during a certain stage, own resources, among others. In particular, the scenario for our content authentication protocol may be specified in these two main procedures:

- `Originator`. The content owner is created as a process and is defined by the following fields in the `Node` structure/class:

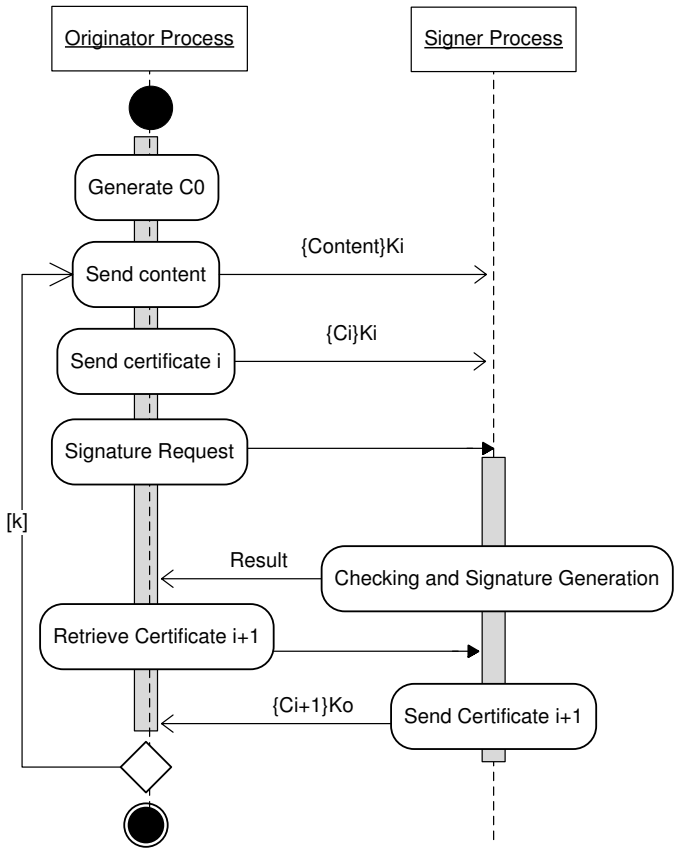


Figure 5.11: Invoked methods for the two roles in content authentication protocol: Sequence diagram.

Data	**Type**
Alias	`string`
Public Key	`byte array`
Private Key	`byte array`
IP address	`byte array`
Port	`integer`
Nature	`string`

Figure 5.11 shows the sequence of the invoked methods for the two functionalities.

- `Signer`. Signers are also created as processes and are defined by the fields in the aforementioned `Node` structure.

The table of its previously contacted peer may be also created in this phase. Moreover, both types of nodes may be easily identified by $< alias >$ @ $< ip\_address >$.

5.4.3.2 Authenticate Public Key

The implementation of the public key authentication protocol is briefly described and analyzed in this section.

First, we adopt a particular format for the messages that protocol participants will exchange, as follows:

Message items	,	$\cdots$	$s_{n_i}\{\cdot\}$ = Sender's signature on message items

$$\text{Admission: } \begin{cases} A \rightarrow B : \{A, B, admission\_request, K_A\}, s_A\{\cdot\} \\ B \rightarrow P_i : \{B, P_i, authentication\_request, s_A\{A, K_A\}\}, s_B\{\cdot\} \end{cases}$$

$$\text{Challenge-Resp.: } \begin{cases} P_i \rightarrow A : C_{iA} = \{P_i, A, challenge, E_{K_A}(r_i)\}, s_{P_i}\{\cdot\} \\ A \rightarrow P_i : R_{iA} = \{A, P_i, response, r_i\}, s_A\{\cdot\} \end{cases}$$

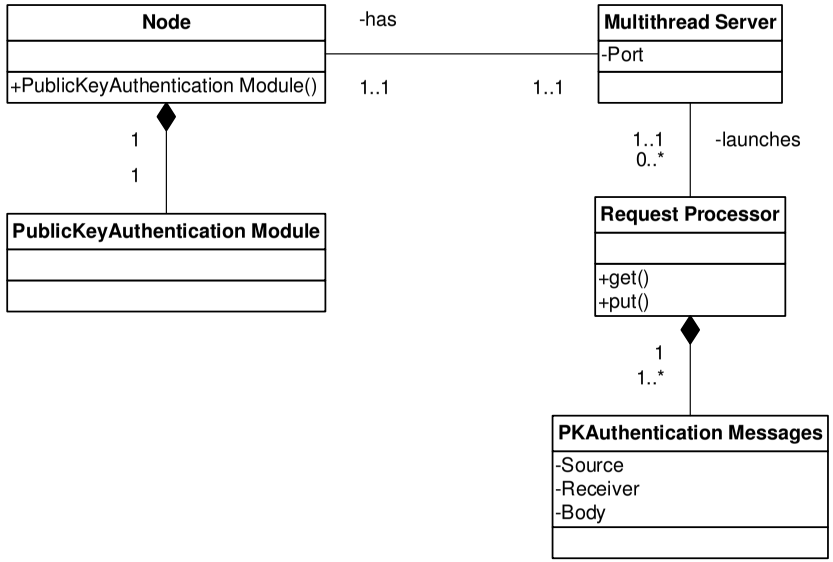
$$\text{Distr. Authentic.: } \{P_i \rightarrow B : \{P_i, B, dis\_Authent, s_A\{C_{iA}, R_{iA}\}\}, s_{P_i}\{\cdot\}$$

$$\text{Byz. Agreement: } \begin{cases} B \rightarrow A : \{B, A, proof\_rqt\}, s_B\{\cdot\} \\ A \rightarrow B : \{A, B, proof\_sent, \nu_A[]\}, s_A\{\cdot\} \end{cases}$$

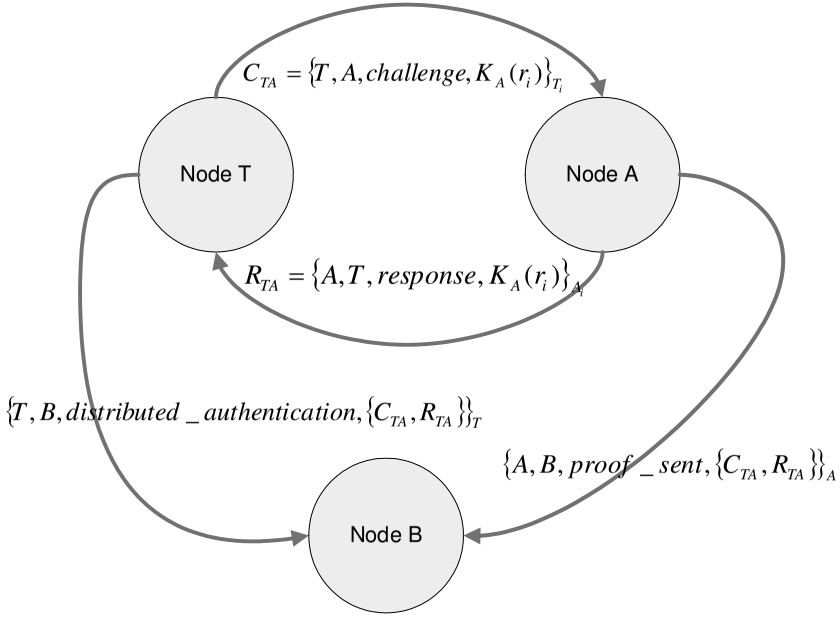
$$\begin{cases} B \rightarrow P_i : \{B, P_i, byz\_fault, \nu_B[]\}, s_B\{\cdot\} \\ P_i \rightarrow P_j : \{P_i, P_j, byz\_agreement, \nu_i[]\}, s_{P_i}\{\cdot\} \end{cases}$$

As an example, Figure 5.3-(b) depicts the formatted messages exchange between the three types of participants in the public key authentication protocol. As it is pointed out in the figure, it is an essential requirement that messages are signed (using `Java` this requirement is easily solved by means of the application of *SignedObjects*, predefined in `java.Security.`) Moreover, Figure 5.3-(a) shows the extensions of the class diagram for the public key authentication protocol.

Furthermore, the communication model to execute the public key authentication protocol is based on TCP socket connections. Similarly to a client/server model, the server establishes a port and listens it for clients' requests. During the connection, data exchange is performed through handlers, called *streams*. We need the following classes to manage the connections: `SocketServer`, `SocketHandler` y `SocketClient`. Basically, a new instanced `Node` creates a permanently active thread which type is `SocketServer`. The `SocketServer` object keeps on waiting for a given request, and detecting a connection on a given port. Once a connection is accepted, an object



(a) Public key authentication messages: Class diagram



(b) Exchange of evidences during Byzantine agreement

Table 5.3: Public key authentication: Class and state diagram.

`SocketHandler` is created; it is responsible of receiving messages and launching the corresponding actions. When a node wants to send a message, he needs a `SocketClient` object. Both roles' operation are implemented as follows:

```
public Message getMessage(SignedObject s_obj){
        Message mess=null;
        try {
            //Obtener el mensaje
            mess=(Message)s_obj.getObject();
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
        return mess;
}

public void protocolar(SignedObject s_obj){
        register.addMessage(s_obj);
        Message message=this.getMessage(s_obj);
        //First protocol stage
        if("admission request".equals(message.getTipo())){
            authenticationRequest(message);
        }else{
            //Second
            if("authentication request".equals(message.getTipo())){
                challenge(message);
            }else{
                //challenge response
                if("challenge".equals(message.getTipo())){
                    try {
                        response(s_obj);
                    } catch (IOException ex) {
                        ex.printStackTrace();
                    }
                }if("response".equals(message.getTipo())){
                    proofToMaster(s_obj);
                }else{
                    if("proofs sent".equals(message.getTipo())){
                        loadProofs(s_obj);
                        //all proofs received?
                        if( (register.getProofs().size())==(register.getTrusted().size())){
                            distributedAuthentication();
                        }
                    }else{
                        if("proof request".equals(message.getTipo())){
                            sendProofs();
                        }else{
                            if("proof".equals(message.getTipo())){
                                agreement(s_obj);
                            }else{
                                if("byzantine fault".equals(message.getTipo())){
                                    byzantineAgreement(s_obj);
```

```
                            }else{
                                if("byzantine agreement".equals(message.getTipo())){
                                    retrieveProofs(s_obj);
} } } } } } } } }
```

On the other hand, nodes need a cryptographic infrastructure to complete the following tasks:

- Challenge encryption. Honest nodes should encrypt the challenge (data type: `byte array`) using the public key to authenticate. We show some samples of the `Java` source code of some procedures:

```
public void authenticationRequest(Message mess){
    for(int i=0;i<register.getTrusted().size();i++){
         Peer p =(Peer)register.getTrusted().get(i);
         PublicKey key=mess.getSignature();
         Message m=new Message(register.getID(),p.getID(),key,"authentication request",
            register.getModule().getPublicKey(),register.getPort());
         SocketClient s=new SocketClient(p.getPort(),
            register.getModule().signMessage(m,register.getModule().getPrivateKey()),
                                              register.log);
        }
    }

public synchronized void challenge(Message mess){
        if(!register.honest){
            register.changeKeys();
        }
        Random r=new Random();
        Integer num=r.nextInt(1000000);
        String nonce=num.toString();
        register.setNonce(nonce);
        String challenge=(register.getModule().encrypt(nonce,(PublicKey)mess.getBody()));
        PublicKey key=register.getModule().getPublicKey();
        String source=register.getID();
        String recipient=register.getProbationary().getID();
        Message m2=new Message(source,recipient,challenge,"challenge",
                               key,register.getPort());
        SignedObject challengeSent=register.getModule().
                          signMessage(m2,register.getModule().getPrivateKey());
        SocketClient s2=new SocketClient(register.getProbationary().getPort(),
                                         retoSent,register.log);
        register.setChallenge(challengeSent);
        if(!register.honest){
            register.changeKeyss();
        }
 }
public void proofToMaster(SignedObject s_obj){
        Message mess=this.getMessage(s_obj);
        String response=(String)mess.getBody();
        boolean b=false;
```

```
        if (register.getNonce().equals(response)){
            b=true;
        }

        if(!register.honest){
            b=false;
        }
        register.setResponse(s_obj);
        register.setValidity(b);

        String source=register.getID();
        String recipient=register.getMaster().getID();
        int recipPort=register.getMaster().getPort();
        PublicKey key=register.getModule().getPublicKey();
        Message m=new Message(source,recipient,register.getProof(),
                              "proofs sent",key,register.getPort());
        SocketClient s=new SocketClient(recipPort,register.getModule().
             singMessage(m,register.getModule().getPrivateKey()),register.log);
    }
```

- Challenge decryption. For the inverse operation.

```
public void response(SignedObject s_obj) throws IOException{
        Message mess=this.getMessage(s_obj);
        //randomly challenge generation
        int recipPort=mess.getPort();
        String challenge=(String) mess.getBody();
        String resp=register.getModule().decrypt(challenge,register.
                                            getModule().getPrivateKey());
        String source=register.getID();
        String recipient=mess.getSource();
        PublicKey key=register.getModule().getPublicKey();
        Message m2=new Message(source,recipient,resp,"response",
                                  key,register.getPort());
        SignedObject respSent=register.getModule().signMessage(m2,register.
                                       getModule().getPrivateKey());
        SocketClient s2=new SocketClient(recipPort,respSent,register.log);
        Proof p=new Proof();
        p.setChallenge(s_obj);
        p.setResponse(respSent);
        p.setID(recipient);
        registwe.addProof(p);
    }
```

- Message signature and its verification. Note that all the protocol messages should be correctly signed by the sources, while honest nodes should verify signed messages.

Moreover, at the initialization of this protocol, we can set first the size of the trusted group, and then create the nodes. On the one hand, node A

(newly joined) generates her public key, B will try to authenticate that public key, and creates his subset of trusted nodes. Secondly, we can determine each participant nature in order to test all protocol steps, as follows:

```
private static void setTrusted() {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        try {
            System.out.print("Enter the number of nodes in trusted group: ");
            numTrusted = Integer.parseInt(br.readLine());
        } catch (Exception e) {
            e.printStackTrace();
        }
}

public static void setScenario(){
        master=new Peer(1521,"B","master",miLog,stop);
        proba=new Peer(1523,"A","proba",miLog,stop);
        trusted=new Peer[numTrusted];

        for(int i=0;i<numTrusted;i++){
            String id="P"+i;
            int port=2000+i;
            trusted[i]=new Peer(port,id,"trusted",myLog,end);
        }
        proba.setMaster(master);
        master.setProbationary(proba);
        for(int i=0;i<numTrusted;i++){
            master.addTrusted(trusted[i]);
        }

        for(int i=0;i<numTrusted;i++){
            trusted[i].setMaster(master);
            trusted[i].setProbationary(proba);
            for (int j=0;j<numTrusted;j++){
                String s1=trusted[i].getID();
                String s2=trusted[j].getID();
                if(!s1.equals(s2)){
                    if(!trusted[i].isTrusted(trusted[j])){
                        trusted[i].addTrusted(trusted[j]);
} } } } }

public static void setHonestNature(){
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String s=null;
        try {
            System.out.print("\n"+"Honest behavior of A (y/n) ");
            s = br.readLine();
        } catch (Exception e) {
            e.printStackTrace();
        }
        if(s.equalsIgnoreCase("no")){
```

```
            proba.setHonestidad();
        }
        InputStreamReader isr2 = new InputStreamReader(System.in);
        BufferedReader br2 = new BufferedReader(isr);
        String s2=null;
        try {
         System.out.print("\n"+"Enter the name of malicious nodes in the trusted group:");
         s2 = br.readLine();
        } catch (Exception e) {
            e.printStackTrace();
        }
        if(!s2.equalsIgnoreCase("")){
            String [] nodes=s2.split(" ");
            String s3;
            for(int j=0;j<nodes.length;j++){
                s3=nodes[j];
                for(int i=0;i<numTrusted;i++){
                    if(trusted[i].getID().equalsIgnoreCase(s3)){
                        trusted[i].setHonestNature();
}} }} }

public static void run(){
        System.out.println("\n"+"-Begining of the protocol-");
        System.out.println(".............................");
        master.start();
        proba.start();
        for(int i=0;i<numTrusted;i++){
            trusted[i].start();
        }
}
```

Finally, we need an especial structure to store all the proofs-of-possession (challenge responses) that B will require at Byzantine agreement stage, as follows:

```
public void byzantineAgreement(){
        System.out.println("\n"+"-No consensus: Byzantine Agreement-");
        System.out.println(".............................");
        String source=register.getID();
        String recipient=register.getProbationary().getID();
        String body="XXXX";
        String type="proof request";
        int port=register.getPort();
        Message m=new Message(source,recipient,body,type,
                        register.getModule().getPublicKey(),port);
        SocketClient s=new SocketClient(register.getProbationary().getPort(),
        register.getModule().signMessage(m,register.getModule().getPrivateKey()),
        register.log);
}

public void sendProofs(){
        if(!this.register.honest){
            this.register.changeKeys();
```

```
        }
        ArrayList proofs=register.getProofs();
        String source=register.getID();
        String recipient=register.getMaster().getID();
        String type="proof";
        int port=register.getPort();
        Message m=new Message(source,recipient,proofs,type,
                              register.getModule().getPublicKey(),port);
        SocketClient s=new SocketClient(register.getMaster().getPort(),
        register.getModule().signMessage(m,register.getModule().getPrivateKey()),
                                         register.log);
}
```

5.4.3.3 Content Certificate Generation

Figure 5.12 depicts the structure of the system by showing the system's classes, some of their attributes, and the relationships between the classes for content authentication.

For this process, we apply the signature algorithm and the message digest function over a particular file in the system. We include the header of the main procedures initiated by the owner (already implemented in `C`):

1. Compute the file message digest using a SHA-2 hash function:

   ```
   int compute_hash(char *file, char *sha2);
   ```

2. Fill the content certificate with the file information:

   ```
   int new_certificate(char *sha2, char *id_content, char
   **ols, int nodes);
   ```

3. The owner signs the certificate:

   ```
   int sign_certificate(char *input,char *output);
   ```

   This function uses GnuPG v1.4.6 (GNU/Linux) and calls the following routine:

   ```
   gpg -output h(c).0 -clearsign h(c).c
   ```

   $h(c).c$ being the initial certificate C. In turn, after the owner includes his signature the certificate becomes $h(c).0$. Next section details the syntaxis to name successive signed certificates.

Figure 5.13 depicts the interaction between instanced objects at the content certificate generation stage.

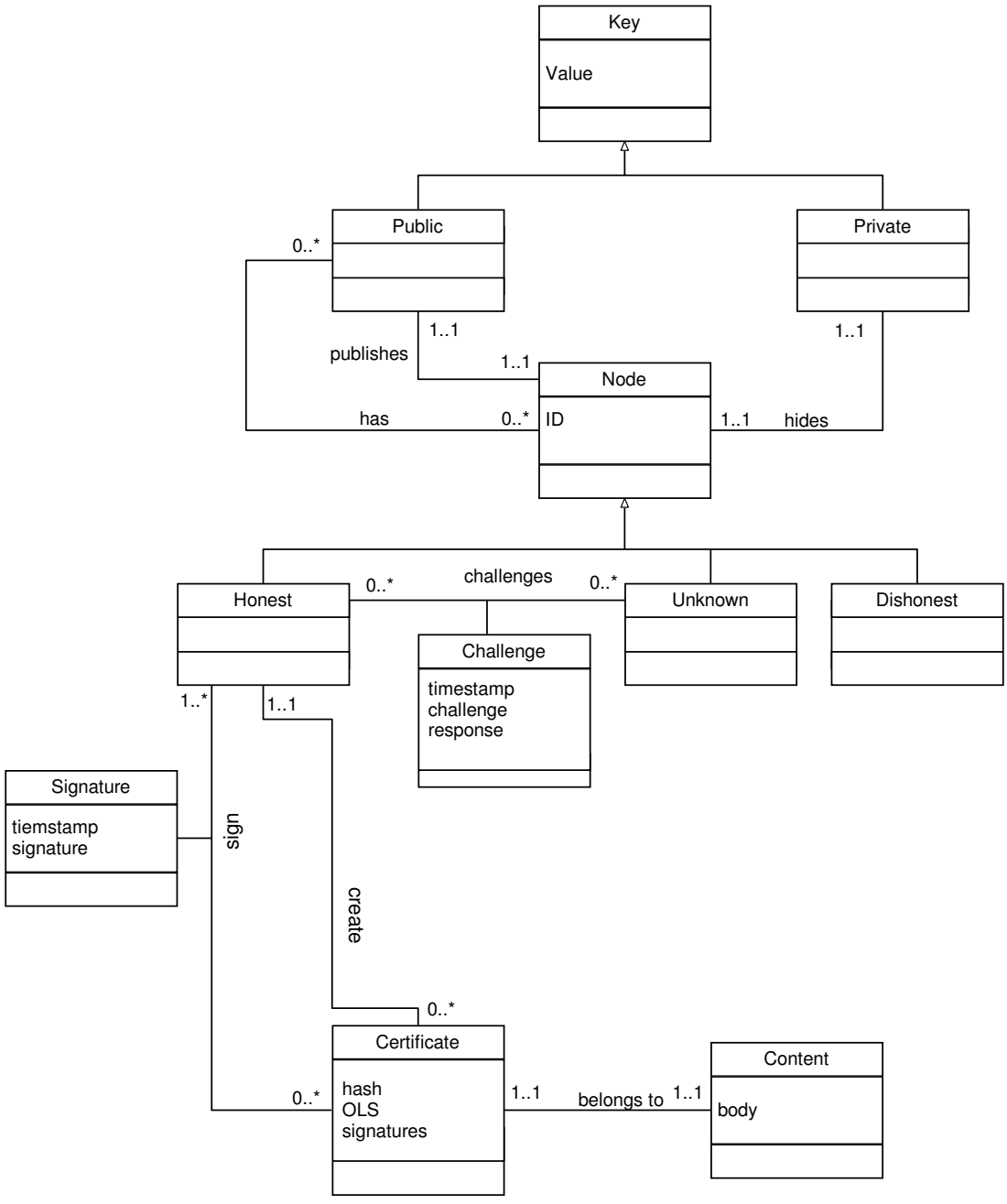


Figure 5.12: Content authentication protocol: Class diagram.

5.4.3.4 Content Certificate Signing Process

The signing process requires the following two actors and procedures (already implemented in `C`):

- Owner (initiator of a content authentication instance):

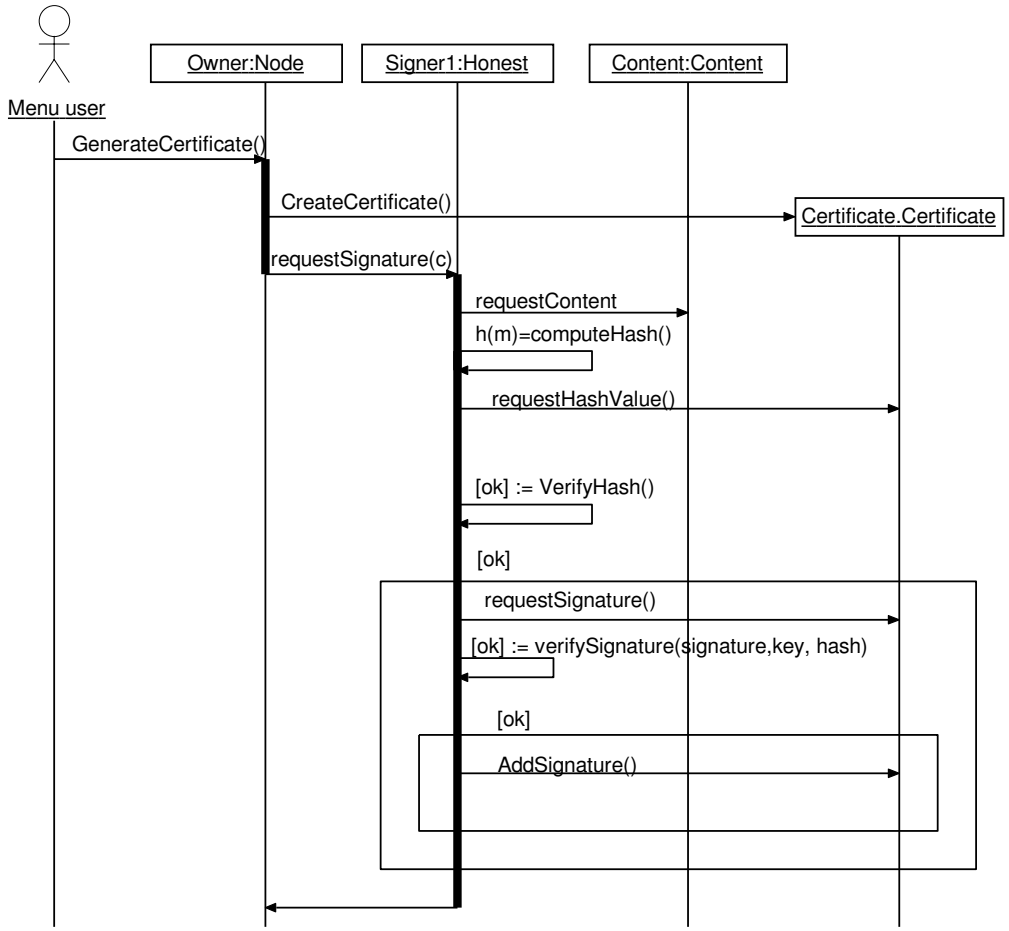


Figure 5.13: Content certificate generation: Object interaction.

1. Manages the signing process for all the nodes included in the OLS:

   ```
   int manage_signatures(int signers, char **ols, char
   *cert, char *doc, char *hash, int *node_error);
   ```

2. Manages the `put` command:

   ```
   int send(char *file, char *machine);
   ```

3. Requests for signature to signers:

   ```
   int request_sign(char *file, char *signer);
   ```

4. Manages the `get` command:

   ```
   int retrieve(char *file, char *signer)
   ```

5. Remote procedure to send a particular file:

   ```
   int * put_1_svc(args1 *argp, struct svc_req *rqstp);
   ```

6. Remote procedure to retrieve a particular file:

   ```
   char ** get_1_svc(arg2 *argp, struct svc_req *rqstp);
   ```

7. Remote procedure which returns a particular file size:

   ```
   int * size_1_svc(char **argp, struct svc_req *rqstp);
   ```

8. Remote procedure to request for the content certificate signature:

   ```
   int * sign_1_svc(char **argp, struct svc_req *rqstp);
   ```

- Signers:

1. Load a content certificate data:

   ```
   char **load_certificate(char *certificate, char
   *sha2_c, char *id_content, int *nodes);
   ```

2. Compute the message digest of a particular file:

   ```
   int compute_hash(char *file, char *sha2);
   ```

3. Compare two hash values:

   ```
   int verify_hash(char *sha2_c,char *sha2_d);
   ```

4. Verify the signatures in the content certificate:

   ```
   int verify_signatures(char *certificate,char **ols, int
   nodes, int *failed_node);
   ```

   We can use the following routine to verify the signature:

   ```
   gpg -output h(c).0 -decrypt h(c).1
   ```

5. Verify that the signatures in the content certificate are correct according to the order imposed by the OLS:

   ```
   int verify_sig_ols(char *buffer, char **ols, int
   nodes);
   ```

6. Check the local register of previous signed certificates:

   ```
   int verify_table(char *tableS, char *hash);
   ```

7. Add a signature to the content certificate:

   ```
   int sign_certificate(char *input,char *output);
   ```

8. Add a new entry in the local register of previous signed certificates:

   ```
   int add_S(char *hash);
   ```

The following table shows the sequence of certificate identification in order to distinguish each step in the content certificate signing process:

	Step	Certificate ID
1	Owner creates C'	$h(c).c$
2	$s_{n_0}(C')$	$h(c).0$
3	$s_{n_1}(C_0)$	$h(c).1$
4	$s_{n_2}(C_1)$	$h(c).2$
5	$s_{n_3}(C_2)$	$h(c).3$
6	$s_{n_k}(C_{k-1})$	$h(c).k = h(c)$
7	Certificate C_m	$h(c)$

5.4.3.5 Join Process

The Join process requires the following two actors and procedures (already implemented in `C`):

- Requester (initiator of an access request):

    1. Sends a request for a clearance to the corresponding owner, in order to gain access to the desire content. Requesters can obtain ownership information from the P2P search engine and the location service.

       ```
       int send_RQT (struct acc_req *rqsfrm, char *machine);
       ```

    2. Evaluates the certificate received from the owner after sending the corresponding request form.

       ```
       Initially this is empty.  A specific policy of
       acceptance/rejection should be inserted here.
       ```

    3. Verifies clearance and signatures in the authorization certificate:

       ```
       int verify_clearance (char *clearance, int
       *failed_node);
       ```

    4. Signs and returns the clearance confirmed:

       ```
       int sign_certificate(char *input,char *output);
       int *put_clearance(args1 *argp, char *machine);
       ```

    5. Add a new entry in the local register of authorization certificates.

       ```
       char **store_clearance(char *clearance, char *label,
       char *machine);
       ```

- Owner:

    1. After retrieving the request form sent by a certain requester, she checks her local register of contacted peers and the table of subscribers as well. On the other hand, if the requester is unknown, the owner must execute an instance of the public key authentication.

       ```
       int retrieve_RQT (struct acc_req *rqsfrm, char
       *machine);
       ```

       ```
       int verify_table(char *tableSs, char *hash);
       ```

       ```
       int verify_table(char *tableT, char *hash);
       ```

       ```
       If the requester is unknown, call Byzantine class:
       public void protocolar(SignedObject s_obj);
       ```

    2. Generates, signs and sends an authorization certificate (challenge-response mechanism in the subsection bellow.) Moreover, for simplifying the implementation, the owner will generate a credential for the requester by default. However, a specific policy of acceptance/rejection should be inserted here:

       ```
       char **load_clearance(char *clearance, char *label,
       char *machine);
       ```

       ```
       int *put_clearance(args1 *argp, char *machine);
       ```

    3. Verifies clearance and signature, signs again and returns to requester.

5.4.3.6 Updating Clearance Process

Upgrading a clearance relies on a challenge-response protocol, which is started by a particular request message sent by a particular subscriber to the owner:

```
int send_RQT (struct update_req *rqsfrm, char *machine);
```

Once the owner accepts the request, both engage in the following four-step protocol (implemented in `Java`):

1. The owner challenges requester. For implementation purposes, we can use basic cryptographic routines, as follows:

```
PuzzleMessage (Handler SourceID, Handler RequesterID, byte
content[], byte puzzle[], byte f[], BigInteger b1, BigInteger b2,
BigInteger b3, String c)
{       signature = f;
        n = b1;
        a = b2;
        t = b3;
        cryptogram = c;
}
//Puzzle helps requester to retrieve an authorization certificate
String dataPuzzle = puzzle.getPuzzle(SessionKey);

public String getPuzzle(SessionKey){
        int period = 7;
        int operations = 10000;
        int primesize = 1028;
        BigInteger p = new BigInteger(primesize,10,new Random());
        BigInteger q = new BigInteger(primesize,10,new Random());
        n = p.multiply(q);
        a = p;
        BigInteger fi = p.subtract(BigInteger.valueOf(1));
        fi = fi.multiply(q.subtract(BigInteger.valueOf(1)));
        t = BigInteger.valueOf(period * operations);
        BigInteger dos = BigInteger.valueOf(2);
        BigInteger e = dos.modPow(t,fi);
        BigInteger b = a.modPow(e,n);
        byte puzzlebody[] = getPuzzlebody(b.toByteArray());
        String puzzle=new String(SessionKey.getEncryption())+new String(puzzlebody);
        return puzzle;
}

// Puzzle is encrypted using requester's public key
byte encryptedPuzzle[] = encrypt(dataPuzzle.getBytes(),
                                 node.getPublicKey(requester));

byte signature[] = node.sign(dataPuzzle.getBytes());

Message = new PuzzleMessage(node.getID(),requester.getID(),...);
node.send(requesterID, message);

private byte[] getPuzzlebody(byte puzzle[])
{
        int numBytes = 96;
        byte body[] = new byte[numBytes];

        for(int i = 0;i < numBytes;i++){
            body[i] = puzzle[i];
        }
return body;
}
```

2. The requester computes the response, solving the challenge and sending it back:

```
public String solvePuzzle(BigInteger n, BigInteger a, BigInteger t,
                          String puzzle)
{       BigInteger dos = BigInteger.valueOf(2);
        BigInteger e = dos.pow(t.intValue());
        BigInteger b = a.modPow(e,n);
        byte solution[] = getPuzzlebody(b.toByteArray());
        String SessionKey = new String(getSolution(puzzle.getBytes(),solution));
        return SessionKey;
}
private byte[] getSolution(byte puzzle[],byte puzzlebody[])
{       int numBytes = puzzle.length - puzzlebody.length;
        byte solution[] = new byte[numBytes];
        for(int i = 0;i < numBytes;i++){
            solution[i] = puzzle[i];
        }
return solution;
}


\\On the other hand, in this stage, we may use trapdoor functions:
\\Assumptions:

\\   Key length is set on 128 bits -> unsigned char key[16];

   l = given trapdoors bytes (most significant) -> unsigned char w[16]
   -> w[0:l] trapdoor bits;

   Ciphers: AES -> aes_context ctx
   (using TEA algorithm: unsigned long key[4];)

for(i=0;i<l;i++)
     key[i]=w[i];

key[l:n-1]=0x00; k=0; stop=0;

while ( (!stop) && (k<(pow(2,32))) )
    {
      /* Decrypt Part */
      aes_set_key(&ctx, key, 128);
      aes_decrypt(&ctx, ibuf, obuf);

      /*For experiments we compare if the two strings are equal, strcmp returns 0.*/
      if (memcmp(ibuf, obuf, 16 )==0)
           stop=1;
      k++;

     /* Example: l = 128 - 32 (hidden bits) = 96*/

     key[12]=k>>24;
     key[13]=(k<<8)>>24;
     key[14]=(k<<16)>>24;
     key[15]=(k<<24)>>24;
```

```
     /* Try experiments for l= 96, 100, 104, 108*/
}// while
```

3. The owner verifies the response in her local registers of challenges sent to requesters. The owner generates and sends an upgraded clearance to the requester.

```
private void retrievePuzzleMessage(PuzzleMessage message)
{     if (message.getRecipientID().compareTo(node.getID()) == 0)
        {   // Verifies signature
            if (node.verify\_signature(message.getBytes(), message.getSignature(),
            node.getPublicKey(requester)))
            {  \\Check table of challenge-response
                if (message.getpuzzle().compareTo(node.getTablePuzzles(message.
                                                        getpuzzle())) == 0)
                 { \\Generate upgraded clearance
                   clearance= node.getClearance(message.getRecipientID);
                   clearance=reload\_clearance(clearance, clearance.getlabel + 1,...);
}} } }
```

4. The requester verifies the new clearance and locally stores it.

```
\\Requester can easily get the session key by means of solving the puzzle
byte puzzle[] = node.decrypt(message.getPuzzleCifrado());
Puzzle p = new Puzzle();
String key = p.solvePuzzle(message.getN(),message.getA(),message.getT(),new String(puzzle));
ReconstructorSessionKey reconstructor = new ReconstructorSessionKey(key.getBytes());

byte contentfile[] = descryptSessionKey(message.getContent(),reconstructor.recoverKey());
//See next section for recover the key from the trapdoor

if (contentfile != null)
{ Downloaded fd = new Donwloaded ("users\\" + node.getID().toString() +
                                      "\\" + message.getFile().toString());
fd.write(content);
}
```

It is interesting to examine to what extent the usage of this kind of effort-aware access control mechanism can be applicable and reliable in real networks. For instance by means of cryptographic puzzles based on trapdoor functions, we have found that more than 32 hidden bits would be impracticable.

The results on average are shown in Appendix B. Briefly, as the number of the revealed bits increases, the number of permutations decreases, so total exploration time as well. Moreover, the required time depends on peers' computational resources, i.e. computing power, memory size and speed.

5.4.3.7 Content Certificate Verification Process

After downloading, requesters should verify the content certificate, as follows (already implemented in `C`):

1. Load a content certificate data:

   ```
   char **load_certificate(char *certificate, char *sha2_c,
   char *id_content, int *nodes);
   ```

2. Compute the message digest of the downloaded file:

   ```
   int compute_hash(char *file, char *sha2);
   ```

3. Compare two hash values:

   ```
   int verify_hash(char *sha2_c,char *sha2_d);
   ```

4. Verify the signatures in the content certificate according to the OLS:

   ```
   int verify_signatures(char *certificate,char **ols, int
   nodes, int *failed_node);
   ```

   ```
   int verify_sig_ols(char *buffer, char **ols, int nodes);
   ```

   As an example, the sequence of verifications of a content certificate signed by three nodes may use:

   Verification of n_2 signature: `gpg -output h(c).1 -decrypt h(c).2`

   Verification of n_1 signature: `gpg -output h(c).0 -decrypt h(c).1`

   Verification of n_0 =owner signature: `gpg -verify h(c).0`

Moreover, Figure 5.14 depicts the interaction between instanced objects at the content certificate verification stage. Although it is not explicitly pointed out in the figure, if any signer's public key is unauthenticated, an instance of the public key authentication protocol is created for each one.

5.4.3.8 Content Access Process

We assume that requesters will not deviate from the Join protocol, and that their clients will correctly use the information enclosed into their authorization certificates to gain access to the content. On the other hand, the search engine

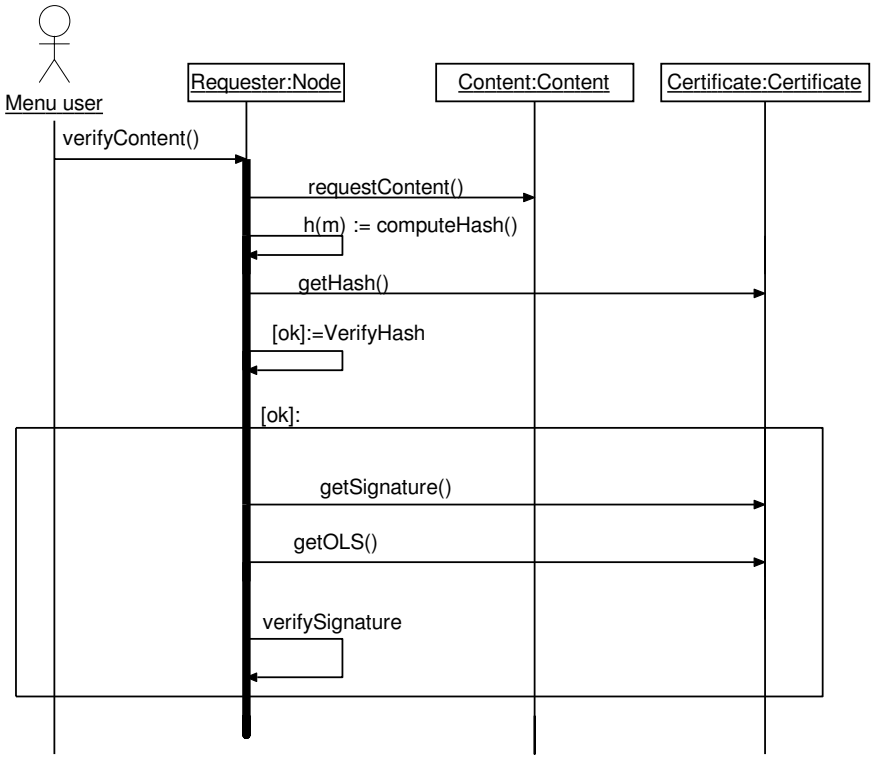


Figure 5.14: Content certificate verification: Object interaction.

will only show those available replicas stored by the owner and/or the authorized providers, who have an authorization certificate with the corresponding replication privileges issued by that owner.

The sequence of operations should be the following (designed in `Java`):

1. The requester formulates her query, using a descriptor of the desired content. From the query response, the requester can select one of the returned list of sources, i.e. either the owner or an authorized provider. First, the system must verify the providers' credential, and will only show delegated nodes.

```
QueryResponse(..., contentID, )
public String getQuery (String fileName){
  result = new QueryResponse();
  registers = net.searchSharedRegisters(new Id(fileName));
  for(int i = 0;i < registers.numBytes;i++){
      if(registers[i] != null){
         if ( node.verifyCredential(registers[i].getCredential()) &
             (registers[i].getClearance() >= registers[i].getContentLabel())
               & !owner.searchCRL(registers[i].getCertificateSN() )
                     {result[i] = registers.show();}
```

```
  }    }
...
```

2. For implementation purposes, the requester first selects a source at random (if the selected provider is unknown the system will visualize the message and show the direct shortcut.) Finally, the requester sends her credential to the source, as follows:

```
...
        int x = ((int) Math.random() * 5);
        String output = "";
        Node source = net.searchNode(result[x].getID());
        Message request = new downloadRequest(Id,result[x].getID(),node.getClearance());

        node.send(result[x].getID(), request);
        int counter = 0;
        do{
           try{
               wait(1000);
               counter += 1000;
           }
           catch(InterruptedException ie){
               System.out.println("Exception: Time-out");
           }
        }while(node.getThreadControl() == null && counter < 7000);

        if(node.getThreadControl() != null){
            do{
                try{
                    wait(1000);
                }
                catch(InterruptedException ie){
                    System.out.println("Exception: Time-out");
                }
           }while(node.getThreadControl().isAlive());

            sharedRegister sharedReg = registers.getRegister(result[x].getID());
            File f = new File(sharedReq.Download(Id).toString());
            if (f.exists()){
                output = "Successful Downloading";
            }
        }
        else{
             ...
        }
        node.asignThread(null);
        return output;
}
```

3. The provider client checks requester's clearance and, whether the verifications conclude with success, this process then securely sends the

content and the associated content certificate to the requester client.

```
private void retrieveDownloadRequest(DownloadRequest message)
{ if (message.getRecipientID().compareTo(node.getID()) == 0)
   { if ( node.verifyCredential(message.getClearance()) &
         (message.getClearance() >= node.getContentLabel(message.getId())) &
           !node.searchCRL(message.getCertificateSN()))
      {
        Message response = new downloadResponse(node.getContent(message.getId()),
                                    message.getSenderID(),node.getClearance());
        node.send(message.getSenderID(), response);
      }
          else
               \\reject download request
        }
```

5.4.3.9 Delegation and Revocation Processes

This subsection overviews the main routines for the proposed authorization services. As explained in the analysis of these processes, both are related to a new data structure called CRL. We have designed in `Java` programming language the sequence of tasks, as follows:

1. For implementation purposes, the originator of delegation services is the issuer, i.e. a content owner who already has, at least, a subscriber belonging to a particular security level. This subscriber, who already has the corresponding clearance, will receive a replication request from the issuer. Once the subscriber accepts, both engage in Join process. The successful end of Join involves the following two targets: a new clearance with the replication privilege for the new provider, together with the sending of the issuer's CRL.

```
public String sendDelegation (String level) {
  subscriber = new Node(node.getSubscriber(level));
  Message request = new Repli_Rqt (node.getID(),subscriber.getID());
  node.send(subscriber.getID(), request);
  int counter = 0;
  do{
     try{
         wait(1000);
         counter += 1000;
         }
     catch(InterruptedException ie){System.out.println("Exception: Time-out");
           }
  }while(node.getThreadControl() == null && counter < 7000);
```

```
if(node.getThreadControl() != null){
  do{
     try{
         wait(1000);
        }
     catch(InterruptedException ie){System.out.println("Exception: Time-out");
              }
  }while(node.getThreadControl().isAlive());
\\Suscriber send the ACCEPT message
\\retrieve_msg ();
\\call Join with Replication parameter not null
Message crlMessage = new crlMessage ();
node.send(subscriber.getID(), crlMessage);
}}
```

2. In principle, the inclusion of a new entry in the CRL depends on the issuer suspicions of a compromised clearance, a trust-based accusation model, or even the expiration of the validity period. For a start, we test the revocation procedure by choosing a certain subscriber at random. A new entry will be added into the CRL: the current date, and the subscriber's certificate serial number.

5.5 Deployment and Testing

In the current development stage, we have tested the functional requirements of both content authentication and access control processes.

5.5.1 Functional Testing

In order to evaluate the system's compliance with its specified requirements, we have tested the resulted software product in the terms describes in Tables 5.4 and 5.5.

Next, we show the results for test "1. Generate C_{doc}" using three signers (see Fig. 5.15.) Furthermore, we include some screenshots resulted from different executions of the public key authentication protocol implanted.

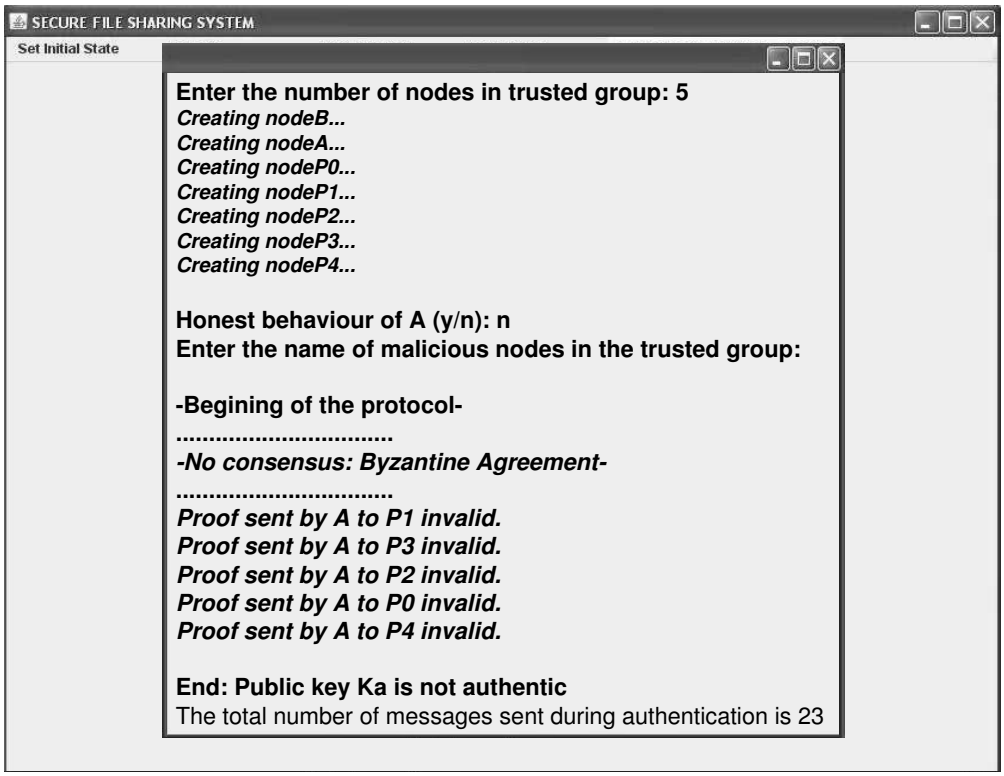
Concretely, Figure 5.15 is a Linux console screenshot resulted from running a simulation of the content authentication module. We can distinguish the following: the content of the file labeled as "doc" (line (1)), line (2) prints the procedure call along with the parameter expressing the target content, and the obtained content certificate for "doc". This certificate is identified using the

Test	**Expected result**	**Final result**	**Comments**
Content Certificate Generation Stage			
Generate C_m	New content certificate for m	✓	Visualization of C_m (Fig. 5.15)
Generate $C_{m'}$ where m' is missing	Message "m' is not a current file"	✓	Visualization of the error
m previously signed	1st signer notifies a duplication message	✓	We repeat this test
Incorrect signature	The corresponding signer notifies an "Error in signature verification" error message.	✓	Visualization of the error and the failed signature
Content Certificate Verification Stage			
Correct C_m	Correct content certificate for m	✓	Visualization of C_m
Modification over m	Message error: "C_m is missing. m is possibly altered."	✓	Visualization of the error
Incorrect signature	Message error: "Error in signature verification".	✓	Visualization of the error and the failed signature
Hashes do not fit	Message error: "Hashes do not fit".	✓	Visualization of the error
Public Key Authentication Stage			
Correct scenario	A node B initiates a protocol instance to authenticate A using some of her trusted nodes	✓	Visualization of traces (Fig. 5.16)
A is dishonest	There is no consensus and node B gets evidence of A's dishonest behavior	✓	Visualization of traces (Table 5.6-(a))
One or more P_i are dishonest	There is no consensus. Node B checks A's evidences, and initiates a "Byzantine agreement" phase	✓	Visualization of traces (Table 5.6-(b))

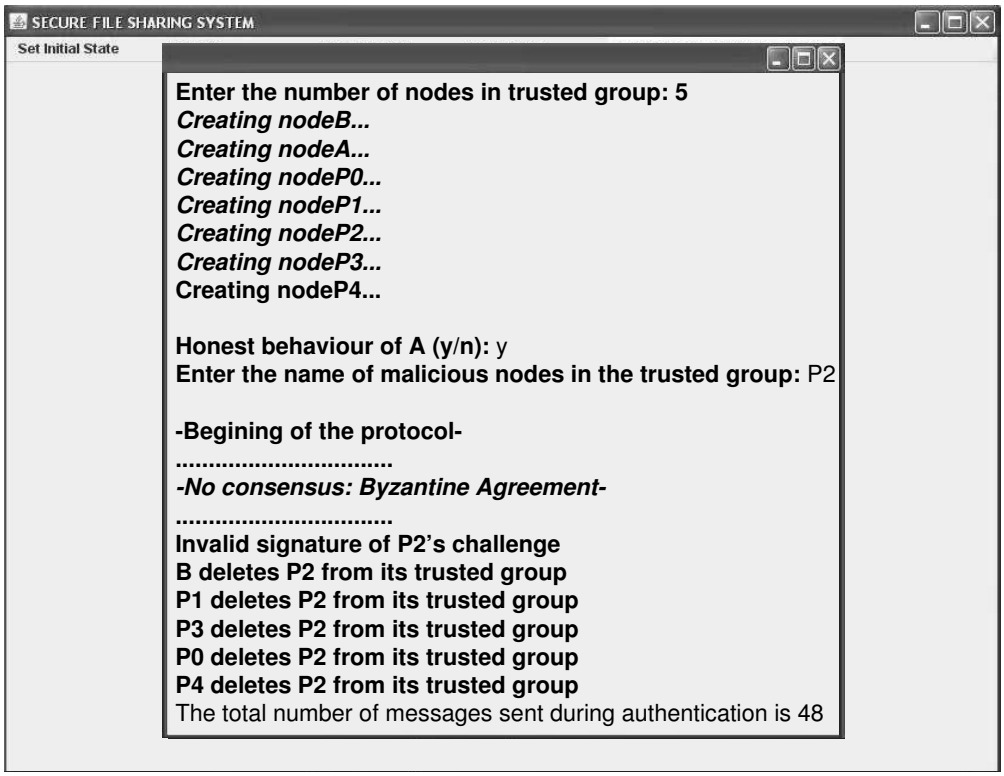
Table 5.4: Functional requirement testing: Content and Public key authentication schemes.

Test	**Expected result**	**Final result**	**Comments**
Join Stage			
Generate C_B^A	New clearance for B issued by A	✓	Visualization of C_B^A)
Incorrect signature	The corresponding node notifies an "Error in signature verification" error message.	✓	Visualization of the error and the failed signature
Upgrade C_B^A	Updated clearance for B issued by A	✓	Visualization of C_B^A
Generate/Upgrade C_B^A when B is unknown or malicious	Issuer A notifies the rejection	✓	Visualization of the rejection
Content Access Stage			
Correct C_B^A	Access the content	✓	Visualization of (m, C_m)
Get m from a downgraded C_B^A	Choose between Join/Update	✓	Visualization of both Join/Update
Get m from an unknown provider	Authenticate her public key	✓	Visualization of the authentication message and selection button

Table 5.5: Functional requirement testing: Access Control scheme.



(a) Public Key Authentication: A is dishonest.



(b) Public Key Authentication: P_i is dishonest.

Table 5.6: Public Key Authentication: Functional testing.

```
(1) \$ cat doc

Esto es una prueba

(2) \$./p2p_cliente doc

(3) \$cat 4f50502679fab5f643887cbbdc8a8eb690e1f23c

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA2

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA2

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA2

(4) $4f50502679fab5f643887cbbdc8a8eb690e1f23c#manten@127

.0.0.1#doc#manten@127.0.0.1&razorin@127.0.0.1&mmunoz@127.0.0.1$

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iQCVAwUBRlACAIUVUdNfyEGaAQJQ8QQAnYH6UL72swlhOSRaOCbNH/I2PYtM3OjD
+Y+Q/g/iziZ622fd1qD+FI4s/cb65pC8izcviIbbmYlGz4Ohtv2AKwwcEl1gY/lz
CGGXXsjbUk5yAhDg9SaV6diMIF0J7OKwtNuJ+kmFXI2LPHocYAbSVIc9QIU3QWb7
sVh11VarKsM= =WdJJ

-----END PGP SIGNATURE-----

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iQCVAwUBRlACAIUVUdNfyEGaAQKaMgQAgnJxy6oVl7A2J0plfK7QozzAp41kin+3
5XngBaumeGR/O/jYRO6ao4cSQaVsjGb9/jCznUYf9UADl4fOTRgGeUS/OoLn1a8/
vMUnMBUUQwhkWVBy0yns9o4B3zC6ZfVfcZAnHPyzluvoYOMZvkdpWuWAmEXdgGcm
V86S1feUkSA= =Px/y

-----END PGP SIGNATURE-----

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iQCVAwUBRlACAIUVUdNfyEGaAQIdkgQAnM2hmsL1M5EPUDsoJSAebl67rmTZ+WrP
ZBwcohKBG26uikCiJ+FdlOw2hFQgx3Veq95r0L3kvIWFJQNx8EmJsjksTO0wg0sU
JlUoQ9z7cNfb+ltBHmSv+EgrX3tA0akOBvWkrXzDCos9NNKQossZXlhwyQg5D3ZU
ZbG58JmA4YY= =RoVK

-----END PGP SIGNATURE-----
```

Figure 5.15: Execution of an scenario where three signers generate the certificate for a particular content.

content hash, as lines (3) and (4) illustrates. The program prints some traces showing the signing process, the content hash and signers' identifiers, and the delimitation of signers' signatures.

After the successful evaluation of the content authentication module, a pre-built executable version is accessible from `SourceForge.net` projects at

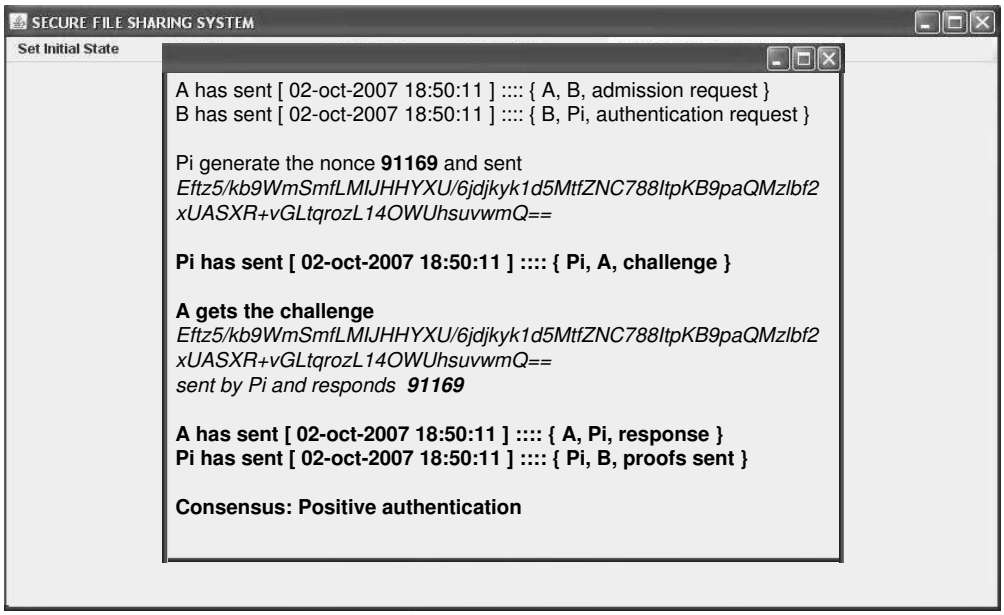


Figure 5.16: Public Key Authentication: Testing a basic scenario.

Figure 5.17: Main menu.

`http://sourceforge.net/projects/p2pcontent-auth/`.

5.5.2 Interface Design

We tried to design a user interface as close as possible to a common file sharing client, including well-known functionalities along with the security features offered by our proposal.

Figure 5.17 shows the main menu of the software application. We can identify the most important task in file sharing such as `Conect/Disconect`, and `Search` content by a key term, together with our content authentication scheme and authorization options: `Publish` contents with the corresponding content certificate generation, `Verify` content certificates, *Request for a authorization* certificate, among others. We include some screenshots and summarize the

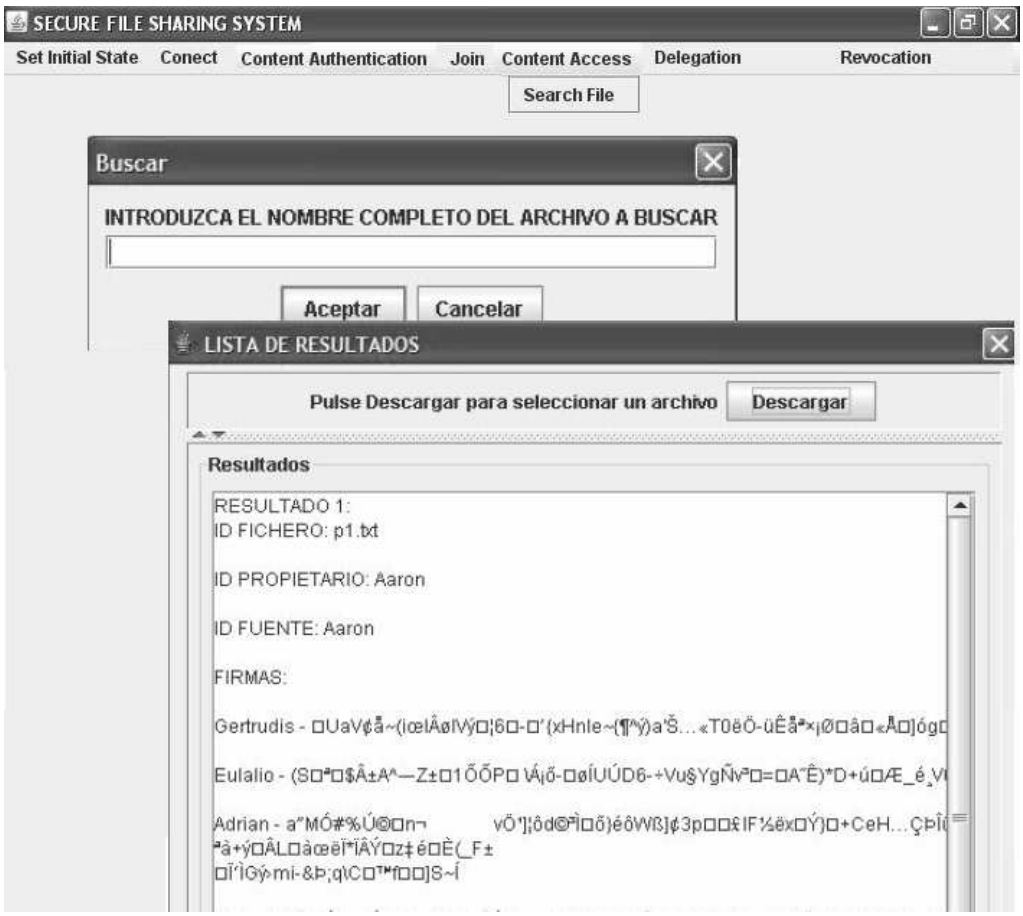


Figure 5.18: Search menu.

called procedure on each submenu as follows:

- The `Conect/Disconect` menu allows users to easily join/quit the P2P network. This function should be supported by the overlay. Instead, our implementation includes an additional menu called `Set Initial State` which allows users to specify a network scenario (number of nodes and their nature, i.e. honest or non-collaborative) for the simulation. During this process, an identification and a new key-pair are generated for each node (see Fig. 5.16 and figures at Table 5.6.)

- Our secure file sharing client also includes a common option to `Search File`. Users can type a descriptor term of the desired content. The search response should only show references (managed by the overlay) of shared contents, and authorized sources. However, downloading is managed by

Figure 5.19: Content access menu.

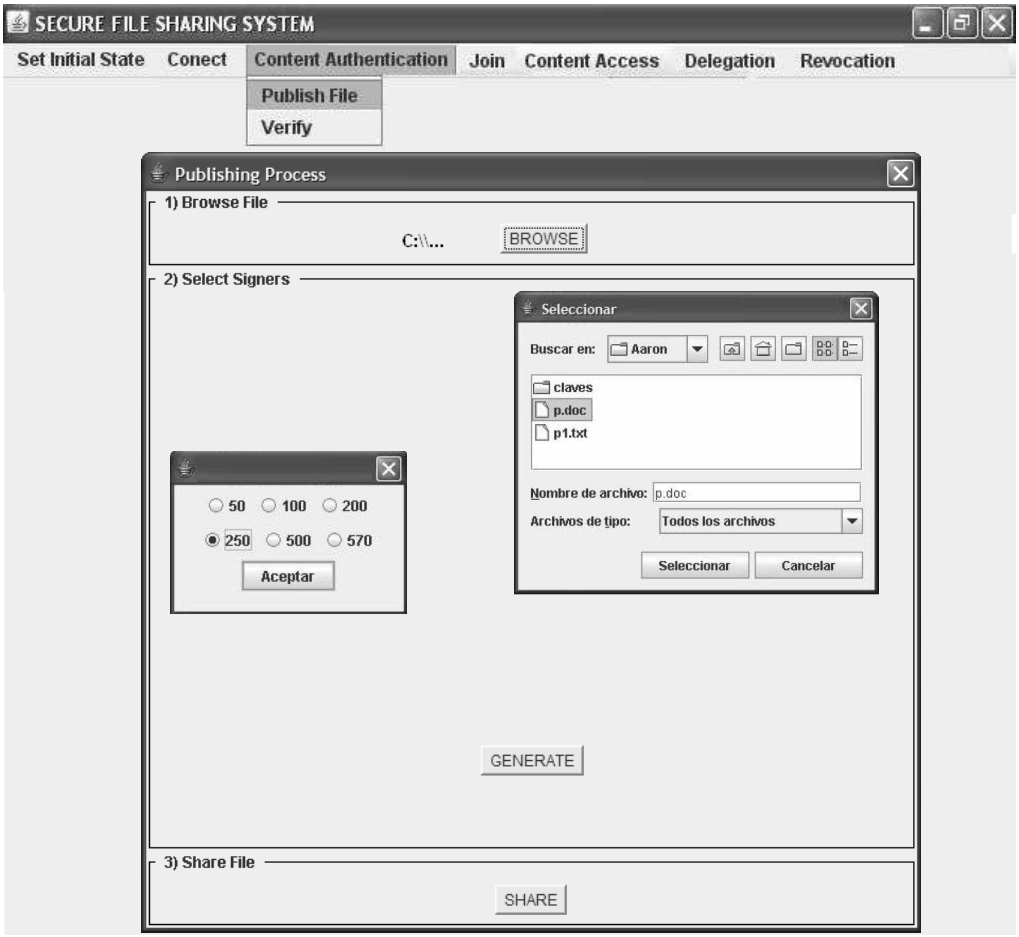


Figure 5.20: Content authentication menu.

the `Content Access` menu, which automatically presents and requires the corresponding credentials without involving the user (see Figs. 5.18 and 5.19.) The lack of authorized providers is visually notified by a warning popup. On the other hand, the content will be downloaded if only if the requester node has an appropriate authorization certificate. Similarly, the system will inform about the reason of the denied access,

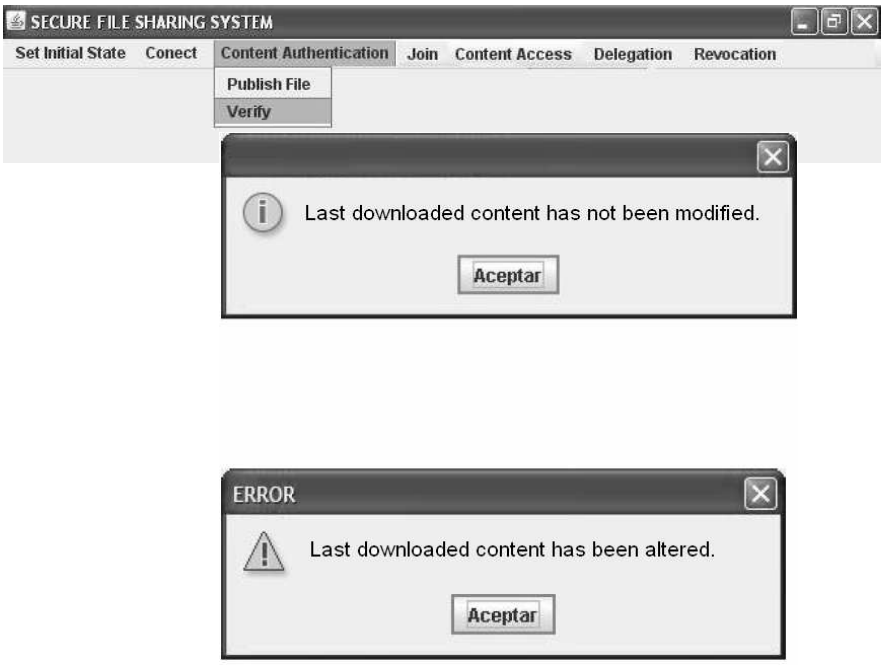


Figure 5.21: Content certificate verification menu.

and provide the corresponding alternative routine.

- Owners who want to publish original content must start the `Content Authentication` process from the corresponding toolbar, as shown in Fig. 5.20. The system will show a browser popup window to select the file. Regarding the functional evaluation of this protocol, we can choose among different number of signers. When the content certificate signing procedure ends, the system shows all the content certificate fields. On the other hand, the verification of the downloaded content certificate will show a popup window accordingly (as shown in Fig. 5.21.)

- The `Join`(/Update) menu allows users to request an authorization certificate to a certain owner by means of our Join subprotocol. The evaluation of this request by the owner's client is empty and, therefore, the three-step join subprotocol just shows the final authorization certificate gain by this user.

- The `Delegation` menu allows owners to request subscribers for replicating a certain content. The user can select one of her subscribers according to the particular security level defined by the content. The system, once

Figure 5.22: Join menu.

receiving the affirmative response from the subscriber, automatically informs the user, updates the subscriber's clearance with the replication label, and also sends the corresponding CRL to the new provider. Finally, issuers, i.e. owners who have previously assigned clearances, can revoke a certain subscriber's authorization certificate by means of the `Revocation` menu. Similarly Join and Delegation, this function first displays the subscriber database. The user selection on a nodeID automatically adds a new entry in the CRL with the serial number of this node's certificate. The system shows the updated CRL.

5.6 Project Management

We have summarized the research schedule in a time estimation presented in the Table 5.23. By means of Gantt charts, we illustrate all the components comprise the work breakdown structure of this thesis and the set of planned outcomes that we have appropriately met. As we can see in the figures, the graphics track and chart the research project time-line from the initial study of the state-of-the-art on P2P Security to the final stages of this thesis, i.e. the implementation, evaluation of the proposal and final writing of this document.

The figure in the bottom shows the time-line but using percent complete indications. As shown, in October last year we had completed a few stages such as the publication of the initial contribution of this thesis i.e. the content authentication scheme, while others (although already started) remain incomplete until the first half of this year 2008.

The security analysis of the entire proposal has been the longest (in time), but note that the implementation, the performance evaluation, and the effi-

ciency analysis had to wait for the DEA exam. These tasks have advanced in parallel together with the writing process.

Summarizing, we can detail the achievements on each phase, as follows:

1. **State of the Art and Problem Formulation. October 2005 – October 2006.**

   1.1 Study and analysis of state of the art on P2P Security.

   1.2 Explore an initial contribution: Content authentication protocol [93].

   1.3 Specification of a refined approach, providing access control using proof-of-work mechanisms.

   1.4 Begin an informal security analysis of main contributions [92].

   1.5 Advances Studies Degree "DEA".

2. **Initial Implementation and Analysis. November 2006 – August 2007.**

   2.1 Design and development of the proposed model. Initial implementation of the secure content distribution scheme based on proof-of-work.

   2.2 Study the protocol's viability through an extensive performance and efficiency analysis.

   2.3 Analysis from a Game Theory approach.

   2.4 Writing in parallel.

   2.5 Thesis Proposal and Pre-defense presentation.

3. **Testing and Finishing dissertation writing. December 2007 – May 2008.**

   3.1 Evaluation: Testing Security Vulnerabilities and Performance overheads.

   3.2 Aggregation and analysis of research results.

   3.3 Based on feedback from pre-defense Ph.D Committee: Finalize the written dissertation. Ph.D. Defense.

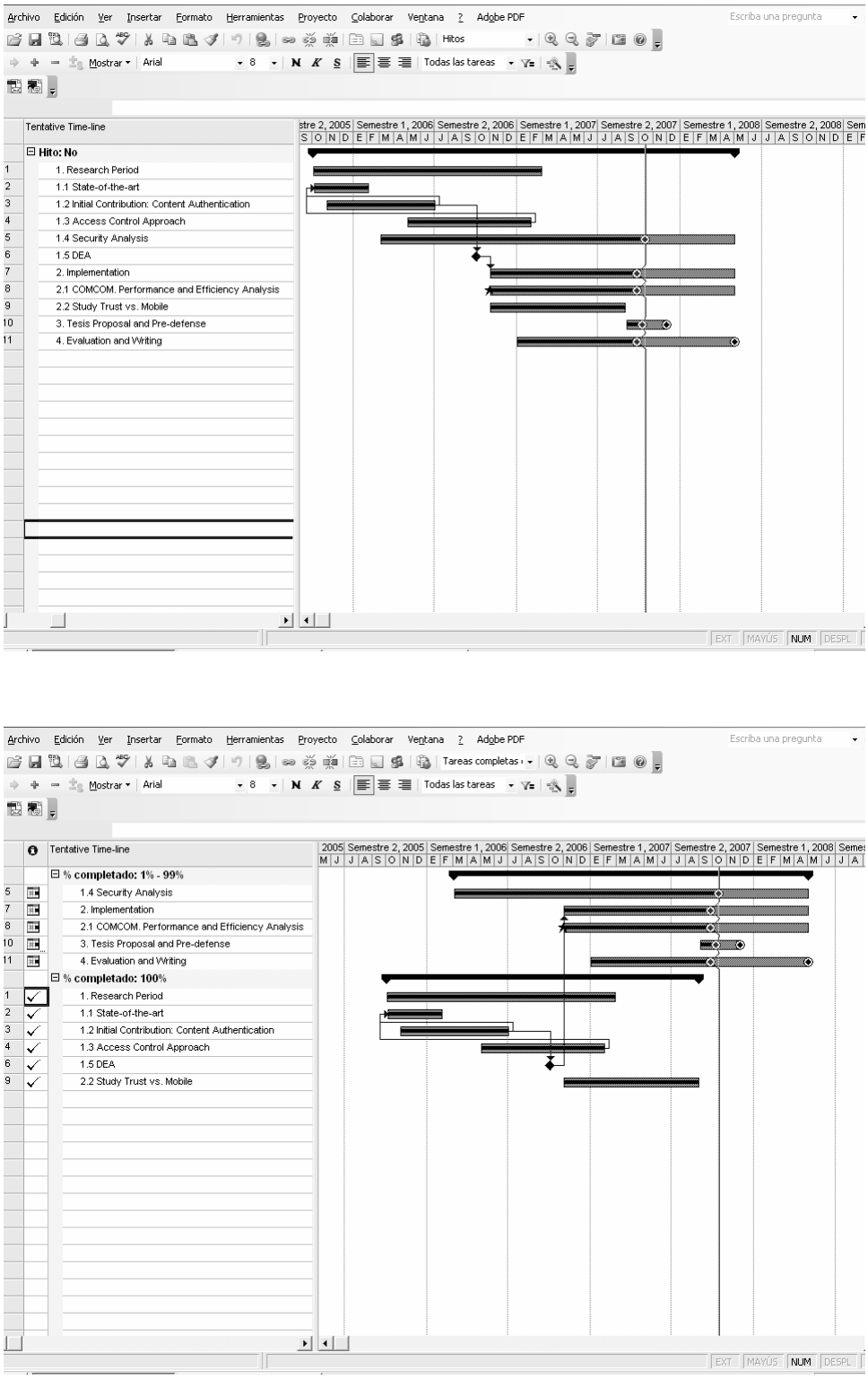


Figure 5.23: Brief illustration of the thesis schedule through these Gantt charts.

Chapter 6

Conclusions and Future Work

We conclude this thesis by summarizing our main contributions and pointing out the conclusions reached and some lines for future research.

6.1 Contributions

We have introduced a secure content distribution protocol especially oriented to P2P file-sharing systems, which are environments characterized by node transience and the lack of any centralized authority. The proposed solution provides a content authentication scheme which allows secure content replication among peers, thus ensuring the integrity of the published contents. Non-authorized accesses are prevented by ensuring that only a user having the proper security clearance will be able to decrypt the downloaded content. Moreover, content owners can control who accesses their contents by discretionally issuing clearances in the form of authorization certificates. On the other hand, we have investigated ways to extend the authentication and access control services here presented to provide more complex capabilities, such as privilege delegation and revocation. In fact, replication is also controlled since the inclusion of a delegation protocol assures that only authorized providers will be able to distribute others' replicas. Furthermore, by assuming that every integrating node will exhibit a malicious behavior, even if they have always behaved correctly in the past, we have addressed a revocation scheme as well.

In summary, the key goals of this research achieved so far are summarized below:

1. Explore and formalize the need of basic security services, such as authentication and authorization, in pure P2P and ad hoc environments. A state-of-the-art survey of P2P security has been presented in Chapter 2.

2. Design a secure content distribution protocol for fully decentralized networks. The proposed scheme consists of five main components:

   (a) Content Authentication protocol (Chapter 3.)

   (b) Access Control protocol (Chapter 4) which includes the following subprotocols:

      i. Join subprotocol.

      ii. Content Access subprotocol.

      iii. Delegation subprotocol.

      iv. Revocation subprotocol.

3. Analyze and evaluate the P2P file sharing protocol obtained. We obtain two main results:

   - A security analysis. The implemented application ensures the security requirements aforementioned to construct several testing scenarios for the proposal.
   - A performance evaluation. Security services of any kind have significant performance overhead, more so in case of fully distributed networks. Results are quantified in practical terms concerning time requests, vulnerabilities, threats and other factors. Moreover, standard statistical tests have been used to draw conclusions regarding performance differences.

4. Design and implant a software implementation in order to evaluate and analyze the functional requirements of our proposal. We have used both Java APIs and C language for the implementation (see Chapter 5.)

5. Design and implant a Public Key Authentication Protocol based on Byzantine agreement aimed at analyzing the performance of our proposal (see Appendix A.)

6. Design and evaluation of a challenge-response mechanism which provides interesting capabilities for content access control, for instance by applying proof-of-work technique, such as cryptographic puzzles (see Appendix B.)

6.1.1 Comparison with Related Work

We include an informal comparison of our entire scheme with other works proposed so far by the research community which look into similar objectives. Table 6.1 summarizes the analysis done according to if the proposed models apply the basic security properties that our model has considered.

In particular, we can distinguish that those schemes based on threshold cryptography are the most similar to our proposal regarding the features provided. However, we try to meet fairness and address DoS attacks. Moreover, threshold-based schemes rely generally on a secret dealer, while our protocols are fully decentralized.

On the other hand, the benefits of cooperative storage and serving offered by earlier P2P file sharing systems (such as Napster [4], Gnutella [3], and Freenet [1]) have motivated numerous works on P2P storage, particularly the approaches presented in [57, 35, 88]. These works agree with our proposal on one of our main contributions, i.e. these systems are limited to read-only or single-publisher data, in the sense that only the original publisher of each piece of data can modify it. Subsequently, recent works extend this limitation to a P2P storage system to build a read/write file system that multiple users can share. Moreover, the most important requirement of these approaches, including our implementation, is that they can operate in a relatively open decentralized P2P environment because they do not require participants to trust each other. We have identified, however, important differences, and some advantages of our work. For instance, the simplicity of the implementation and deployment of our proposal makes it more suitable for standard, highly transient users, e.g. a user who uses an Ivy file system from multiple hosts concurrently must have one log per host. Furthermore, our approach is not overlay-dependant, i.e. both structured or unstructured architectures can support our model. For example, CFS requires the Chord location service based on distributed hash tables, and similarly SFSRO file system must deal with the periodically updating of a global database.

	Replication Control	**Integrity and Authentication**		**Access Control**		
	[34],[81],*Rank*, P-GRID [8]	Trust/ Rep. Ss.	Threshold Crypto.	[67], [7]	[96] [122], [53], [78], [90], [106]	**Our proposal**
Identification						
Pseudospoofing	✓	–	✓	✓	✓ Partially	✓
Blacklisting	×	✓	–	–	–	–
Service Avail.						
DoS	✓	×	×	×	×	△
Churn	✓	×	–	△	△	–
Dishonest Nodes						
Cheating	×	✓ Partially	✓	×	✓	✓
Sybil	×	–	×	×	△	×
Free-riding	✓(*Ranking*)	–	–	–	–	✓
Man-in-the-Middle	×	✓ Partially	✓	×	✓ Partially	✓
Collusion	×	×	✓	✓	×	✓ H. majority
Fairness	✓	–	×	✓	–	✓
Properties						
Availability	✓	✓	×	×	×	–
Integrity	✓(GRID)	△	✓	×	✓	✓
Authentication	✓(GRID)	△	✓	✓ Partially	✓	✓
Confidentiality	×	×	✓	–	–	✓
Anonymity	×	×	✓	✓	×	✓
Authorization	×	×	–	✓	✓	✓
Decentralization	×	×	×	✓ Partially	✓	✓

Table 6.1: Informal comparison between our proposal and similar works according to if they apply detection and protection mechanisms (✓), when none applied (×), only detection (△), or when N/A (–)

Regarding access control, our proposal does not require any central entities, decision points, or global storage structures either, e.g. one or more LDAP directories to store attribute certificates such as those used by GRID infrastructures and PERMIS, a recent X.509 role-based PMI presented in [29].

6.2 Publications

Some of the contributions presented in this thesis have already been published in various peer-reviewed conference proceedings and journals. Below we list them.

6.2.1 Conference/Workshop Publications

1. **Bayesian Analysis of Secure P2P Sharing Protocols.**

   E. Palomar, A. Alcaide, J.M.E. Tapiador and J.C. Hernandez-Castro.

   Proc. of the 2nd Int. OTM Symposium on Information Security.

   LNCS Vol. 4804, pp. 1701–1717. Springer-Verlag, 2007.

2. **A P2P File-Sharing Protocol based on Cryptographic Puzzles.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of the 5th Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing.

   LNCS (To appear.) Springer-Verlag, 2008.

3. **Dealing with Sporadic Strangers,or the (Un)Suitability of Trust for Mobile P2P Security.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of the 4th Int. Workshop on P2P Data Management, Security and Trust.

   IEEE Press., pp. 779–783, 2007.

4. **Protocolo para la Autenticación de Contenidos en Redes P2P.**

   E. Palomar, A. Ribagorda, M.V. Muñoz and D. Onoro.

   Proc. of 2nd Simposio sobre Seguridad Informática, CEDI 2007, pp. 143–150.

5. **Certificate-based Access Control in Pure P2P Networks.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of the 6th Int. Conf. on Peer-to-Peer Computing

   IEEE Press., pp. 177–184, September 2006.

6. **A Protocol for Secure Content Distribution in Pure P2P Networks.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of 3rd Int. Workshop on P2P Data Management, Security and Trust.

   IEEE Press., pp. 712–716, September 2006.

7. **Aspectos de Seguridad en Redes P2P: Un Análisis Comparativo.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of the RECSI 2006 (IX Reunión Española sobre Criptología y Seguridad de la Información), pp. 674–688, Barcelona (Spain), September (2006.)

8. **Security in P2P Networks: Survey and Research Directions.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of IFIP Int. Conf. on Embedded and Ubiquitous Computing (NCUS 2006 Workshop.)

   LNCS Vol. 4097, pp. 183–192, August 2006.

9. **A P2P Content Authentication Protocol Based on Byzantine Agreement.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Proc. of the Int. Conf. on Emerging Trends in Information and Communication Security (ETRICS 2006.)

   LNCS Vol. 3995, pp. 60–72, June 2006.

6.2.2 Journals Articles

1. **Secure Content Access and Replication in Pure P2P Networks.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Computer Communications vol. 31/2, pp. 266–279. Elsevier 2008.

2. **Estudio y Análisis de la Distribución de Contenidos Falsos en una Red P2P.**

   E. Palomar, J.L. Cadiz, P. Peris and A. Ribagorda.

   ALI Base Informatica vol. 42, pp. 37–41, 2006.

6.2.3 Book Chapters

1. **Cooperative Security in Peer-to-Peer and Mobile Ad Hoc Networks.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Cooperative Wireless Communications.

   (To be published by Auerbach Publications, Taylor&Francis Group, 2008.)

2. **Secure Content Distribution in Pure Peer-to-Peer and Ad Hoc Networks.**

   E. Palomar, J.M.E. Tapiador, J.C. Hernandez-Castro and A. Ribagorda.

   Handbook of Research on Secure Multimedia Distribution.

   (To be published by IGI Global, 2008.)

6.3 Research Plan and Future Work

Our future work includes several research lines, as follows:

1. First, we will validate the scheme in terms of detecting practical attacks and measuring performance overheads.

2. In principle, the process for generating content certificates may be performed by using a *multisignature* scheme [23]. The large number of

needed signers and the communication overhead could probably be improved with the use of other approaches, notably those based in *secret sharing* schemes. This is generally a more efficient way to gather up a number of signatures over a given document than by doing it sequentially. The use of Threshold Cryptography in P2P systems for reaching consensus is also an interesting research line that will be tackled in future work.

3. A second line is the adaptation of a bootstrapping phase aimed at protecting all the messages exchanged between our protocol entities by the network/transport layer, e.g. using TLS. This fact, therefore, prevents from several attacks such as those mounted by an attacker who can listen the messages transmitted during some phase of the protocol.

4. A third line is related to the study of specific ways according to which the clearance granting/updating process can be integrated with other mechanisms which serve as rewarding nodes appropriately for their investment in the system. We have identified two alternatives:

   (a) Using a classic TMS. This can greatly reduce the computational cost of the proposal by decreasing the number of cryptographic operations needed. However, trust and security are concepts somewhat related but definitely different in nature. As a consequence, delegating security processes to trust systems is something that should be done carefully and in the appropriate contexts.

   (b) Using a e-commerce model. Regarding the fair sharing aspect of the common used P2P systems, the research community has dealt with fairness by means of applying multiple mechanisms such as incentives, credits, micropayments, electronic checks, to name a few. Now, our solution addresses fairness based on a proof-of-work model which requires a computational effort through cryptographic puzzles. Similarly, new platforms for e-commerce are working on adapting P2P technology to create different kind of marketplaces not just for content, but for any source in general, for instance money. A totally innovative framework is the emerging commercial P2P lending systems where an individual or group of individuals can lend directly to a borrower [82].

5. Timestamps, and therefore time synchronization, are also open issues for taking into account in our proposal.

6. Both the Sybil attack and the use of pseudonyms are challenging issues for peer joining in fully decentralized systems. A CA can enforce randomly chosen node identifiers and limit each node to have only one alias. Nevertheless, the absence on such an entity generally forces the need of a consensus agreement which depends on the collaboration within a social group. This group, for example using threshold cryptography, can all together decide whether a new joining node is genuine or sybil. This is still an interesting open issue. A simple, but limited solution is by not allowing any member to revoke or change its identification item.

7. In MANETs, the inefficient performance of several ubiquitous security works proposed so far is currently an obstacle to the acceptance and usage of several cryptographic models. The analysis of the suitability of our protocol in such environments is very challenging in order to verify common standards. We expect to go beyond simulations and integrate the proposed content authentication protocol in pervasive environments. This will surely result in conceiving additional ways to combine trust schemes and traditional cryptographic protocols for ubiquitous P2P networks.

8. Finally, since our entire protocol basically relies on the collaboration among a fraction of peers in the system, we can formally analyze the community behavior in order to empirically observe a networking social pattern. This is possible since our model establishes a rational content access control by means of a challenge-response mechanism, whereby nodes may achieve good reputation and privileges. Contrary to classic trust systems where trust decisions are directly or indirectly given by nodes' past behavior, our scheme uses cryptographic proofs of work to discourage selfish behavior and to reward cooperation. Thus, applying rational models, such as the Prisoners' Dilemma [48], we can describe a formal framework to analyze some security aspects of the protocol itself, and also the dynamics created when nodes interact following the protocol description, i.e. play by the rules. In particular, by means of Game Theory we can model the dynamics of the community based on evaluating

the requester's trustworthiness and collaboration state through several probability distribution functions, which give us different community profiles. This raises an interesting issue, as we are able to consider non-collaborative nodes and to measure the effect they might have on the overall system performance. Moreover, we can also measure how nodes can dynamically adapt their strategies to highly transient communities.

References

[1] The free network project web page. Website, 2007. `http://freenetproject.org/`.

[2] The gift-fasttrack project home page. Website, 2007. `http://developer.berlios.de/projects/gift-fasttrack/`.

[3] The gnutella home page. Website, 2007. `http://www.gnutella.com/`.

[4] The napster home page. Website, 2007. `http://www.napster.com`.

[5] Speed comparison of popular crypto algorithms. Website, 2007. `http://www.eskimo.com/~weidai/benchmarks.html`.

[6] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.

[7] M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately hard, memory-bound functions. *ACM Transactions on Internet Technology*, 5(2):299–327, May 2005.

[8] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. *ACM SIGMOD Rec.*, 32(3):29–33, 2001.

[9] K. Aberer and M. Hauswirth. An overview on peer-to-peer information systems. In *Proceedings of Workshop on Distributed Data and Structures*, pages 171–188, Paris, France, March 2002.

[10] W.J. Adams and N.J. Davis. Tms: a trust management system for access control in dynamic collaborative environments. In *Proceedings of the 25nd Int. Conf. on Performance, Computing, and Communications*, Arizona, USA, April 2006. IEEE.

[11] A. Alcaide, J.M. Estevez-Tapiador, J.C. Hernandez-Castro, and A. Ribagorda. An extended model of rational exchange based on dynamic games of imperfect information. In *Proceedings of the Int. Conf. on Emerging Trends in Inf. and Comm. Security*, pages 396–408, Germany, June 2006. Springer-Verlag.

[12] F. Almenárez, A. Marín, C. Campo, and C. García R. Ptm: A pervasive trust management model for dynamic open environments. In *Proceedings of the First Workshop on Pervasive Security and Trust at MobiQuitous*, Boston, USA, August 2004.

[13] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, , and D. Werthimer. Seti@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, November 2002.

[14] G. Arboit, C. Crepeau, C.R. Davis, and M. Maheswaran. A localized certificate revocation scheme for mobile ad hoc networks. *Ad Hoc Networks*, 6(1):17–31, January 2008.

[15] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in adhoc wireless networks. In *Proceedings of the Symposium on Network and Distributed Systems Security*, February 2002.

[16] S. Balfe, A. Lakhani, and K. Paterson. Trusted computing: Providing security for peer-to-peer networks. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing*, pages 117–124, Konstanz,Germany, August 2005. IEEE.

[17] M. Barborak, A. Dahbura, and M. Malek. The consensus problem in fault-tolerant computing. *ACM Comput. Surv.*, 25(2):171–220, 1993.

[18] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Proceedings of the 22nd Annual Int. Cryptology Conf. on Advances in Cryptology*, pages 162–177, London, UK, 2002. Springer-Verlag.

[19] M. Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the 2nd annual ACM symposium on Principles of distributed computing*, pages 27–30. ACM, 1983.

[20] Q. Bi, G.L. Zysman, and H. Menkes. Wireless mobile communications at the start of the 21st century. *IEEE Communications Magazine*, 39:110–116, January 2001.

[21] B. Biskupski, J. Dowling, and J. Sacha. Properties and mechanisms of self-organizing manet and p2p systems. *ACM Transactions on Autonomous and Adaptive Systems*, 2, March 2007.

[22] A. Boukerch, L. Xu, and K. EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30:2413–2427, September 2007.

[23] C. Boyd. *Digital multisignatures*, pages 241–246. H. Baker and F. Piper (Eds.). Clarendon Press, 1989.

[24] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the confidant protocol (cooperation of nodes fairness in distributed adhoc networks). In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, Switzerland, June 2002.

[25] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in p2p systems. In *Proceedings of the 3rd Int. Conf. on Peer-to-Peer Computing*, pages 48–56, Linköping, Sweden, September 2003. IEEE Computer Society.

[26] L. Buttyán and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Mobile Networks and Applications*, 8:579–592, 2003.

[27] S. Capkun, L. Buttyán, and J-P. Hubaux. Self-organized public key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, Jan-Mar 2003.

[28] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of the 5th symposium on Operating systems design and implementation*, Boston, Massachusetts, December 2002.

[29] D.W. Chadwick and A. Otenko. The permis x.509 role based privilege management infrastructure. *Future Gener. Comput. Syst.*, 19(2):277–289, 2003.

[30] J. Cheng, Y. Li, W. Jiao, and J. Ma. A utility-based auction cooperation incentive mechanism in peer-to-peer network. In *Proceedings of the 2nd Int. Symp. on Network-Centric Ubiquitous Systems (EUC Workshop)*, pages 11–21, Seoul, Korea, August 2006. Springer-Verlag.

[31] M. Conti, E. Gregori, and G. Maselli. Cooperation issues in mobile ad hoc networks. In *Proceedings of the 24th Int. Conf. on Distributed Computing Systems Workshops*, 2004.

[32] M. Conti, E. Gregori, and G. Turi. Towards scalable p2p computing for mobile ad hoc networks. In *Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 109–113, Orlando,USA, March 2004. IEEE.

[33] G. Di Crescenzo, R. Geb, and G.R. Arce. Threshold cryptography in mobile ad hoc networks under minimal topology and setup assumptions. *Ad Hoc Networks*, 5:63–75, January 2007.

[34] F.M. Cuenca-Acuna, C. Peery, R.P. Martin, and T.D. Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pages 236–246, Washington,USA, June 2003. IEEE.

[35] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215. ACM, 2001.

[36] J. Daemen and V. Rijmen. *The Design of Rijndael: AES The Advanced Encryption Standard*. Springer-Verlag, 2002.

[37] E. Damiani, S. De Capitani, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, Washington,USA, November 2002. ACM.

[38] L.A. DaSilva and V. Srivastava. Node participation in ad hoc and peer-to-peer networks: A game-theoretic formulation. In *Proceedings of the Wireless and Comm. and Networking Conf.*, New Orleans, USA, March 2005. IEEE Computer Society.

[39] N. Daswani and H. García-Molina. Pong-cache poisoning in guess. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 98–109, Washington, USA, October 2004. ACM.

[40] D. Defigueiredo, A. Garcia, and B. Kramer. Analysis of peer-to-peer network security using gnutella. Technical report, 2002.

[41] Yvo Desmedt. Some recent research aspects of threshold cryptography. In *1st. Int. Workshop on Information Security, (ISW'97)*, pages 158–173. Springer-Verlag, 1997.

[42] R. Dingledine, N. Mathewson, and P. Syverson. The free haven project: Reputation in p2p anonymity systems. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, USA, July 2003.

[43] D. Djenouri, L. Khelladi, and A.N. Badache. A survey of security issues in mobile ad hoc and sensor networks. *IEEE Communications Surveys and Tutorials*, 7(4):2–28, 2005.

[44] J.R. Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, pages 251–260, Cambridge, USA, March 2002. Springer-Verlag.

[45] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147. Springer-Verlag, 1992.

[46] W.K. Edwards. Policies and roles in collaborative applications. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 11–20, Boston, Massachusetts, November 1996. ACM Press.

[47] W.K. Edwards. Using speakeasy for ad hoc peer-to-peer collaboration. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 256–265, New Orleans, USA, November 2002. ACM.

[48] T. Ellis and X. Yao. Evolving cooperation in the non-iterated prisoner's dilemma: A social network inspired approach. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 25–28, Singapore, September 2007.

[49] L. Eschenauer, V.D. Gligor, and J. Baras. On trust establishment in mobile ad-hoc networks. In *Proceedings of 10th Int. workshop on Security Protocols*, pages 47–66. Springer Berlin/Heidelberg, April 2004.

[50] S. Farrell and R. Housley. An internet attribute certificate profile for authorization. *RFC 3281*, April 2002.

[51] M. Feldman and J. Chuang. Overcoming free-riding behavior in peer-to-peer systems. *ACM Sigecom Exchanges*, 6(1):41–50, July 2005.

[52] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM Conf. on Electronic commerce*, pages 17–20, New York, USA, May 2004.

[53] P. Fenkam, S. Dustdar, E. Kirda, G. Reif, and H. Gall. Towards an access control system for mobile peer-to-peer collaborative environments. In *Proceedings of the 11th IEEE International Workshops on Enabling Technologies*, pages 95–102, Pittsburgh, USA, June 2002.

[54] A. Fiat and M. Naor. Broadcast encryption. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pages 480–491, California, USA, August 1994.

[55] G. Fox. Peer-to-peer networks. *Computing in Science & Engineering*, 3(3), May 2001.

[56] M. Freedman and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 193–206, Washingtion,USA, November 2002. ACM.

[57] K. Fu, M.F. Kaashoek, and D. Mazières. Fast and secure distributed read-only file system. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*, pages 13–13. USENIX Association, 2000.

[58] R. Gupta and A.K. Somani. Game theory as a tool to strategize as well as predict nodes behavior in peer-to-peer networks. In *Proceedings of the 11th Int. Conf. on Parallel and Distributed Systems*, pages 244–249, Fukuoka, Japan, July 2005. IEEE Computer Society.

[59] M. Haenggi and D. Puccinelli. Routing in ad hoc networks: A case for long hops. *IEEE Communications Magazine*, pages 93–101, October 2005.

[60] D. Hales and O. Babaoglu. Towards automatic social bootstrapping of peer-to-peer protocols. *SIGOPS Oper. Syst. Rev.*, 40(3):56–60, 2006.

[61] M. Haque and S.I. Ahamed. Security in pervasive computing: Current status and open issues. *International Journal of Network Security*, 3(3):203–214, 2006.

[62] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–14, CA, USA, 2000. IEEE Computer Society.

[63] R. Housley, W. Polk, W. Ford, and D. Solo. Internet x.509 public key infrastructure, certificate and certificate revocation list (crl) profile. *RFC 3280*, April 2002.

[64] V.C. Hu, D.F. Ferraiolo, and D.R. Kuhn. Assessment of access control systems. Technical report, September 2006.

[65] Y. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, 2004.

[66] J.P. Hubaux, L. Buttyán, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–155, Long Beach, USA, 2001.

[67] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In *Proceedings of the Networks and Distributed Security Systems*, pages 151–165, California, USA, February 1999.

[68] K. Kane and J.C. Browne. On classifying access control implementations for distributed systems. In *Proceedings of the 11th ACM symposium on Access control models and technologies*, pages 29–38. ACM, 2006.

[69] Y. Kim, W.C. Lau, M.C. Chuah, and J.H. Chao. Packetscore: Statistical-based overload control against distributed denial-of-service attacks. In *Proceedings of the 23rd Conference of the IEEE Communications Society, Infocom*, Hong Kong, China, March 2004. IEEE.

[70] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Proceedings of the 9th International Conference on Network Protocols*, pages 251–260, California, USA, November 2001. IEEE.

[71] L. Lamport, R. Shostak, and M. Pease. The byzantine general problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[72] R. Lepro. Cardea: Dynamic access control in distributed systems. Technical report, November 2003.

[73] B.N. Levine and C. Shields. Hordes: A protocol for anonymous communication over the internet. *Computer Security*, 10(3):213–240, 2002.

[74] N. Li, J.C. Mitchell, and W.H. Winsborough. Beyond proof-of-compliance: Security analysis in trust management. *Journal of the ACM*, 52(3):474–514, May 2005.

[75] C. Lin, V. Varadharajan, Y. Wang, and Y. Mu. On the design of a new trust model for mobile agent security. In *Proceedings of the 1st Int. Conf. on Trust and Privacy in Digital Business*, pages 60–69, Zaragoza, Spain, September 2004. Springer Verlag.

[76] W. K. Lin, D. M. Chiu, and Y. B. Lee. Erasure code replication revisited. In *Proceeding of the 4th IEEE Int. Conf. on Peer-to-Peer Computing*, August 2004.

[77] J. Lopez, R. Oppliger, and G. Pernul. Authentication and authorization infrastructures (aais): A comparative survey. *Computer Communications*, 27(16):1608–1616, October 2004.

[78] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah. First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, pages 25–37. ACM, 2003.

[79] H. Luo and S. Lu. Ubiquitous and robust authentication services for ad hoc wireless networks. Technical report, 2000.

[80] R.T.B. Ma, S.C.M. Lee, J.C.S. Lui, and D.K.Y. Yau. A game theoretic approach to provide incentive and service differentiation in p2p networks. In *Proceedings of the joint Int. Conf. on Measurement and modeling of computer systems*, pages 189–198, New York, USA, June 2004. ACM Press.

[81] P. Maniatis, T.J. Giuli, M. Roussopoulos, D.S.H. Rosenthal, and M. Baker. Impeding attrition attacks in p2p systems. In *Proceedings of the 11th ACM SIGOPS European Workshop*, Leuven, Belgium, September 2004. ACM.

[82] Prosper Marketplace. Personal & small business loans at prosper.com - the p2p lending marketplace, March 2008. http://www.prosper.com/.

[83] S. Marti and H. Garcia-Molina. dentity crisis: anonymity vs reputation in p2p systems. In *Proceedings of the 3rd IEEE Peer-to-Peer Computing*, September 2003.

[84] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MobiCom*, Boston, August 2000.

[85] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[86] A.J. Menezes, P.C. Vaz Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*, chapter Capter 10. Identification and Entity Authentication, pages 385–424. CRC Press, 1996.

[87] J.L. Morant, A. Ribagorda, and J. Sancho. *Seguridad y Protección de la Información*, pages 227–290. Editorial Centro Estudios R. Areces, 1994.

[88] A. Muthitacharoen, R. Morris, T.M. Gil, and B. Chen. Ivy: A read/write peer-to-peer file system. In *Proceedings of 5th Symposium on Operating Systems Design and Implementation*, 2002.

[89] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd annual ACM symposium on Theory of computing*, pages 427–437, Baltimore, Maryland, 1990.

[90] M. Narasimha, G. Tsudik, and J.H. Yi. On the utility of distributed cryptography in p2p and manets: The case of membership control. In *Proceedings of the 11th IEEE International Conference on Network Protocols*, pages 336–345, Atlanta, USA, November 2003.

[91] M. Oguchi, Y. Nakatsuka, and C. Tomizawa. A proposal of user authentication and a content distribution mechanism using p2p connection over a mobile ad hoc network. In *Proceedings of the IASTED International Conference on Communication Systems and Networks*, pages 65–69, Marbella, Spain, September 2004.

[92] E. Palomar, J.M. Estevez-Tapiador, J.C. Hernandez-Castro, and A. Ribagorda. Certificate-based access control in pure p2p networks. In *Proceedings of the 6th International Conference on Peer-to-Peer Computing*, pages 177–184, Cambridge, UK, September 2006. IEEE, IEEE.

[93] E. Palomar, J.M. Estevez-Tapiador, J.C. Hernandez-Castro, and A. Ribagorda. A p2p content authentication scheme based on byzantine agreement. In *Proceedings of the Int. Conf. on Emerging Trends in Inf. and Comm. Security*, pages 60–72, Germany, June 2006. Springer-Verlag.

[94] E. Palomar, J.M. Estevez-Tapiador, J.C. Hernandez-Castro, and A. Ribagorda. Secure content access and replication in pure p2p networks. *Computer Communications*, 31(2):266–279, February 2008.

[95] Th.G. Papaioannou and G.D. Stamoulis. Reputation-based policies that provide the right incentives in peer-to-peer environments. *Computer Networks*, 50(4):563–578, 2006.

[96] J.S. Park and J. Hwang. Role-based access control for collaborative enterprise in peer-to-peer computing environments. In *Proceedings of the 8h ACM symposium on Access control models and technologies*, pages 93–99. ACM, 2003.

[97] V. Pathak and L. Iftode. Byzantine fault tolerant public key authentication in peer-to-peer systems. *Computer Networks*, 50(4):579–596, March 2006.

[98] M.O. Pervaiz, M. Cardei, and J. Wu. Routing security in ad hoc wireless networks. *Network Security*, 2005.

[99] S.P. Ratnasamy. A scalable content-addressable network. Technical report, Berkeley,USA, 2002.

[100] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, April 1998.

[101] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001. ACM.

[102] A.D. Rubin. Method for the secure distribution of electronic files in a distributed environment. patent number: 5,638,446, June 1997.

[103] M.D. Russell. Tinyness: An overview of tea and related ciphers, February 2004. http://www-users.cs.york.ac.uk/ matthew/TEA/TEA.html.

[104] B. Ryabko and A. Fionov. Basics of contemporary cryptography for it practitioners. *World Scientific*, 2005.

[105] G. Sakaryan, H. Unger, and U. Lechner. About the value of virtual communities in p2p networks. In *Proceedings of the 3rd International School and Symposium*, pages 170–185, Guadalajara, Mexico, January 2004.

[106] R. Sandhu and Xs Zhang. Peer-to-peer access control architecture using trusted computing technology. In *Proceedings of the 10h ACM symposium on Access control models and technologies*, pages 147–158. ACM, 2005.

[107] R.S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, 1993.

[108] R.S. Sandhu and Pierangela Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[109] R.S. Sandhu and X. Zhang. Peer-to-peer access control architecture using trusted computing technology. In *Proceedings of the 10th ACM symposium on Access control models and technologies*, pages 147–158, Stockholm, Sweden, June 2005. ACM.

[110] K. Sanzgiri, B. Dahilly, B.N. Levine, C. Shieldsz, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE Int. Conf. on Network Protocols*, 2002.

[111] Stuart E. Schechter, Rachel A. Greenstadt, and Michael D. Smith. Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment. *Economics of Information Security*, 12:1568–2633, 2004.

[112] A.A. Selcuk, E. Uzun, and M.R. Pariente. A reputation-based trust management system for p2p networks. In *Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 251–258, Chicago, USA, April 2004.

[113] A. Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology*, 196:47–53, 1984.

[114] J. Shneidman and D.C. Parkes. Rationality and self-interest in peer to peer networks. In *Proceedings of the IPTPS*, pages 139Ű–148. Springer-Verlag, 2003.

[115] B. Sieka, A. Kshemkalyani, and M. Singhal. On the security of polling protocols in peer-to-peer systems. In *Proceedings of the 4th IEEE Inter-*

national Conference on Peer-to-Peer Computing, pages 36–44, Zurich, Switzerland, August 2004. IEEE.

[116] A. Singh and L. Liu. Trustme: Anonymous management of trust relationships in decentralized p2p systems. In *Proceedings of the 3rd IEEE Int. Conf. on Peer-to-Peer Computing*, September 2003.

[117] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings of IEEE Infocom*, 2003.

[118] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM Conference*, pages 149–160, San Diego, USA, August 2001. ACM.

[119] K. Stoupa, A. Vakali, F. Li, and I. Tsoukalas. Xml-based revocation and delegation in a distributed environment. In *Proceedings of the EDBT 2004 Workshops, LNCS 3268*, pages 299–308. Springer-Verlag, 2004.

[120] G. Theodorakopoulos and J.S. Baras. Trust evaluation in ad hoc networks. In *Proceedings of the ACM Workshop on Wireless Security*, 2004.

[121] W. Tolone, G. Ahn, T. Pai, and S. Hong. Access control in collaborative systems. *ACM Computing Surveys*, 37(1):29–41, March 2005.

[122] H. Tran, M. Hitchens, V. Varadharajan, and P. Watters. A trust based access control framework for p2p file-sharing systems. In *Proceedings of the 38th Hawaii Int. Conf. on System Sciences*. IEEE, January 2005.

[123] C. Wang and L. Xiao. An effective p2p search scheme to exploit file sharing heterogeneity. *IEEE Transactions on Parallel and Distributed Systems*, 18(2):145–157, February 2007.

[124] W. Wang and T. Stransky. Stateless key distribution for secure intra and inter-group multicast in mobile wireless network. *Computer Networks*, 51:4303–4321, October 2007.

[125] B. Yang, T. Condie, S. Kamvar, and H. Garcia-Molina. Non-cooperation in competitive p2p networks. In *Proceedings of the 25th IEEE Int. Conf. on Distributed Computing Systems*, 2005.

[126] H. Yang, X. Meng, and S. Lu. Self-organized network layer security in mobile ad hoc networks. In *Proceedings of the 1st ACM Workshop on Wireless Security*, pages 11–20, Atlanta, USA, 2002.

[127] S. Yi and R. Kravets. Moca: Mobile certificate authority for wireless ad hoc networks. In *Proceedings of the 2nd Annual PKI Research Workshop*, 2003.

[128] F. Zambonelli, M. Gleizesb, M. Mameia, and R. Tolksdorf. Spray computers: Explorations in self-organization. *Pervasive and Mobile Computing*, 1:1–20, March 2005.

[129] L. Zhang, G. Ahn, and B. Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, August 2003.

[130] X. Zhang, S. Chen, and R. Sandhu. Enhancing data authenticity and integrity in p2p systems. *IEEE Internet Computing*, pages 42–49, November–December 2005.

[131] Y. Zhang, L. Lin, and J. Huai. Balancing trust and incentive in peer-to-peer collaborative system. *Int. Journal of Network Security*, 5(1):73–81, July 2007.

[132] B.Y. Zhao, J. Kubiatowicz, and A.D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, 2001.

[133] F. Zhou, L. Zhuang, B.Y. Zhao, L. Huang, A.D. Joseph, and J.Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proceedings of the ACM International Middleware Conference*, pages 1–20, Rio de Janeiro, Brazil, June 2003. ACM.

[134] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[135] B. Zhu, S. Jajodia, and M.S. Kankanhalli. Building trust in peer-to-peer systems: a review. *International Journal of Security and Networks*, 1(1/2):103–112, 2006.

[136] P. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, Massachusetts, 1995.

Appendix A

Empirical Analysis of Public Key Authentication Scheme

This appendix presents an empirical analysis obtained as a result of the implantation of the public key authentication protocol proposed by Pathak and Iftode in [97] and briefly described in Section 2.2.3. We have implemented this proposal in `C`, as explained in Section 5.4.3.2, considering the corresponding messages formats and entity roles. Experiments carried out on a distributed environment give us a practical framework to analyze the scheme considering both the communication effort required by the nodes, as well as the computational (especially cryptographic) cost of the protocol.

A.1 Communication Overhead

The communication cost incurred in public key authentication protocol can be theoretically computed considering the amount of participants that have taken part in a given protocol instance. Similarly, the simulation helps us to corroborate the theoretic approach.

We have identified two levels of complexity according to whether the *Byzantine agreement* stage is needed or not.

A.1.1 Cost of Consensus

When all the participants are honest, we can simply count the messages that they exchange. In particular, each node receives the following messages:

1. Node B receives a admission request message from A, and one containing the proof-of-possession returned by each $P_i/i \in \{1 \dots k\}$.

2. Each node P_i receives an authentication request message and one challenge response from A.

3. Node A receives k challenge messages, one from each P_i.

The total number of messages then follows the expression below:

$$k + (k + 1) + 2k = 4k + 1$$

The complexity of this process is clearly lineal, $\mathcal{O}(k)$, and it is graphically expressed in Fig. A.1-(a).

A.1.2 Cost of Byzantine Agreement

When reaching consensus is not possible, the protocol immerses in the Byzantine agreement stage. Generally, this fact implies that at least one of the participants has behaved maliciously. The number of messages exchanged therefore increases in order for the parties to clarify the situation.

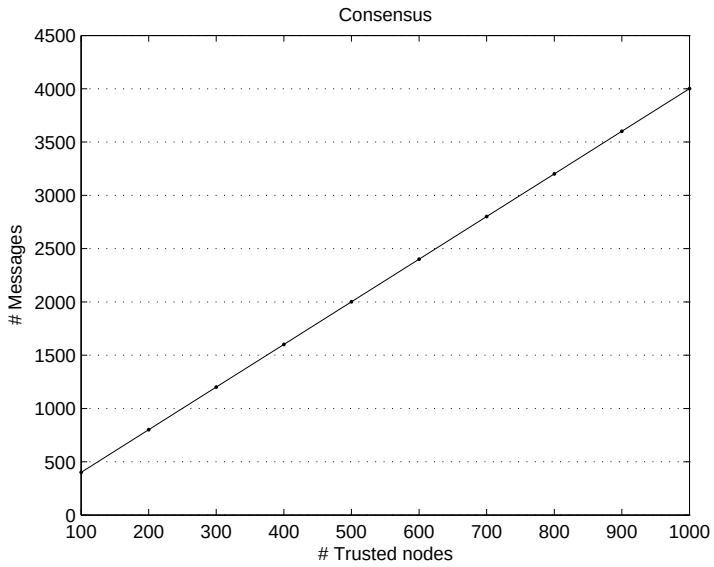
We can identify two scenarios. On the one hand, A may misbehave and, on the other hand, a supposed trusted node P_i may mount a particular attack.

A.1.2.1 Case 1: A is Dishonest

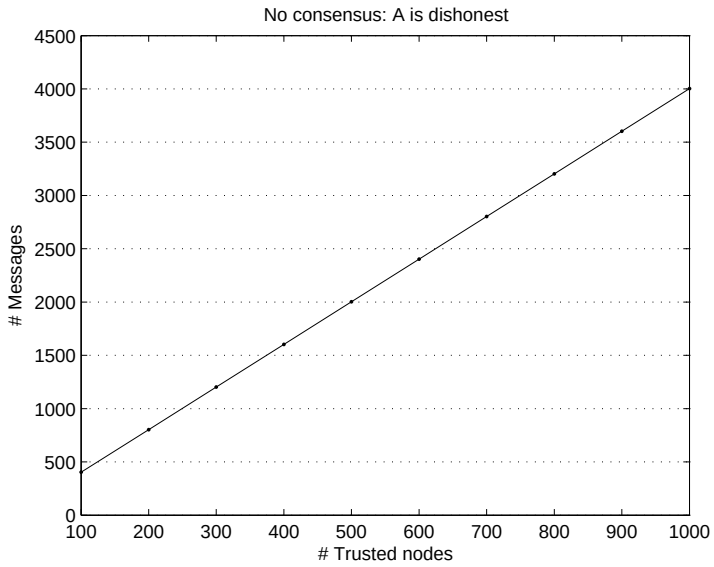
We can simulate a spoofing attack upon a dishonest node who sends an admission request regarding his public key. The attacker tries to masquerade A responding challenges as well.

Contrarily to the attacker's intentions, the protocol participants (if honest) will realize of such a malicious action at the challenge response step, due to the fact that the attacker cannot respond correctly to the received challenges. Thus, the proof-of-possession will be rejected during the Byzantine agreement phase.

Similarly to the previous case, we can count the messages resulted in this situation. The complexity is the same, $\mathcal{O}(k)$, since we have only two more messages, $(4k+3)$ regarding B's requests of proof to A', and her corresponding response.



(a)



(b)

Figure A.1: (a) Communication cost of reaching consensus in Public Key Authentication; and (b) when node A is revealed as dishonest.

In Figure A.1-(b), we can see how the cost of key authentication increases according to the number of implicated honest nodes.

A.1.2.2 Case 2: At Least One P_i is Dishonest

When B announces a Byzantine fault, all participants share their proof-of-possessions (the pairs challenge-response.) Therefore, each participant adds a message `proofs` to the rest of nodes, and B one communicating the start of the Byzantine phase.

The total number of messages involved follows the expression below:

$$4k + 3 + k((k-1)+1) = k^2 + 4k + 3$$

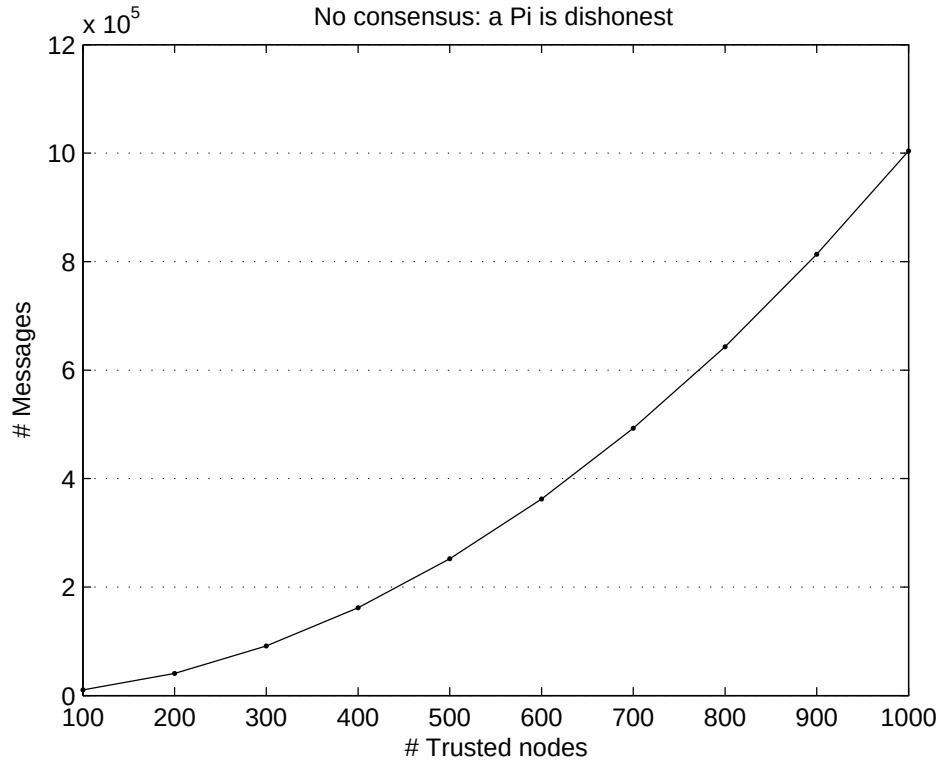


Figure A.2: Communication cost of public key authentication when a P_i is dishonest.

Now, the cost of authenticating a public key increases quadratically with respect to the number of trusted nodes used by B (see Fig. A.2.) Moreover, note that either the computational or the communication cost keeps constant having one or more malicious participants. The elimination of several nodes in the trusted group requires, in terms of communication cost, exactly the same effort than the elimination of a unique node, since proofs are equally verified

once the Byzantine fault is announced to the group.

```
(1)  Enter the number of nodes in trusted group: 5
(2)
(3)  Creating node B...
(4)  Creating node A...
(5)  Creating node P0...
(6)  Creating node P1...
(7)  Creating node P2...
(8)  Creating node P3...
(9)  Creating node P4...
(10)
(11) Scenario:
(12)
(13) B's trusted are: P0 P1 P2 P3 P4
(14)
(15) Honest behavior of A (yes/no): yes
(16)
(17) Enter the name of malicious nodes in the trusted group and press ENTER: P2 P3
(18)
(19) -Beginning of the protocol-
(20)
(21) -No consensus: Byzantine Agreement-
(22)
(23) ..............................
(24) Node P3 incorrectly signs the challenge message
(25) B deletes P3 from its trusted group
(26)
(27) Node P2 incorrectly signs the challenge message
(28)
(29) B deletes P2 from its trusted group
(30) P0 deletes P2 from its trusted group
(31) P3 deletes P2 from its trusted group
(32) P4 deletes P2 from its trusted group
(33) P0 deletes P3 from its trusted group
(34) P2 deletes P3 from its trusted group
(35) P1 deletes P3 from its trusted group
(36) P1 deletes P2 from its trusted group
(37) P4 deletes P3 from its trusted group
(38)
(39) The total number of messages sent during authentication is 48
(40)
(41) Messages in C:\PKAuthentication\src\result.log
```

Figure A.3: Execution of an scenario where two participant nodes are dishonest when authenticating a public key.

Figure A.3 shows the output of executing a particular scenario for Case 2. We can see in line (1) the establishment of the environment, set in 5 participants nodes P_0, P_1, P_2, P_3, P_4. From line (3) to (9) the toolkit prints traces of the creation of nodes. In line (13) we can set which nodes will be used by B to authenticate A's public key. Moreover, the system allows us to specify whether any node has a malicious behavior. When the challenge-response process ends, the system returns a result message. In this case, there is no consensus and the protocol reaches the Byzantine stage (see line (21).) Finally, from line (24) to (37), B and his selected trusted nodes remove

the detected malicious nodes from their local databases. As an additional result, the system has stored the number of transmitted messages during this simulation. Moreover, we collect the run-time messages in a log file, which we use later in order to analyze the communication overhead of this protocol.

A.2 Computational Overhead

The computational effort required to each node is very low, and it is determined by the complexity of the cryptographic algorithms used for message decryption. Obviously, the more the number of messages nodes must decrypt, the more effort the protocol will require. In particular, the Byzantine agreement stage incurs the maximum cost. Table A.1-(a) summarizes the time sequence, the number of cryptographic operations and the computational complexity for each stage of the protocol. In order to compute this cost, we can also use the speed benchmarks shown in Table 3.1 for the cryptographic algorithms used in a protocol execution.

Finally, Figure A.1-(b) presents the computational time required by the protocol operations. As it is pointed out in the figure, authenticating a public key takes less than 1 minute with the cooperation of 600 nodes. Note that the Byzantine agreement stage supposes the greatest impact on the entire protocol.

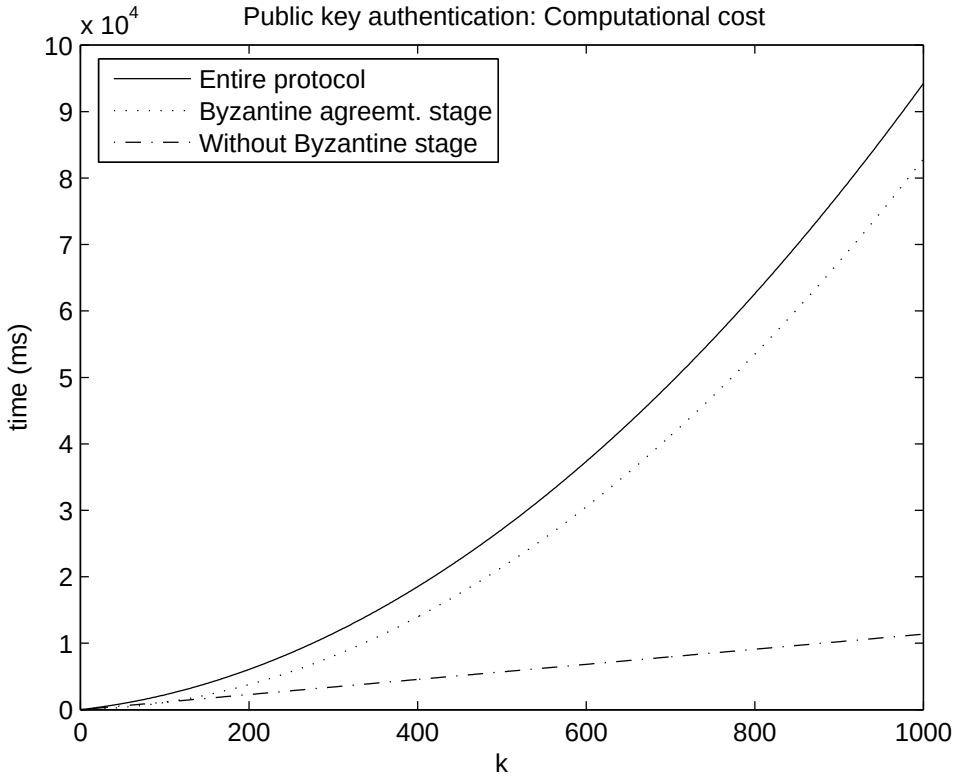
Moreover, as similar protocols implementing distributed authentication, the implemented public key authentication protocol is expensive in messaging cost. However, authors, in their analysis, try to reduce this cost using an epidemic algorithm called *Public key infection* for lazy propagation of protocol messages. By means of this improvement, the complexity gets reduced to $\mathcal{O}(k \log k)$. Readers interested in a more detailed description should refer to [97].

The implementation of this protocol is a main building block for evaluating the suitability of our protocol in real environments. In particular, the simulation of our content authentication experiments along with several scenarios for public key authentication serve us for actually measuring the computation and communication effort that real nodes have to spend.

Stage	No. crypto operations	Complexity
1. Admission request	$2H + 2S + 1V$	
2. Challenge response	$(kV + kN + kE) + 2kH + 2kS+$ $+(kD + kV)$	$\mathcal{O}(k^2 \log k)$
3. Distributed authentic.	kS	
4. Byzantine agreement	$(k+1)H + (k+1)S + k(k+1)V$	
Total:	$k(E + D + N) + 3(k+1)H+$ $+(4k+3)S + (k^2 + 3k + 1)V$	

Legend: H: Hash generation; E: Asymmetric encryption; D: Asymmetric decryption; S: Signature generation; V: Signature verification; k: No. of participants; N: Nonce generation;.

(a) Computational complexity for each stage of public key authentication.



(b) Computational cost of public key authentication.

Table A.1: Public key authentication: Efficiency analysis (computational effort).

Appendix B

Experimenting with Cryptographic Puzzles

Among the possible ways in which rationality can be ensured, we have considered computational puzzles as incentives for users to behave in a uniform fashion, and discourage free riders. In particular, our access control protocol, concretely Join subprotocol, is based on such a kind of proof-of-work mechanism in order to evaluate requesters' collaboration nature. Thus, cryptographic puzzles serve owners as challenges for measuring requesters' interest.

This appendix is devoted to present some analysis concerning the efficiency and security of our proposal through a number of scenarios. Experimental results are reported to illustrate the computational consumption and the time spent by the protocol according to several cryptographic algorithms.

B.1 Simulation Framework

Apart from the mechanisms mentioned in Section 4.3.1.1, there exist other possibilities in order to use cryptographic primitives for building up similar puzzles (e.g. hash functions in which a preimage of a given value have to be found.)

We have considered two block ciphers as the basis for cryptographic puzzles, AES-128 and TEA [36, 103]. Both were coded in C, compiled with Microsoft Visual C++, and ran on a AMD ATHLON(tm)2600 2.09GHz processor, 1GB RAM under Windows XP SP2.

The factor responsible for the complexity is the cost of executing several

decryptions, testing each candidate key. We have carried out 1000 experiments for different values of l-bits $(0 \ldots 2^{32})$, and also randomly varying the challenges and key used. We used the Mersenne Twister algorithm [85] for generating uniform pseudorandom numbers.

It is interesting to examine to what extent the usage of this kind of effort-aware access control can be applicable and reliable in real networks. First, we have considered that more than 32 hidden bits would result in a impracticable number of potential keys to test in a common platform.

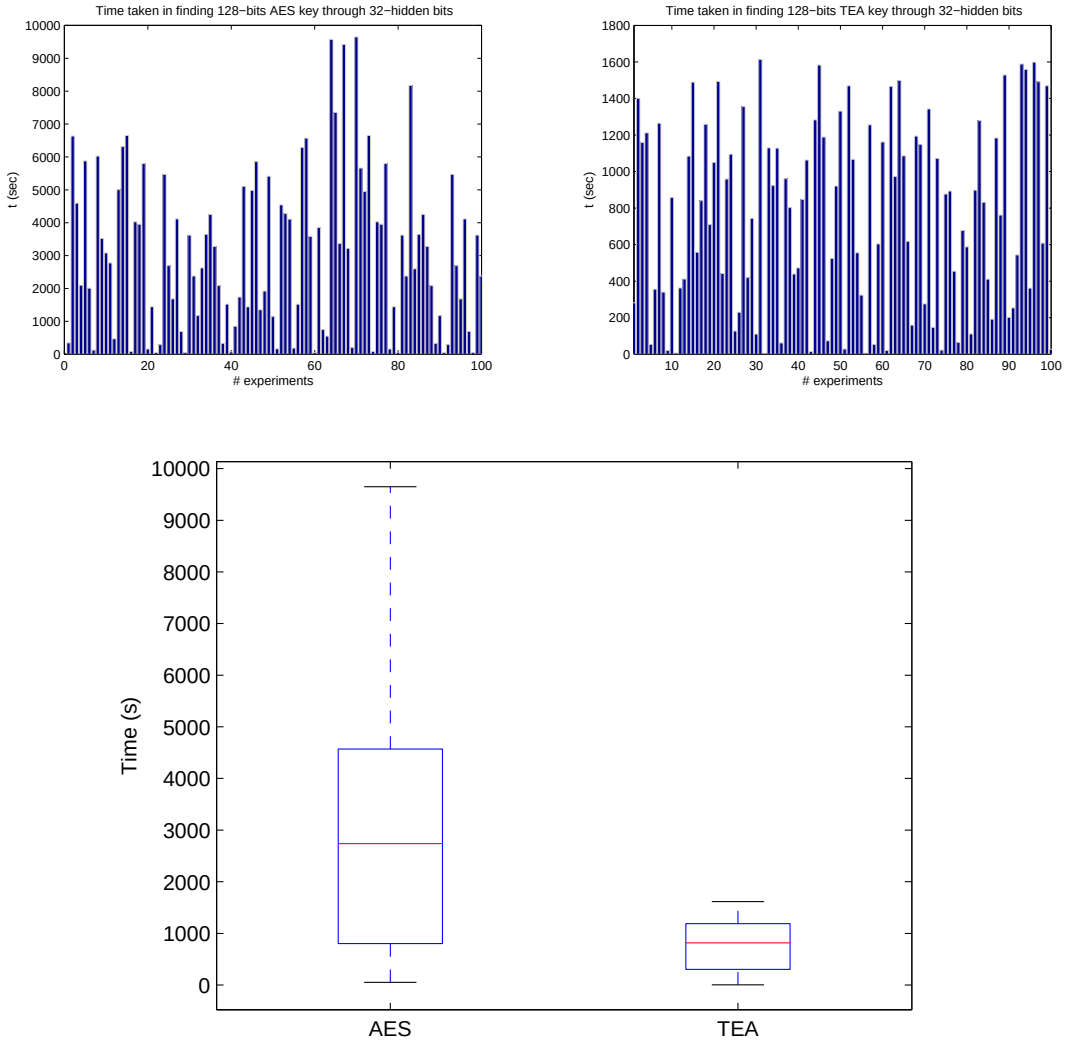


Figure B.1: AES (left) and TEA (right) computational consumption in seconds to find the corresponding K_S through a 32-bits trapdoor in a sample of 100 experiments.

ω bits	l bits	**Time required on average (sec)**	
		AES	TEA
32	96	5495	761
28	100	544	47
24	104	15	0.22
20	108	0.01	0.01

Table B.1: Computational effort spent on average (in 1000 experiments) by each algorithm regarding the amount of given bits.

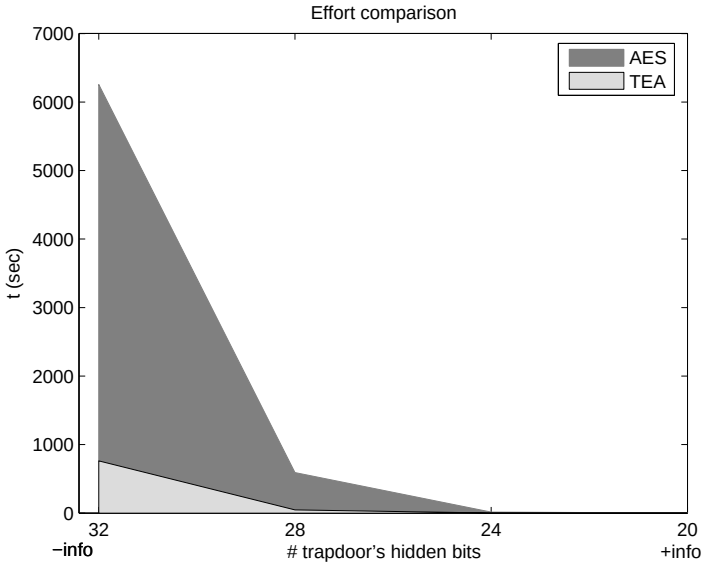


Figure B.2: Comparison of both algorithms' computational effort spent on average (in 1000 experiments) regarding the amount of given information of the encryption key.

The average results are shown in Table B.1. For each case, as the l threshold increases, the number of candidate keys obviously decreases, so the exploration time too. Fig. B.1 presents the exploration effort using both algorithms and trapdoors with 32 hidden bits, in a sample of 100 experiments. In this case, AES takes 90 minutes with 32-trapdoor bits, while TEA takes 27 minutes, on average. However, this time decreases to 9 minutes and 0.7 minutes, respectively, when providing 4 additional bits. Therefore, we can consider them good examples of puzzles for our work-aware access control model. Contrarily, with 2^{64} large number of potential keys to test, it literally takes a matter of years. Fig. B.2 compares both schemes in terms of different amount of supplied l bits.

The complexity increases when the given l key bits are not organized in the same key blocks, but randomly.

In summary, experiments show us signs of the reliability of the proposal according to users' computational resources and interests. We, as many authors, try to demonstrate that peers tend to collaborate even if they must spend some resources, and playing down the importance of indirect trust. Therefore file sharing can be encouraged by imposing a cost on the downloads, where each transaction implies to be worthy of access each time.